ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

Τμήμα Μηχανικών Ηλεκτρονικών Υπολογιστών

Τηλεπικοινωνιών και Δικτύων

ΘΕΜΑ:

ΕΡΓΑΛΕΙΑ ΑΝΑΠΤΥΞΗΣ ΕΦΑΡΜΟΓΩΝ ΓΙΑ ΤΟΝ ΠΑΓΚΟΣΜΙΟ ΙΣΤΟ

Φοιτητής: Γλυκύς Νικόλας

ΑΕΜ 612

nikosglikis@gmail.com

ΕΠΙΒΛΕΠΟΝΤΕΣ ΚΑΘΗΓΗΤΕΣ:

Κατσαρός Δημήτριος

Μποζάνης Παναγιώτης

Περιεχόμενα

Contents

Ευχαριστίες

Θερμές ευχαριστίες εκφράζω στους καθηγητές μου κ Δημήτριο Κατσαρό και κ Παναγιώτη Μποζάνη για την επίβλεψη, καθοδήγηση και βοήθεια που μου παρείχαν για την εκπόνηση της πτυχιακής μου εργασίας.

Περίληψη

Στην παρούσα πτυχιακή μελετάται και υλοποιείται η ανάπτυξη εργαλείων που επιτρέπουν τη γρήγορη ανάπτυξη ιστοσελίδων και εργαλείων για τον παγκόσμιο ιστό. Θα παρουσιαστούν τα εργαλεία και οι τεχνολογίες που χρησιμοποιήθηκαν καθώς και οι αρχιτεκτονικές και μοτίβα(patterns) που ακολουθούνται για την υλοποίηση. Η τελική υλοποίηση θα γίνει με τη μορφή ενός MVC (Model View Controller) Framework που θα περιέχει τα κατάλληλα εργαλεία και θα επιτρέπει ταχεία ανάπτυξη εφαρμογών για τον παγκόσμιο ιστό, σε σύγκριση με τις άλλες μεθόδους. Επίσης θα αναπτυχθεί ένα Σύστημα Διαχείρισης Περιεχομένου (CMS, Content Management System) που θα χρησιμοποιεί το παρόν Framework. Τέλος θα παρουσιαστεί η υλοποίηση της εφαρμογής xyz ως παράδειγμα χρήσης του Framework-CMS.

ΚΕΦΑΛΑΙΟ 1

1 Εισαγωγή

1.1 Ορισμός του προβλήματος

Δημιουργία ενός framework το οποίο να επιτρέπει τη δημιουργία ιστοσελίδων και διαδικτυακών εφαρμογών σε πολύ σύντομο χρόνο και να επιτρέπει εύκολη συντήρηση και γρήγορες αλλαγές στην εφαρμογή. Επίσης να επιτρέπει διαχωρισμό του κώδικα λογικής από τον κώδικα που αφορά στην εμφάνιση της ιστοσελίδας. Επίσης θέλουμε ένα βασικό σύστημα διαχείρισης περιεχομένου που να είναι κτισμένο πάνω στο framework ώστε να μπορεί ο προγραμματιστής να έχει κάποιες πολύ συνηθισμένες λειτουργίες έτοιμες ώστε να βασιστεί πάνω τους για τη δημιουργία της εφαρμογής.

1.2 Λύση του προβλήματος και σκοπός εργασίας

Για να λύσουμε το πρόβλημα θα δημιουργήσαμε ένα MVC framework βασισμένο στο MVC (Model View Controller) μοντέλο που θα περιέχει όλα τα εργαλεία που θα επιτρέπουν γρήγορη ανάπτυξη εφαρμογών (εύκολη πρόσβαση σε βάση δεδομένων, config system, υποστήριξη caching, υποστήριξη πολυγλωσσικού συστήματος, εργαλεία για ασφάλεια, σύστημα registry (εύκολη αποθήκευση και ανάκληση μεταβλητών συστήματος), routing (δημιουργία aliases για urls), εύκολο σύστημα οργάνωσης κώδικα σε component-module και εύκολος τρόπος εγκατάστασης-απεγκατάστασης των modules).

Το MVC μοντέλο στηρίζεται στο διαχωρισμό του κώδικα σε 3 κατηγορίες

- Κώδικας που αφορά τη λογική της εφαρμογής
- Κώδικας που είναι υπεύθυνος για την εμφάνιση της εφαρμογής και
- Κώδικας που είναι αφορά τα δεδομένα, (σύνδεση σε βάσεις δεδομένων, API, αρχεία δεδομένων κλπ)

Αυτός ο διαχωρισμός επιτρέπει εύκολη συντήρηση του κώδικα, και ταυτόχρονα επιτρέπει σε πολλούς προγραμματιστές/designers να δουλεύουν στο ίδιο project χωρίς προβλήματα.

Επίσης αναπτύσσεται ένα βασικό CMS (Content Management System, Σύστημα Διαχείρισης Περιεχομένου) που θα προσφέρει στον χρήστη κάποια βασικά πράγματα όπως διαχείριση χρηστών, σύστημα authentication των χρηστών, εγκατάσταση απεγκατάσταση modules. Επίσης το CMS θα λειτουργεί και σαν σκελετός για περαιτέρω ανάπτυξη των εφαρμογών του κάθε προγραμματιστή.

1.3 Υπάρχουσες λύσεις

Υπάρχουν διάφορα συστήματα που μπορούν να μας βοηθήσουν για τη λύση του προβλήματός μας.

Τα κυριότερα είναι:

Joomla

Το Joomla είναι ένα ανοιχτού κώδικα ελεύθερο λογισμικό-σύστημα διαχείρισης περιεχομένου (CMS). Είναι γραμμένο σε php και χρησιμοποιεί τις αντικειμενοστραφείς δυνατότητες της γλώσσας και όπως και αυτή η εργασία είναι βασισμένη στη MVC αρχιτεκτονική.

Τα θετικά του Joomla είναι ότι έχει μια τεράστια κοινότητα, και έχει πάρα πολλά έτοιμα components, οπότε κάποιος developer χρειάζεται να αναπτύξει κάποιο νέο component πολύ σπάνια και σε πολύ εξειδικευμένες εφαρμογές. Έχει προεγκαταστημένο σύστημα διαχείρισης χρηστών και άρθρων/κατηγοριών και είναι σχετικά ασφαλές και η ομάδα ανάπτυξης του είναι ενεργή.

Τα αρνητικά του δεν είναι πολλά αλλά είναι απαγορευτικά σε πολλές περιπτώσεις. Είναι κάπως δύσκολο για ένα νέο προγραμματιστή να μάθει να αναπτύσσει modules, αλλά το πιο απαγορευτικό είναι το γεγονός ότι είναι ένα εξαιρετικά αργό σύστημα και είναι απλά αδύνατο να φτιάξει κάποιος μια ιστοσελίδα που θα έχει πολλούς χρήστες με Joomla.

CakePHP

Το cakePHP είναι ένα πολύ δυνατό εργαλείο για ανάπτυξη εφαρμογών. Είναι και αυτό γραμμένο σε PHP και ακολουθεί την αρχιτεκτονική. Περιέχει όλα τα απαραίτητα εργαλεία για ανάπτυξη οποιασδήποτε εφαρμογής και είναι υπερπλήρες σε αυτό τον τομέα. Έχει εξαιρετικό σύστημα Scaffold. Αυτό με απλά λόγια σημαίνει ότι έχει αυτόματα

εργαλεία που μπορούν να κτίσουν ένα σκελετό της εφαρμογής, αφού τους δώσεις input και πληροφορίες σχετικά με τους τύπους και τη σχέση των tables στη βάση δεδομένων. Δημιουργεί αυτόματα controllers models και views και δημιουργεί ακόμα και τις φόρμες για εισαγωγή δεδομένων. Το πρόβλημα με αυτή την προσέγγιση είναι ότι κρύβει πολλά από την υλοποίηση και δεν είναι τόσο ευέλικτο, και αυτό συνήθως δεν αρέσει στους προγραμματιστές.

Επίσης ο κώδικας είναι πολύ αργός, ακόμα πιο αργός και από το Joomla. Είναι κατάλληλο για μικρές εφαρμογές αλλά δεν είναι κατάλληλο για τους σκοπούς αυτή της εργασίας, όπου ένας από τους στόχους είναι η εξαιρετική απόδοση.

Επίσης δεν υπάρχει κάποιο είδος συστήματος διαχείρισης προεγκαταστημένο για όποιον ενδιαφέρεται να το χρησιμοποιήσει ως βάση για να συνεχίσει με την υλοποίησή του.

Codeigniter

Το Codeigniter είναι και αυτό ένα PHP Framework φτιαγμένο με MVC αρχιτεκτονική. Είναι εξαιρετικά πιο γρήγορο και από τα προηγούμενα και αφήνει πολλή ελευθερία στον προγραμματιστή. Υστερεί σε λειτουργικότητα των δύο προηγούμενων συστημάτων, αλλά είναι πολύ δημοφιλές εξαιτίας της εκπληκτικής του απόδοσης (είναι δύο φορές πιο γρήγορο από το δεύτερο πιο γρήγορο Framework που υπάρχει, το ZendFramework).

Υπάρχει επιλογή να απενεργοποιήσεις αχρείαστα modules για καλύτερη απόδοση.

Όπως και στο CakePHP το σύστημα αποτελείται μόνο από το Framework και δεν έχει κάτι άλλο πέρα από αυτό. Η αρχιτεκτονική του συστήματος μας δανείζεται πολλά στοιχεία και μερικές φορές ακόμα και κώδικα από το CodeIgniter (το Active Record σύστημα που χρησιμοποιήθηκε σε αυτή δανείζεται πολλά κομμάτια κώδικα από το CodeIgniter)

1.4 Δομή της Εργασίας

Η πτυχιακή αυτή εργασία ακολουθεί μια δομή που αποτελείται από 5 κεφάλαια. Το πρώτο κεφάλαιο αποτελεί την «Εισαγωγή» και

περιλαμβάνει τον ορισμό του προβλήματος και τη λύση στο πρόβλημα αυτό που αποτέλεσε και το σκοπό της εργασίας. Σε αυτό το κεφάλαιο αναφέρονται οι λόγοι που οδήγησαν στη συγγραφή της πτυχιακής αυτής και αναλυτικά ποιους στόχους έχει σκοπό να επιτύχει με την ολοκλήρωσή της.

Στο δεύτερο κεφάλαιο γίνεται λόγος για τις τεχνολογίες και εργαλεία που χρησιμοποιήθηκαν για την υλοποίησή του. Οι τεχνολογίες αυτές περιλαμβάνουν τις γλώσσες HTML, PHP και SQL για τη μορφοποίηση, τη διαμόρφωση των λειτουργιών και τη διαμόρφωση της βάσης δεδομένων αντίστοιχα. Από την άλλη πλευρά τα εργαλεία που χρησιμοποιήθηκαν περιλαμβάνουν τον Apache server (διακομιστή παγκόσμιου ιστού), το σύστημα διαχείρισης σχεσιακών βάσεων δεδομένων MySQL (διακομιστή), το εργαλείο phpMyAdmin γραμμένο σε PHP για τη διαχείριση του MySQL και τέλος τα εργαλεία Visual Paradigm και Pacestar Uml Diagrammer για τη δημιουργία των εξής: σχήματος σχέσης, οντότητας σχέσης και διαγράμματος περιπτώσεων χρήσεως.

Το τρίτο κεφάλαιο αναφέρεται στη σχεδίαση του συστήματος και της αρχιτεκτονικής του. Περιγράφονται αναλυτικά οι λειτουργικές απαιτήσεις του συστήματος.

Το τέταρτο κεφάλαιο αναφέρεται στην υλοποίηση του συστήματος. Περιγράφονται αναλυτικά πως υλοποιήθηκε κάθε λειτουργία.

Το πέμπτο κεφάλαιο περιλαμβάνει μια ανακεφαλαίωση, συμπεράσματα και μελλοντικές προτάσεις για την υλοποίηση και την επέκταση του συστήματος.

Το Παράρτημα Α δείχνει την εγκατάσταση και την ρύθμιση των Permissions ενός καινούργιου Module στο CMS καθώς επίσης και την χρήση του.

ΚΕΦΑΛΑΙΟ 2

2 Εργαλεία τεχνολογίες αρχιτεκτονικές και patterns που χρησιμοποιήθηκαν

2.1 PHP

Η PHP είναι μια γλώσσα προγραμματισμού/script για τη δημιουργία σελίδων web με δυναμικό περιεχόμενο. Μια σελίδα PHP περνά από επεξεργασία από ένα συμβατό διακομιστή του Παγκόσμιου Ιστού (web server) (π.χ. Apache), ώστε να παραχθεί σε πραγματικό χρόνο το τελικό περιεχόμενο, που θα σταλεί στο πρόγραμμα περιήγησης των επισκεπτών σε μορφή κώδικα HTML. Η PHP ήταν αρχικά ακρωνύμιο του Personal Home Page αλλά άλλαξε σύμφωνα με τη σύμβαση GNU και τώρα είναι το ακρωνύμιο του PHP Hypertext Preprocessor ενώ η τρέχουσα έκδοση που χρησιμοποιείται κυρίως είναι η 5 η οποία χρησιμοποιήθηκε για τις ανάγκες της πτυχιακής αυτής εργασίας.

Πλεονεκτήματα της PHP

Παρακάτω αναφέρονται επιγραμματικά τα σημαντικότερα πλεονεκτήματα της PHP:

- Υψηλή απόδοση: Η PHP είναι πολύ αποτελεσματική. Με ένα φθηνό διακομιστή (server) μπορούν να εξυπηρετηθούν εκατομμύρια επισκέψεων καθημερινά. Εάν υπάρχουν πολλοί διακομιστές τότε η χωρητικότητα μπορεί να είναι απεριόριστη.

- Διασυνδέεται με πολλά διαφορετικά συστήματα βάσεων δεδομένων: Η PHP έχει δικές της συνδέσεις με πολλά συστήματα βάσεων δεδομένων. Εκτός από τη My SQL μπορεί να συνδεθεί κατευθείαν με τις βάσεις δεδομένων PostgreSQL, mSQL, Oracle, dbm, filePro, Hyperware, Infomix, Interbase, SQLite και Sybase.

- Ενσωματωμένες βιβλιοθήκες για πολλές συνηθισμένες web διαδικασίες: Η PHP σχεδιάστηκε για να χρησιμοποιείται στο web και για το λόγο αυτό έχει πολλές ενσωματωμένες βιβλιοθήκες που εκτελούν πολλές χρήσιμες λειτουργίες σχετικά με το web. Δίνεται η δυνατότητα για δημιουργία εικόνων GIF δυναμικά, σύνδεση με

άλλες υπηρεσίες δικτύων, ανάλυση XML, αποστολή ηλεκτρονικού ταχυδρομείου και άλλα μόνο με λίγες γραμμές κώδικα.

- Κόστος: Η PHP είναι δωρεάν.
- Ευκολία μάθησης και χρήσης: Η σύνταξη της PHP βασίζεται σε άλλες γλώσσες προγραμματισμού όπως της C και της Perl.
- Δυνατή αντικειμενοστραφής υποστήριξη: Η έκδοση 5 έχει πολύ καλά σχεδιασμένες αντικειμενοστραφείς λειτουργίες.
- Μεταφερσιμότητα: Η PHP είναι διαθέσιμη για πολλά διαφορετικά λειτουργικά συστήματα. Μπορεί να γραφεί κώδικας php ο οποίος λειτουργεί χωρίς αλλαγές για δωρεάν συστήματα τύπου unix (linux, FreeBSD), για εμπορικές εκδόσεις του unix (Solaris, IRIX) ή για διαφορές εκδόσεις των Microsoft Windows.
- Διαθεσιμότητα κώδικα προέλευσης (source code): Υπάρχει η δυνατότητα για πρόσβαση στον κώδικα προέλευσης της PHP. Με την PHP αντίθετα από τα εμπορικά κλειστού κώδικα προϊόντα αν υπάρχει κάτι που θέλετε να αλλάξετε ή να προσθέσετε στη γλώσσα μπορείτε να το κάνετε.
- Διαθεσιμότητα υποστήριξης: Η Zend Technologies, η εταιρεία πίσω από τη μηχανή που στηρίζει την PHP χρηματοδοτεί την ανάπτυξή της προσφέροντας υποστήριξη και σχετικό λογισμικό σε εμπορική βάση.

2.2 MySQL

Η MySQL είναι ένα πολύ γρήγορο δυνατό σύστημα διαχείρισης σχεσιακών βάσεων δεδομένων. Μια βάση δεδομένων επιτρέπει να αποθηκεύετε, αναζητάτε, ταξινομείτε και να ανακαλείτε τα δεδομένα σας αποτελεσματικά. Ο MySQL server ελέγχει την πρόσβαση στα δεδομένα σας για να διασφαλίσει ότι πολλοί χρήστες μπορούν να δουλέψουν ταυτόχρονα, για να παρέχει πρόσβαση και για να διασφαλίσει ότι μόνο οι πιστοποιημένοι χρήστες θα μπορούν να έχουν πρόσβαση. Συνεπώς, η MySQL είναι ένας πολυνηματικός διακομιστής πολλαπλών χρηστών. Χρησιμοποιεί την SQL την τυπική γλώσσα ερωτημάτων για βάσεις δεδομένων. Η MySQL είναι διαθέσιμη από το 1996 αλλά η ιστορία της

ξεκινά από το 1979.Είναι παγκοσμίως η πιο δημοφιλής βάση δεδομένων ανοικτού κώδικα και έχει κερδίσει το βραβείο Choise Award του Linux Journal Readers αρκετές φορές.

Η MySQL είναι τώρα διαθέσιμη με άδειες δυο ειδών. Μπορείτε να την χρησιμοποιήσετε δωρεάν με άδεια ανοικτού κώδικα (open source -GPL), εφόσον συμφωνήσετε με τους όρους αυτής της άδειας.

Η MySQL έχει πρόσφατα αγοραστεί από την Sun (η εταιρεία που βρίσκεται πίσω από την Java) η οποία με τη σειρά της αγοράστηκε από την ORACLE μια εταιρεία τις οποίας κύριο προϊόν είναι το ORACLE DATABASE. Αυτό το προϊόν είναι ένα commercial σύστημα διαχείρισης δεδομένων (δεν είναι δωρεάν), παρόλα αυτά η MySQL είναι ακόμα λογισμικό ανοιχτού κώδικα, διανέμεται δωρεάν και συνεχίζει να αναπτύσσεται όπως πριν.

Πλεονεκτήματα της MySQL

Η MySQL έχει πολλά πλεονεκτήματα όπως:

- Υψηλή απόδοση: Η MySQL είναι χωρίς αμφιβολία γρήγορη. Πολλές δοκιμές δείχνουν ότι η MySQL είναι ακόμη πιο γρήγορη από τον ανταγωνισμό.

- Χαμηλό κόστος: Η MySQL είναι δωρεάν με άδεια ανοικτού κώδικα ή με χαμηλό κόστος αν πάρετε εμπορική άδεια. Χρειάζεται μια άδεια αν θέλετε να διανείμετε την MySQL μαζί με την εφαρμογή σας και δε θέλετε η εφαρμογή σας να είναι ανοικτού κώδικα.

- Εύκολη διαμόρφωση και εκμάθηση: Οι περισσότερες σύγχρονες βάσεις δεδομένων χρησιμοποιούν SQL. Αν έχετε χρησιμοποιήσει ένα άλλο σύστημα διαχείρισης βάσεων δεδομένων θα είναι πολύ εύκολο να προσαρμοστείτε με αυτή καθώς η MySQL είναι ευκολότερη από παρόμοια προϊόντα και χρησιμοποιεί το πρότυπο SQL.

- Μεταφερσιμότητα : Η MySQL μπορεί να χρησιμοποιηθεί σε πολλά διαφορετικά συστήματα UNIX όπως επίσης και στα Microsoft Windows.

- Διαθεσιμότητα κώδικα προέλευσης (source code): Όπως και με την PHP ο κώδικας προέλευσης MySQL μπορεί να τροποποιηθεί.

- Διαθεσιμότητα υποστήριξης: Δεν έχουν όλα τα προϊόντα ανοικτού κώδικα μια γονική εταιρεία που να προσφέρει υποστήριξη, εκπαίδευση, συμβουλές και πιστοποιητικά αλλά μπορείτε να πάρετε όλα αυτά από την MySQL AB.

2.3 Apache Server

Ο HTTP Apache server, που αναφέρεται συνήθως απλά ως Apache, είναι ένας ανοιχτού λογισμικού, εξυπηρετητής ιστού (HTTP web server) που έπαιξε ένα σημαντικό ρόλου στην αρχική επέκταση του World Wide Web. Ο Apache ήταν η πρώτη βιώσιμη εναλλακτική λύση στον Netscape Communications Corporation και έχει εξελιχθεί από τότε για να συναγωνιστεί άλλους βασισμένους σε Unix web server από άποψη λειτουργίας και απόδοσης.

Ο Apache αναπτύσσεται και διατηρείται από μια τεράστια κοινότητα προγραμματιστών, υπεύθυνων για την ανάπτυξη του, υπό την αιγίδα του ιδρύματος λογισμικού Apache. Η εφαρμογή είναι διαθέσιμη για μια ευρεία ποικιλία λειτουργικών συστημάτων, συμπεριλαμβανομένου του Unix, FreeBSD, Linux, Solaris, Novell NetWare, MAC OS X, Windows της Microsoft, OS/2, TPF, και eComStation. Απελευθερωμένο με άδεια Apache, Apache χαρακτηρίζεται ως ελεύθερο λογισμικό ανοιχτού κώδικα. Από τον Απρίλιο του 1996 ο Apache είναι ο HTTP server στο World Wide Web. Από τον Φεβρουάριο του 2011 ο Apache εξυπηρετεί 59.13% όλων των websites και το 66.62% των πιο πολυχρησιμοποιημένων websites.

Το Apache χρησιμοποιείται καταρχάς για να εξυπηρετήσει και στατικές και δυναμικές ιστοσελίδες στο World Wide Web. Πολλές εφαρμογές Ιστού σχεδιάζονται αναμένοντας το περιβάλλον και τα χαρακτηριστικά γνωρίσματα που παρέχει ο Apache . Ο Apache είναι το ο HTTP web server του δημοφιλούς LAMP (Linux Apache MySQL PHP), παράλληλα με τη MySQL, και τη γλώσσα προγραμματισμού PHP.

Ο Apache ανακατανέμεται ως τμήμα των διάφορων ιδιόκτητων πακέτων λογισμικού συμπεριλαμβανομένης της βάσης δεδομένων Oracle ή του web server IBM WebSphere. Υποστηρίζεται επίσης με κάποιο τρόπο από

τη Borland στα εργαλεία ανάπτυξης Kylix. Ο (IIS) της Microsoft είναι ο κύριος ανταγωνιστής του Apache. Μερικοί από τους μεγαλύτερους ιστοχώρους στον κόσμο οργανώνονται χρησιμοποιώντας Apache. Το front end των μηχανών αναζήτησης του Google είναι βασισμένο σε μια τροποποιημένη έκδοση του Apache, που ονομάζεται GWS (Google Web Server).

2.4 PhpMyAdmin

Το phpMyAdmin είναι ένα εργαλείο ανοιχτού λογισμικού που προορίζεται για να πραγματοποιήσει τη διαχείριση της MySQL μέσω του Διαδικτύου και του browser του χρήστη. Μπορεί να δημιουργήσει και να καταστρέψει τις βάσεις δεδομένων, να κάνει create/drop/alter των πίνακων και delete/edit/add των πεδίων τους, να εκτελέσει οποιαδήποτε δήλωση SQL, να διαχειριστεί τους χρήστες και τις άδειες, και να διαχειριστεί τα κλειδιά στα πεδία.

2.5 MVC

Το MVC είναι μία αρχιτεκτονική προγραμματισμού που βασίζεται στο διαχωρισμό κώδικα λογικής από τον κώδικα εμφάνισης και αλληλεπίδρασης με το χρήστη. Αυτή η αρχιτεκτονική αν υιοθετηθεί επιτρέπει καλύτερη οργάνωση του κώδικα, καθώς και εύκολη συντήρηση του κώδικα. Επίσης επιτρέπει σε πολλούς προγραμματιστές να δουλεύουν στο ίδιο σύστημα ταυτόχρονα, καθώς και ανάπτυξη και αποσφαλμάτωση κάθε κομματιού κώδικα ξεχωριστά.

Εδώ χρησιμοποιούμε την εξής παραλλαγή του MVC:

Το πρόγραμμα αποτελείται από τρεις βασικές κατηγορίες κώδικα:

- Τα ***Models*** που είναι υπεύθυνος για να αποθηκεύει να ανακαλεί και να τροποποιεί δεδομένα. Τα δεδομένα μπορεί να είναι αποθηκευμένα σε μία σχεσιακή βάση δεδομένων, σε κάποιο αρχείο, η ακόμα να είναι προσβάσιμα μέσω κάποιου απομακρυσμένου API. Συνήθως καλούνται από τον ***Controller***

- Τα ***Views*** είναι τα κομμάτια κώδικα που είναι υπεύθυνα για την εμφάνιση της εφαρμογής, και την αλληλεπίδραση με το χρήστη. Τυπικά στις εφαρμογές είναι το GUI της εφαρμογής, που στην

περίπτωση του web development αποτελείται καθαρά από HTML/CSS κώδικα.

- Οι ***Controller*** είναι αυτοί που παίρνουν το input από τα views, καλούν τους ***Models*** για να πάρουν τα απαραίτητα δεδομένα, τα επεξεργάζονται και τα παρουσιάζουν με τα κατάλληλα ***Views.***

Μία MVC εφαρμογή είναι ένα σύνολο από Controllers, Models και views που αλληλεπιδρούν μεταξύ τους.

Το MVC είναι ιδιαίτερα δημοφιλές στο web development όπου το input είναι οι παράμετροι GET και POST της εφαρμογής και παίρνουν τα δεδομένα συνήθως από μια βάση δεδομένων και τα εμφανίζουν στο τελικό χρήστη ή τα τροποποιούν.

Για του σκοπούς αυτής της εργασίας αναπτύσσουμε ένα Framework που είναι βασισμένο στο MVC.

2.6 Active Record

Το Active Record είναι ένα μοτίβο προγραμματισμού (Pattern) που αποθηκεύει δεδομένα σε σχεσιακές βάσεις δεδομένων. Το Active Record είναι ένας τρόπος να προσπελάσεις μία βάση δεδομένων. Ο κάθε πίνακας (table) της βάσης δεδομένων αντιπροσωπεύεται από μία κλάση σε μία γλώσσα προγραμματισμού. Οπότε κάθε εγγραφή είναι ένα αντικείμενο αυτής της κλάσης. Με τη δημιουργία ενός νέου αντικειμένου εισάγεται μια νέα εγγραφή στον πίνακα στη βάση δεδομένων. Όταν ένα αντικείμενο τροποποιείται το ίδιο γίνεται και στη βάση δεδομένων.

Η κλάση έχει κάποιες μεθόδους που βοηθούν με τις πιο πάνω λειτουργίες.

Υπάρχουν πολλές υλοποιήσεις PHP για το Active Record, και για αυτή την εργασία υιοθετούμε αυτή του Codeigniter, το οποίο δεν είναι ακριβώς Active Record αλλά υλοποίηση μιας τροποποιημένης έκδοσης του Active Record, που δίνει βάρος στην εύκολη ανάγνωση, τροποποίηση και εισαγωγή δεδομένων στη βάση δεδομένων με τον ελάχιστο κώδικα.

Με την υιοθέτηση του Active Record μπορεί η εφαρμογή να χρησιμοποιήσει διαφορετικά συστήματα διαχείρισης δεδομένων (πχ από MySQL σε SQL Server) με τον ίδιο κώδικα.

Κεφάλαιο 3

Σχεδιασμός

3.1 Ποιες λειτουργικές απαιτήσεις στόχευσε να υποστηρίξει το σύστημα

3.1.1 Λειτουργική απαίτηση 1 – Εύκολη και γρήγορη ανάπτυξη εφαρμογών.

Πρέπει το σύστημά μας να επιτρέπει την εύκολη και γρήγορη ανάπτυξη εφαρμογών. Πρέπει να προσφέρει τα κατάλληλα εργαλεία και να βασίζεται στις αρχές εκείνες που θα επιτρέπουν ελαχιστοποίηση του χρόνου ανάπτυξης του λογισμικού.

3.1.2 Λειτουργική απαίτηση 2 – Να επιτρέπει εύκολη συντήρηση και τροποποίηση κώδικα και εύκολη αλλαγή της εμφάνισης της εφαρμογής.

Το σύστημα θα πρέπει να επιτρέπει εύκολη συντήρηση κώδικα, καθώς και διαχωρισμό κώδικα λογικής, κώδικα εμφάνισης και κώδικα πρόσβασης δεδομένων. Στο web development είναι σημαντικό για μια εφαρμογή να μπορεί να αλλάζει εύκολα εμφάνιση (template). Το σύστημά μας θα πρέπει να μπορεί να επιτρέπει στον προγραμματιστή να το κάνει αυτό γρήγορα και αποδοτικά.

3.1.3 Λειτουργική απαίτηση 3 – Να επιτρέπει την ταυτόχρονη ανάπτυξη της εφαρμογής από πολλούς προγραμματιστές.

Θα πρέπει το framework μας να είναι φτιαγμένο έτσι ώστε να επιτρέπει σε μία ομάδα προγραμματιστών να δουλεύει ταυτόχρονα στην ανάπτυξη μιας εφαρμογής και η δουλεία του ενός να μην επηρεάζει τον άλλο.

3.1.4 Λειτουργική απαίτηση 4 – Αυτόματη πρόσβαση στη βάση δεδομένων

Θα πρέπει το σύστημα να έχει την απαραίτητη αφαίρεση ώστε να μπορεί ο ίδιος κώδικας που χρησιμοποιείται σε μία σχεσιακή βάση δεδομένων να μπορεί να χρησιμοποιηθεί και σε μία άλλη, όταν ο προγραμματιστής το θέλει. Επίσης θα προσφέρονται οι κατάλληλες εντολές από το Framework που να επιτρέπουν εύκολη και γρήγορη πρόσβαση στα στοιχεία στις βάσεις δεδομένων. Το σύστημα θα παρέχει config αρχείο

όπου ο χρήστης θα μπορεί να δίνει στοιχεία (Τύπος βάσης δεδομένων, username, password, όνομα βάσης δεδομένων, και το σύστημα θα χρησιμοποιεί αυτά κάθε φορά που θα χρειάζεται.

3.1.5 Λειτουργική απαίτηση 5 – Το σύστημα να είναι γρήγορο.

Θα πρέπει το σύστημα να είναι γρήγορο από τον ανταγωνισμό και να μπορεί να λειτουργήσει αποτελεσματικά σε low end συστήματα και να έχει καλή απόδοση.

3.1.6 Λειτουργική απαίτηση 6 – Search Engine friendly urls

Το σύστημα θα πρέπει να είναι έτσι φτιαγμένο ώστε να λειτουργεί με όμορφα URLS φιλικά προς τις μηχανές αναζήτησης πχ:

Http://www.domain.com/articles/read/1/News_headline

Και όχι

http://www.domain.com/index.php?module=articles&action=read&id=1

3.1.7 Λειτουργική απαίτηση 7 – Ασφάλεια κώδικα

Το framework θα πρέπει να είναι φτιαγμένο με τέτοιο τρόπο ώστε στον public φάκελο του web server να υπάρχουν μόνο τα html αρχεία εικόνες javascript και css αρχεία και όλος ο κώδικας php να είναι σε άλλο φάκελο του συστήματος που δεν είναι public στο διαδίκτυο.

3.1.8 Λειτουργική απαίτηση 8 – Single point of entry (Μοναδικό σημείο εισόδου)

Θα πρέπει όλες οι επισκέψεις του χρήστη να περνούν από το ίδιο αρχείο εισόδου, πχ index.php και μετά να φορτώνονται αυτόματα οι κλάσεις και ο άλλος κώδικας που χρειάζεται. Με αυτό τον τρόπο βελτιώνεται η ασφάλεια, ο προγραμματιστής μπορεί να θέσει κάποιες global μεταβλητές, ή να διαχειριστούν εύκολα πράγματα όπως γλώσσα παρουσίασης περιεχομένου, lightweight διαχείριση caching κλπ.

3.1.9 Λειτουργική απαίτηση 9 – Ασφάλεια

Θα πρέπει το framework να προσφέρει τα κατάλληλα εργαλεία στον προγραμματιστή για εργασίες όπως φιλτράρισμα του input των χρηστών, validation input, αποφυγή sql injections Κλπ. Επίσης θα πρέπει να προσφέρεται user system όπου θα υπάρχει η δυνατότητα οργάνωσης των

users σε διάφορα groups, και θα μπορεί ο χρήστης να καθορίζει την πρόσβαση του κάθε group στα διάφορα modules του συστήματος.

3.1.10 Λειτουργική απαίτηση 10 – Υποστήριξη modules

Πρέπει να υποστηρίζεται οργάνωση του κώδικα σε modules, τόσο λογικά όσο και σε επίπεδο συστήματος αρχείων. Κάθε module έχει απαραίτητα τον controller του και προαιρετικά όσους Models και Viewers χρειάζεται. Να υποστηρίζεται διαχωρισμός του κάθε module σε admin και Public κομμάτι. Το. Κάθε module θα έχει κάποια registered actions και μόνο σε αυτά τα actions θα εξυπηρετούνται από το σύστημα. Τα modules θα καλούνται με τον εξής τρόπο:

http://www.domain.com/controller/action/parametr1/parametr2.

Κάθε τέτοια κλήση θα αντιστοιχεί σε μία μέθοδο μίας κλάσης php ως εξής:

```
Class Controller

{

        Public function action($parametr1, $parametr2)

}
```

3.1.11 Λειτουργική απαίτηση 11 – Δημιουργία backend για τη βασική διαχείριση του συστήματος

Θα πρέπει να δημιουργηθεί ένα βασικό CMS (Content Management System) που θα βασίζεται στο framework και θα εξυπηρετεί σαν βάση για περαιτέρω ανάπτυξη εφαρμογών και Modules. Θα πρέπει να περιέχει ένα βασικό σύστημα διαχείρισης χρηστών και σύστημα οργάνωσης τους σε groups. Κάθε χρήστης μπορεί να ανήκει σε πολλά Groups.

Πρέπει να υπάρχει σύστημα διαχείρισης των Modules που θα επιτρέπει Install Uninstall και διαχείριση στην πρόσβαση των Modules από τα διάφορα Groups, τόσο σε Admin επίπεδο όσο και σε public. Επίσης θα υπάρχει δημιουργία αυτόματου menu στο admin κομμάτι του CMS ανάλογα με τα modules. Το install αρχείο του κάθε module θα δίνει στο σύστημα πληροφορίες για το ποιες μέθοδοι της κλάσης είναι «actions» και μπορούν να προσπελαστούν από τους χρήστες, και ποιες είναι

βοηθητικές συναρτήσεις. Επίσης θα δίνεται πληροφορία για το ποια από τα actions θα φαίνονται στο main menu της εφαρμογής.

3.1.12 Λειτουργική απαίτηση 12 – Υποστήριξη templates

Θα πρέπει τα views των Modules, και οι εικόνες αρχεία javascript css flash κλπ να είναι οργανωμένα σε φακέλους που αντιστοιχούν σε templates, και να καθορίζεται σε config αρχείο ποιο template είναι ενεργό και να χρησιμοποιούνται τα ανάλογα views που θα χρησιμοποιούν τα ανάλογα αρχεία εικόνων css κλπ από το φάκελο public. Θα πρέπει να υποστηρίζεται η αλλαγή του template μέσα από το config αρχείο.

Κεφάλαιο 4

Αρχιτεκτονική συστήματος

4.1 Γενικά

Γενικά το σύστημα έχει 3 βασικά είδη αρχείων κώδικα.

4.1.1 – Κώδικας εκκίνησης

Το πρώτο είδος είναι ο κώδικας bootstrap (κώδικας εκκίνησης) που αναλύει το url που καλείτε, και φορτώνει την αντίστοιχη κλάση και καλεί την μέθοδο που χρειάζεται με τις παραμέτρους που υπάρχουν στο url. Όλες οι κλήσεις της εφαρμογής μας περνούν από αυτό τον κώδικα. Εκεί γίνονται και άλλες εργασίες όπως το να θέτουμε διάφορες σταθερές του συστήματος, όπως το directory που είναι ο κώδικας, φορτώνονται τα config αρχεία με τις προτιμήσεις της κάθε εφαρμογής, ρυθμίζεται το mode που τρέχει η εφαρμογή (debugging ή normal), γίνονται κάποιες εργασίες για ασφάλεια και συμβατότητα όπως η αφαίρεση των magic quotes αν υπάρχουν (http://php.net/manual/en/security.magicquotes.php), όπως και αφαίρεση των register globals (http://php.net/manual/en/security.globals.php). Επίσης εκεί δηλώνεται και η function __autoload(). Η autoload είναι μια ειδική συνάρτηση που καλείται από την php αυτόματα κάθε φορά που δημιουργείται ένα αντικείμενο μιας κλάσης που δεν υπάρχει στα αρχεία που έχουν ήδη φορτωθεί από το σύστημα. Τέλος στον κώδικα εκκίνησης ενεργοποιείται και η συμπίεση του html output αν υποστηρίζεται από το σύστημα, και το σύστημα caching που υλοποιήθηκε.

4.1.2 – Κώδικας framework

Το δεύτερο είδος κώδικα είναι ο κώδικας του framework. Αυτός ο κώδικας είναι η βάση για την ανάπτυξη των εφαρμογών. Περιέχει αρχεία για όλες τις εργασίες που θα χρειαστεί ο χρήστης, όπως ασφάλεια, πρόσβαση στη βάση δεδομένων, caching, υποστήριξη γλώσσας κλπ. Σε αυτόν τον κώδικα περιέχεται και το σύστημα που αναγκάζει την εφαρμογή του MVC και καλεί αυτόματα τις άλλες κλάσεις όταν χρειάζονται. Εδώ επίσης υλοποιείται και σύστημα που διαχωρίζει τον κώδικα της εφαρμογής που είναι προαιρετική η χρήση του από το

προγραμματιστή, αλλά κάνει τον κώδικα πιο οργανωμένο και τη συντήρηση του κώδικα πιο εύκολη. Επίσης υπάρχει διαχωρισμός του κώδικα σε public και admin.

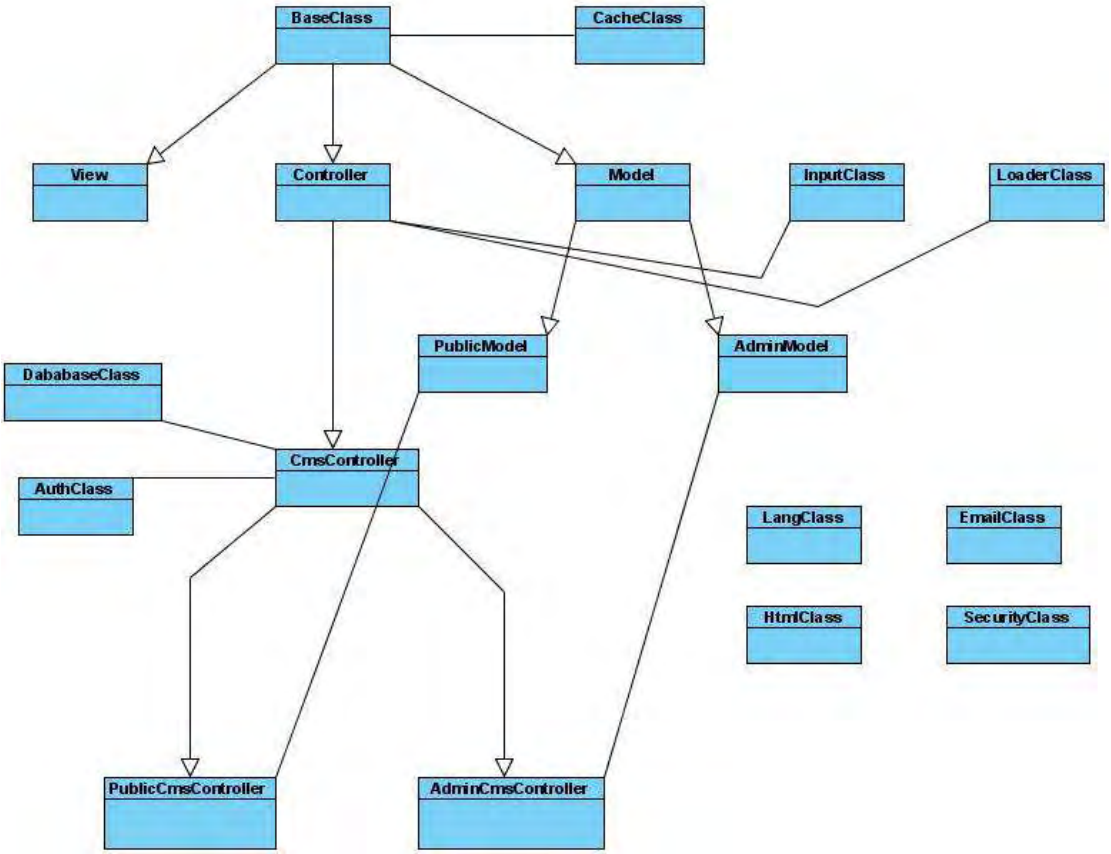
4.1.3 – Κώδικας CMS (Content Management System)

Το τρίτο είδος κώδικα είναι ο κώδικας που αποτελεί το CMS. Λειτουργεί σαν μια βάση για περαιτέρω ανάπτυξη της κάθε εφαρμογής. Προσφέρει μια βασική διαχείριση χρηστών και groups, authentication system, σύστημα που κάνει αυτόματα install τα modules, διαχείριση πρόσβασης του κάθε group σε κάθε action του κάθε module, τόσο σε public επίπεδο όσο και σε admin επίπεδο και αυτόματη διαχείριση menu. Δεν είναι υποχρεωτικό να χρησιμοποιηθεί από τον προγραμματιστή, αλλά είναι αρκετά ελαφρύ και προσφέρει σημαντικά εργαλεία που είναι απαραίτητα σε οποιαδήποτε εφαρμογή.

4.2 – Αρχιτεκτονική

4.2.1 Διάγραμμα κλάσεων

Ακολουθεί διάγραμμα με τις βασικές κλάσεις του συστήματος. Στο διάγραμμα υπάρχουν όλες οι κλάσεις, οι κλάσεις που ανήκουν στο Framework και οι κλάσεις που ανήκουν στο CMS.

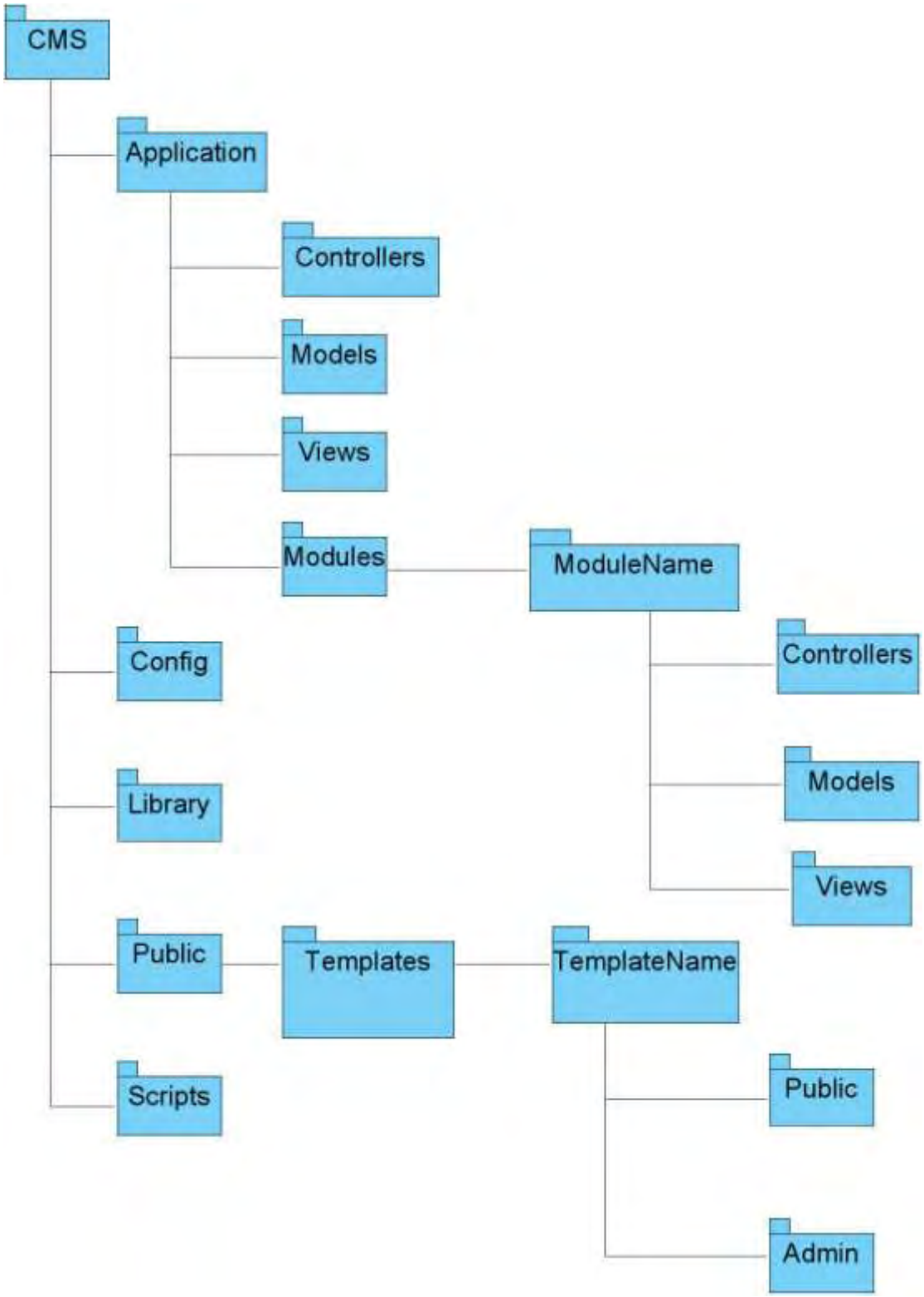


4.2.2. Επεξήγηση διαγράμματος

Στο διάγραμμα φαίνονται οι σχέσεις κληρονομικότητας των κλάσεων και η σχέση μεταξύ των διαφόρων κλάσεων. Όπως φαίνεται στο διάγραμμα υπάρχει μία βασική κλάση Controller που επεκτείνετε από την κλάση CmsController, που με την σειρά της επεκτείνετε από τις PublicCmsController και AdminCmsController. Η κλάση Controller περιέχει ένα instance της κλάσης Input που αναλαμβάνει το input του χρήστη, και ένα αντικείμενο της κλάσης Loader που χρησιμέυει στο να φορτώνει άλλες βοηθητικές κλάσεις, loader, views κλπ. Η CmsController είναι η βασική κλάση του CMS και περιέχει ένα αντικείμενο της κλάσης DatabaseClass και ένα αντικείμενο της κλάσης AuthClass

4.2.3 Διάγραμμα ιεραρχίας κώδικα

Το πιο κάτω διάγραμμα παρουσιάζει τον τρόπο με τον οποίο οργανώνονται τα αρχεία κώδικα του συστήματος.



Ο πρώτος φάκελος είναι ο φάκελος "**application**". Αυτός ο φάκελος περιέχει όλο τον κώδικα μιας συγκεκριμένης εφαρμογής. Ο προγραμματιστής έχει δύο επιλογές. Μπορεί να χρησιμοποιήσει τους φάκελους "controllers", "modules", "views" για να προγραμματίσει την εφαρμογή του, ή μπορεί να χωρίσει τον κώδικά του σε "Modules" που το

καθένα module περιέχει controllers models και views. Το σύστημα ψάχνει πρώτα στο φάκελο application/controllers για να βρει τον controller και μετά στο φάκελο modules.

Μετά υπάρχει ο φάκελος config που περιέχει τα αρχεία παραμετροποίησης της εφαρμογής. Εκεί ο προγραμματιστής δίνει διάφορα στοιχεία για την εφαρμογή, όπως το όνομα του template που θα χρησιμοποιηθεί, τα στοιχεία της βάσης δεδομένων, την default γλώσσα, το βασικό email της εφαρμογής, και τα options του κάθε module ξεχωριστά.

Ο φάκελος library είναι η καρδιά του συστήματος, περιέχει όλες τις κλάσεις, κύριες και βοηθητικές, όπως controller, model, cmscontroller,security,auth κλπ. Μερικές κλάσεις φορτώνονται αυτόματα οι υπόλοιπες μπορούν να φορτωθούν με την εντολή $this->load->library(“libraryname”) και να χρησιμοποιηθούν με το $this->libraryname->action(“parametrs”);

Ο φάκελος public περιέχει όλα τα αρχεία που είναι ορατά από το web. Τα αρχεία κώδικα ΔΕΝ είναι ορατά από το web με την παρούσα αρχιτεκτονική πράγμα που βοηθά και την ασφάλεια του συστήματος. Εδώ είναι όλα τα αρχεία εικόνας όπως jpg, png, gif, τα javascript και css αρχεία, flash κλπ.

Τα αρχεία αυτά μπορούν να είναι σε φάκελο ανάλογα με το template και να είναι χωρισμένα σε public και admin. Υπάρχει μεταβλητή στο config.php αρχείο που ορίζει ποιο είναι το active template και ο προγραμματιστής μπορεί να χρησιμοποιήσει αυτή τη μεταβλητή για να φορτώσει κάθε φορά το σωστό αρχείο από το σωστό φάκελο.

Τέλος είναι ο φάκελος scripts που περιέχει απλά αρχεία με βοηθητικό κώδικα που δεν είναι οργανωμένος σε κλάσεις.

Κεφάλαιο 5

Υλοποίηση

5.1 – Τρόπος λειτουργίας

Ο βασικός τρόπος λειτουργίας έχει ως ακολούθως:

- Ο χρήστης καλεί ένα url πχ:

  http://www.domain.com/articles/read/1/

- Το σύστημα ψάχνει για ένα controller με το όνομα articles και καλεί τη συνάρτηση read, αν υπάρχει και δίνει παράμετρο το “1”.

Ο προγραμματιστής για να χειριστεί αυτό κάνει τα εξής:

- Μέσα στο φάκελο application/controllers/ δημιουργεί ένα αρχείο με το όνομα articlescontroller.php που θα περιέχει την κλάση ArticlesControllers που θα πρέπει να επεκτείνει την κλάση Controller ή PublicCmsController

- Μέσα σε αυτή την κλάση δημιουργεί μία συνάρτηση function read($id){ } και βάζει μέσα τον κώδικα που χειρίζεται το request.

Όταν η λέξη κλειδί “admin” εμφανίζεται μέσα στο url μετά το domain (πχ http://www.domain.com/admin/articles/read/1/) τότε το σύστημα φορτώνει το admin μέρος του εκάστοτε module, ή ψάχνει για να βρει τον ArticlesAdminController για να τον τρέξει. Αυτό γίνεται για να υπάρχει διαχωρισμός του κώδικα σε admin για τους διαχειριστές και public για όλους τους υπόλοιπους, για περισσότερη ασφάλεια και περισσότερη ευελιξία στον κώδικα.

Μέσα από το CMS, ο administrator μπορεί να ορίζει ποια groups χρηστών θα έχουν πρόσβαση σε όλα τα actions (συναρτήσεις) των controllers, φτάνει αυτά να επεκτείνουν κάποια από τις κλάσεις PublicCmsController ή AdminCmsController).

Ο προγραμματιστής μέσα από το controller μπορεί να:

- Φορτώσει διάφορες βοηθητικές κλάσεις και να τις χρησιμοποιήσει με τις εξής εντολές:
    - $this->load->database(); και να χρησιμοποιήσει με το $this->database->query(…)
    - $this->load->view(“”) για να φορτώσει ένα κομμάτι του template.
    - $this->load->model(“ModelName”) για να φορτώσει κάποιο Model που μπορεί μετά να καλέσει με την εντολή: $this->ModelName->someFunction
    - Μπορεί να φορτωθεί μία βοηθητική κλάση από τη βιβλιοθήκη, για παράδειγμα security: $this->load->library(“security”) και να χρησιμοποιηθεί: $this->security->someFunction
    - Να φορτώσει μεταβλητές για το view με την εντολή $this->load->vars($somearray). Έστω ότι $somearray[‘title’] = “this is the title”;, μέσα στο view για να χρησιμοποιηθεί η μεταβλητή πρέπει να χρησιμοποιήσουμε $title και όχι $somearray[‘title’].
    - Ο προγραμματιστής μπορεί να πάρει το input του χρήστη (POST input) με την κλάση $this->input->post(“somevariable”). Αυτό επιστρέφει τα περιεχόμενα του πίνακα $_POST[‘somevariable’]. Με την κλάση input έχουμε και τις συναρτήσεις $this->input->post_databaseclean και $this->post_xssdatabaseclean() που επιστρέφουν το περιεχόμενο του πίνακα, αλλά καθαρό (από άποψη ασφάλειας) για εισαγωγή στη βάση δεδομένων, ή για εκτύπωση μαζί με το output της σελίδα.
- Ο προγραμματιστής μπορεί να χρησιμοποιήσει αυτή τη λειτουργικότητα ή οποιοδήποτε άλλο εργαλείο του συστήματος και σε άλλες κλάσεις εκτός από τους controllers, με τον ίδιο ακριβώς τρόπο.

5.2 – Αρχεία Κώδικα Framework

5.2.1 – library/controller.php

Αυτή είναι η βασική κλάση του συστήματος. Ο προγραμματιστής πρέπει να επεκτείνει τουλάχιστον αυτή την κλάση για να χρησιμοποιήσει το Framework. Περιέχει κάποιες βοηθητικές μεταβλητές που περιέχουν το όνομα του controller, τη συνάρτηση get_instance που επιστρέφει ένα δείκτη στο controller και χρησιμοποιείτε από σε διάφορα σημεία του κώδικα, και την function set που αλλάζει τις τιμές των μεταβλητών του controller.

Μέσα στον constructor δημιουργείται ένα αντικείμενο της κλάσης Input που είναι απαραίτητο για να χειριζόμαστε το input των χρηστών (Cookies POST και GET). Επίσης δημιουργείται και ένα αντικείμενο της κλάσης Loader που είναι υπεύθυνο για να φορτώνει τις διάφορες βοηθητικές κλάσεις. (Περισσότερη ανάλυση για την Input πιο κάτω).

5.2.2 – library/cmscontroller.php

Αυτό το αρχείο περιέχει την κλάση CmsController που είναι κλάση που χρησιμοποιείται για το Content Management System κομμάτι του συστήματος. Η διαφορά είναι ότι φορτώνονται ορισμένα εργαλεία που είναι απαραίτητα για το CMS. Αυτή η κλάση κληρονομείται από τις κλάσεις PublicCmsController και AdminCmsController.

Πιο συγκεκριμένα αρχικοποιείται το session, φορτώνεται η βιβλιοθήκη batabase και auth (σύνδεση σε βάση δεδομένων και authentication system για του χρήστες αντίστοιχα). Με την auth (authentication) βιβλιοθήκη έχουμε πρόσβαση σε πληροφορίες όπως το username του χρήστη και το group που ανήκει.

Μετά στον Constructor ελέγχεται αν ο χρήστης έχει πρόσβαση στο συγκεκριμένο controller και action. Αν δεν υπάρχει πρόσβαση το πρόγραμμα τερματίζει με μήνυμα λάθος, αν έχει πρόσβαση ο χρήστης στο συγκεκριμένο module και action συνεχίζει η εκτέλεση κανονικά. Επίσης ελέγχεται αν το Module είναι registered στο σύστημα.

Αν δεν έχουμε πληροφορία για το username του χρήστη, σημαίνει ότι ο χρήστης δεν είναι registered και τότε προωθείτε στην σελίδα του login.

5.2.3 – library/publiccmscontroller.php

Αυτή είναι η κλάση που κληρονομείται από όλα τα controllers που ανήκουν σε public κομμάτια των module.

Σε αυτή την κλάση ορίζεται ο φάκελος του template, ο οποίος είναι διαφορετικός ανάλογα με τον αν ο controller ανήκει σε public ή admin κομμάτι. Αυτή η μεταβλητή χρησιμοποιείται από τα views για να φορτώνονται οι εικόνες τα javascript αρχεία, flash κλπ. Αυτή η μεταβλητή περιέχει μία άλλη μεταβλητή που ορίζεται σε config αρχείο και περιέχει το όνομα του ενεργού template. Αυτό το όνομα αντιστοιχεί σε φάκελο στο σύστημα αρχείων και σε συγκεκριμένο url.

5.2.4 – library/admincmscontroller.php

Αυτή είναι η κλάση που κληρονομείται από όλα τα admin κομμάτια των modules του CMS. Όπως και πριν ορίζεται η μεταβλητή του TEMPLATE.

Μετά φορτώνεται το adminmodel (σύνδεση με τη βάση δεδομένων) και αυτό χρησιμοποιείται από τον constructor για διάφορες εργασίες.

Αρχικά φορτώνεται το βασικό μενού, μαζί με υπό-μενού σύμφωνα με config αρχείο που υπάρχει στο φάκελο κάθε module. Έτσι εμφανίζονται τα links αυτόματα σε μενού στο τελικό html αρχείο που παράγεται. Μετά καλείται η συνάρτηση loadOptions().

Μέσα από αυτή τη συνάρτηση, οι μεταβλητές που ορίζονται στα config αρχεία μπορούν να γίνουν overwrite από το CMS.

Η συνάρτηση loadOptions() ξαναγράφει τις config μεταβλητές με τις νέες τιμές τους που έχουν στη βάση δεδομένων.

Σε κάθε module επιτρέπεται να αποθηκεύει και να διαβάζει τις δικές του μεταβλητές, οι οποίες αποθηκεύονται στη βάση δεδομένων.

5.2.5 – library/auth.php

Αυτή η κλάση είναι η κλάση που χρησιμοποιείται από το CMS για οτιδήποτε έχει να κάνει με τη διαχείριση χρηστών. Μέσα στον

Constructor παίρνουμε το username του χρήστη και τη λίστα με τα groups που ανήκει .

Με αυτή την κλάση μπορούμε επίσης να αλλάξουμε τα groups του χρήστη, να κάνουμε logout, να ελέγξουμε αν ο χρήστης είναι logged in.

Αυτές οι εργασίες γίνονται με τη βοήθεια βοηθητικού Model που φορτώνεται στην αρχή του constructor.

5.2.6 – library/cache.class.php

Αυτή η κλάση επιτρέπει στον προγραμματιστή να αποθηκεύσει μια σελίδα ή κάποια τιμή ή κάποιο κείμενο και να το ανακτήσει όταν το ξαναχρειαστεί. Η ίδια κλάση μπορεί να χρησιμοποιηθεί σε πιο ψηλό επίπεδο για αποθήκευση του τελικού output της σελίδας ούτως ώστε όταν ξαναζητηθεί η ίδια σελίδα να φορτωθεί από το cache, και έτσι να αποφύγουμε την εκτέλεση αχρείαστου κώδικα.

Εκτός από το σύστημα αρχείων, οι cached σελίδες μπορούν να αποθηκευτούν και στην RAM και αυτό δίνει πολύ καλύτερη απόδοση σε όλο το σύστημα.

Προσφέρονται 3 συναρτήσεις:

- function store($key,$data,$ttl = ''). Αυτή η συνάρτηση δέχεται την τιμή που θέλουμε να φυλάξουμε στην προσωρινή μνήμη, το κλειδί (που μπορεί να είναι το url ή οποιαδήποτε άλλη τιμή θέλουμε). Επίσης δέχεται μια παράμετρο ttl που είναι η τιμή που αντιπροσωπεύει το χρόνο σε δευτερόλεπτα που θα είναι έγκυρη η τιμή που είναι cached. Αν δε δίνεται η τιμή $ttl τότε χρησιμοποιείται μια default τιμή που ορίζεται σε ένα αρχείο config. Ο φάκελος του cache ορίζεται και αυτός σε config αρχείο.
- function fetch($key), συνάρτηση που του δίνεις ένα key και επιστρέφει την τιμή από την cache μνήμη. Αν η τιμή είναι παλιά (σύμφωνα με το $ttl) τότε το αντίστοιχο αρχείο διαγράφεται.
- function delete($key), διαγράφει το αρχείο που αντιστοιχεί στο κλειδί που δίνεται σαν παράμετρος.

- private function getFileName($key), η συνάρτηση αυτή δίνει το όνομα του αρχείου που αντιστοιχεί στο κλειδί που περνιέται σαν παράμετρος.

5.2.7 – library/email.class.php

Αυτή η κλάση είναι μια βοηθητική κλάση που περιέχει συναρτήσεις που επιτρέπουν στον προγραμματιστή εύκολη αποστολή email. Συναρτήσεις:

- function send($from, $to, $subject ,$message): αυτή η συνάρτηση επιτρέπει την αποστολή email στο οποίο μπορείς να δώσεις την διεύθυνση αποστολής και το subject του email, λειτουργικότητα η οποία δεν είναι διαθέσιμη από την PHP

5.2.8 – library/file.class.php

Αυτή η κλάση περιέχει μια συνάρτηση η οποία διαβάζει και επιστρέφει τα περιεχόμενα ενός αρχείου.

5.2.9 – library/html.class.php

Αυτή η κλάση περιέχει διάφορες βοηθητικές συναρτήσεις που βοηθούν σε λειτουργίες που έχουν να κάνουν με το τελικό output της εφαρμογής μας. Πιο συγκεκριμένα υπάρχουν:

- Function shortenUrls($data): Αυτή η συνάρτηση παίρνει ένα url και φτιάχνει ένα μικρότερο url με την βοήθειά του tinyurl API.
- Function sanitize($data): Καθαρίζει ένα κείμενο από χαρακτήρες που μπορούν να βλάψουν το τελικό output της σελίδας, ή τη βάση δεδομένων.
- Function Link() – Δημιουργεί html κώδικα για τα links ανάλογα με τις παραμέτρους.
- function includeJs($fileName) – φτιάχνει htm κώδικα που φορτώνει το javascript που το όνομα του περνιέται σαν παράμετρος.
- Function includeCss() – Αντίστοιχο με το πιο πάνω αλλά για τα CSS.

5.2.10 – library/html.class.php

Αυτή η κλάση επιτρέπει στον προγραμματιστή να πάρει input από το χρήστη. (Κυρίως POST και COOKIES). Είναι πιο ασφαλές από το $_POST

και τον $_COOKIES πίνακα αντίστοιχα γιατί περιέχει συναρτήσεις που καθαρίζουν το input του χρήστη και το κάνουν κατάλληλο για εισαγωγή στη βάση δεδομένων ή ακόμα και για html output.

Πιο συγκεκριμένα υπάρχει συνάρτηση που καθαρίζει ειδικούς χαρακτήρες που δημιουργούν προβλήματα στην εισαγωγή τους στη βάση δεδομένων, με τη βοήθεια της mysql_real_escale_string() και υπάρχει άλλη συνάρτηση που καθαρίζει το input από οπουδήποτε javascript κώδικα κάνοντας τον έτσι ασφαλή από xss επιθέσεις και έτοιμο να τυπωθεί στο τελικό html κείμενο.

Στον constructor της κλάσης καταστρέφεται ο $_GET πίνακας, καθώς δε χρησιμοποιείται στο πρόγραμμα σε κανένα σημείο.

Η συνάρτηση post($name = false), αν κληθεί χωρίς παράμετρο, επιστρέφει true αν υπάρχουν POST δεδομένα, αλλιώς επιστρέφει false. Αν κληθεί με παράμετρο κάποιο κείμενο, τότε επιστρέφεται το περιεχόμενο του πίνακα $_POST[$name] και αν δεν υπάρχει επιστρέφεται FALSE.

Η συνάρτηση postarray() επιστρέφει όλο τον πίνακα $_POST.

Η συνάρτηση postarray_xssclean() επιστρέφει όλο το ίδιο με τη συνάρτηση postarray, αλλά καθαρό από οποιοδήποτε javascript κώδικα.

Η συνάρτηση postarray_databasexssclean() επιστρέφει το ίδιο με πάνω, αλλά τα περιεχόμενα του πίνακα είναι έτοιμα για αλληλεπίδραση με τη βάση δεδομένων.

Η post_xxsclean($name) επιστρέφει το ίδιο με την post() (ένα στοιχείο του πίνακα $_POST) αλλά καθαρό από javascipt κώδικα.

Η post_databaseclean($name) επιστρέφει το ίδιο με την post αλλά αυτό που επιστρέφει φιλτράρεται και είναι έτοιμο για εισαγωγή στη βάση δεδομένων.

Η post_databasexxsclean($name) κάνει το ίδιο με το post_databaseclean αλλά είναι έτοιμο για τη βάση δεδομένων και είναι και καθαρό από javascript κώδικα.

5.2.11 – library/lang.class.php

Αυτή η κλάση αναλαμβάνει την υποστήριξη αρχείων μεταφράσεων που χρειάζονται για την εφαρμογή μας.

Πιο συγκεκριμένα μπορεί να φορτώσει ένα αρχείο .lang.php που βρίσκεται είτε στον φάκελο application/languages/ είτε σε ένα από τους φακέλους των διαφόρων modules στο φάκελο languages.

Μετά μπορεί να δώσει οποιαδήποτε γραμμή αυτού του αρχείου.

Οι συναρτήσεις αναλυτικά:

Function load($name): φορτώνει ένα αρχείο γλώσσας, ανάλογα με τη μεταβλητή $name και με την global μεταβλητή $config['ft_defaultlang']. Αυτή η μεταβλητή τίθεται από το σύστημα αυτόματα στον κώδικα εκκίνησης και διαβάζεται από το url. Αν δεν έχει τεθεί αυτή η μεταβλητή, τότε σαν default γλώσσα χρησιμοποιείται η γλώσσα $config['ft_defaultlang'] που ορίζεται στο config αρχείο.

Η συνάρτηση ψάχνει πρώτα στο φάκελο application/languages και μετά στους υποφακέλους του φάκελου modules. Από πλευράς απόδοσης είναι προτιμότερο τα αρχεία γλώσσας να είναι στο φάκελο application/languages/

function line($line = ''): αυτή η συνάρτηση επιστρέφει μια γραμμή από το config αρχείο, σύμφωνα με τη μεταβλητή $line.

Τα αρχεία γλώσσας ορίζονται ως εξής:

$lang[‘modulename_variablename’] = “This is a translated text”.

5.2.12 – library/loader.class.php

Αυτή η κλάση είναι από τις πιο σημαντικές κλάσεις του συστήματος. Είναι η κλάση που χρησιμοποιεί ο controller για να φορτώσει άλλες κλάσεις, models, views, libraries και scripts.

Για τους models και τα views ψάχνει πρώτα στο φάκελο applications και μετά στους υποφακέλους του φάκελου applications/modules.

Το ίδιο ισχύει και για τα views.

Εκτός από την load συνάρτηση υπάρχουν και οι συναρτήσεις set get και vars, που είναι υπεύθυνες για να ετοιμάζουν τις μεταβλητές που θα τυπωθούν στα views. Η συνάρτηση vars δέχεται ένα πίνακα από μεταβλητές. Ετοιμάζεται ο πίνακας $data που με τη βοήθεια της εντολής extract ετοιμάζονται για χρήση στα views.

Η συνάρτηση config φορτώνει ένα αρχείο από το φάκελο config, και τον ενσωματώνει στο global πίνακα $config για χρήση όταν χρειάζεται. Με αυτό τον τρόπο δε φορτώνουμε κάθε φορά αχρείαστα config files αλλά τα φορτώνουμε μόνο όταν χρειάζονται.

Η συνάρτηση function database() κάνει τη σύνδεση με τη βάση δεδομένων σύμφωνα με τα στοιχεία που βρίσκονται στο αρχείο config και την κάνει διαθέσιμη σε ολόκληρη την εφαρμογή.

Αν η συνάρτηση function κληθεί με την πρώτη παράμετρο true ($this->load->database(true)) τότε ένα νέο στιγμιότυπο της κλάσης βάσης δεδομένων δημιουργείται και επιστρέφεται μετά την κλήση της συνάρτησης, ενώ η global σύνδεση στη βάση δεδομένων μένει η ίδια.

Αν έχει ξανακληθεί η συνάρτηση database, και η δεύτερη κλήση δεν έχει την παράμετρο true, τότε δε δημιουργείται νέο instance της κλάσης αλλά επιστρέφεται το παλιό.

Η συνάρτηση library (που καλείτε ως $this->load->library(“security”) για παράδειγμα) φορτώνει μία βοηθητική κλάση από το φάκελο library. Αυτή η συνάρτηση ψάχνει αρχεία που τελειώνει το όνομά τους σε .class.php σε αντίθεση με τις άλλες συναρτήσεις της κλάσης load.

5.2.13 – library/model.class.php

Αυτή η κλάση είναι η βασική κλάση model. Δεν έχει κάποια λειτουργικότητα εκτός από μία συνάρτηση που εκτυπώνει το τελευταίο query που έγινε στη βάση δεδομένων.

Δημιουργήθηκε για σκοπούς οργάνωσης και εναρμόνισης με την MVC αρχιτεκτονική. Μέσα στους Models γίνεται οποιαδήποτε εργασία έχει να κάνει με την βάση δεδομένων. Τυπικά έχουμε ένα Model για κάθε πίνακα της βάσης δεδομένων αλλά αυτό δεν είναι υποχρεωτικό.

Συναρτήσεις: getReturn() αυτή η συνάρτηση με την βοήθεια της mysql_error() ελέγχει αν το query εκτελέστηκε κανονικά, και αν δεν εκτελέστηκε κανονικά βάζει στη μεταβλητή error της κλάσης το error που υπήρχε. Είναι χρήσιμη κυρίως για debugging.

printQuery(): Τυπώνει το τελευταίο επερώτημα που έγινε στη βάση δεδομένων.

5.2.14 – library/security.class.php

Αυτή η κλάση έχει μερικές συναρτήσεις που αφορούν την ασφάλεια του συστήματος.

Αναλυτικά οι κλάσεις:

- function isAlphaNumeric($str): ελέγχει αν το $str περιέχει μόνο γράμματα και αριθμούς.
- function isAlphaNumericSpace($str): το ίδιο με πριν αλλά επιτρέπεται και το space.
- function getRandomNumberMD5(): επιστρέφει στο χρήστη ένα το md5 ενός random string. Για τον υπολογισμό του random string λαμβάνεται υπόψη ο χρόνος ώστε να είναι πιο «τυχαίο» αλλά και μία μεταβλητή στο config αρχείο για περισσότερη ασφάλεια. Χρησιμοποιείται για την αποθήκευση των κωδικών στη βάση δεδομένων, και προσφέρει περισσότερη ασφάλεια γιατί αν κάποιος κάπως έχει πρόσβαση στη βάση δεδομένων δεν μπορεί εύκολα να αποκωδικοποιήσει τα md5 με προϋπολογισμένους πίνακες, γιατί το τυχαίο κείμενο αλλάζει σε κάθε εφαρμογή.
- Function getRandomNumberSHA1(): το ίδιο με το πιο πάνω με τη διαφορά ότι τώρα έχουμε sha-1 και όχι md5 (διαφορετικός hashing αλγόριθμος, πιο ασφαλής)
- Function databaseclean($str): καθαρίζει το κείμενο από ειδικούς χαρακτήρες που μπορεί να δημιουργήσουν πρόβλημα αν εισάγουμε ένα κείμενο στη βάση δεδομένων, και έτσι αποτρέπονται επιθέσεις τύπου sql injections.

- function isNumeric($text): ελέγχει αν ένα string περιέχει μόνο αριθμούς.
- Function function xss_clean($text): αυτή η συνάρτηση βρίσκει και αφαιρεί οποιοδήποτε javascript από το κείμενο μετατρέποντάς το σε έτοιμη μορφή για εκτύπωση στο τελικό html κείμενο.
- function getRandomString($length = 50): αυτή η συνάρτηση δίνει ένα τυχαίο text στον χρήστη μεγέθους $length με default τιμή 50. Χρησιμοποιεί την random μεταβλητή που ορίζεται στο config αρχείο. Για περισσότερη ασφάλεια αυτή η τιμή πρέπει να αλλάζει σε κάθε εγκατάσταση.

Υπάρχουν και διάφορες άλλες δευτερεύουσες βοηθητικές συναρτήσεις που βοηθούν κυρίως στο xss clean.

5.2.15 – library/database/*.php

Ο φάκελος αυτός περιέχει την Active Record υλοποίηση της παρούσας εργασίας. Ο προγραμματιστής μπορεί προαιρετικά να χρησιμοποιήσει τη βιβλιοθήκη, ή μπορεί να χρησιμοποιήσει τις κλασικές εντολές της php για σύνδεση στη βάση δεδομένων. Τα πλεονεκτήματα του ActiveRecord είναι πολλά. Αν και είναι κάπως δύσκολο στην αρχή για τον προγραμματιστή να το μάθει, είναι ένα πολύ δυνατό εργαλείο στα χέρια του προγραμματιστή, που προσφέρει νέες δυνατότητες.

- Η ταχύτητα ανάπτυξης της εφαρμογής ελαχιστοποιείται στο ελάχιστο.
- Μπορούμε να έχουμε αντικειμενοστραφή αλληλεπίδραση με τη βάση δεδομένων.
- Μέσα από της βιβλιοθήκες του Active Record γίνεται σε πολλά σημεία αυτόματο φιλτράρισμα των δεδομένων, πράγμα που οι κλασικές εντολές της PHP δεν κάνουν.
- Το Active Record είναι ανεξάρτητο της βάσης δεδομένων που βρίσκεται από πίσω. Το Active Record είναι ένα επίπεδο παραπάνω μεταξύ του προγραμματιστή και της βάσης δεδομένων, οπότε μπορεί να φτιαχτεί άλλος “driver” για αλληλεπίδραση με άλλες

βάσεις δεδομένων, και ο προγραμματιστής με τις ίδιες εντολές μπορεί να έχει πρόσβαση σε διαφορετικά συστήματα βάσεων δεδομένων. Για τους σκοπούς αυτής της εργασίας γράφτηκαν “drivers” για MySQL και SQLite.

Το κλασικό μοντέλο Active Record αναπτύχθηκε για χρήση από την Ruby On Rails.

Ορισμένα αρνητικά:

- Υπάρχει επιπλέον εκτέλεση κώδικα για προετοιμασία των αποτελεσμάτων σε μορφή αντικειμενοστραφή για τον προγραμματιστή. Για το λόγο αυτό χρειάζεται περισσότερη μνήμη και χρόνο εκτέλεσης. Σύμφωνα με μετρήσεις που έγιναν για τους σκοπούς αυτής της εργασίας αν το query επιστρέφει λίγα αποτελέσματα ο χρόνος είναι αμελητέος, αλλά σε περίπτωση που έχουμε μεγάλα ResultSets τότε αυτή η καθυστέρηση είναι σημαντική. Οπότε πρέπει να υπάρχει προσοχή από τον προγραμματιστή ώστε να βάζει LIMIT στα queries που είναι πιθανό να επιστρέψουν πολλά αποτελέσματα.

Εδώ χρησιμοποιούμε μια τροποποιημένη έκδοση του Active Record που χρησιμοποιείται από το Codeigniter. Περισσότερες λεπτομέρειες χρήστης μπορεί κάποιος να βρει εδώ:

http://codeigniter.com/user_guide/database/active_record.html

Στην ανάπτυξη του CMS χρησιμοποιείται τόσο το Active Record όσο και οι κλασικές εντολές αλληλεπίδρασης με τη MySQL.

5.3 – Αρχεία Κώδικα CMS

Το CMS υλοποιείται ένα σύνολο από modules που χρησιμοποιούν τη λειτουργικότητα του Framework και επεκτείνουν τις ανάλογες κλάσεις AdminCmsController και PublicCmsController. Για τους σκοπούς της εργασίας αυτής φτιάχτηκαν μόνο τα admin κομμάτια των Modules. Αυτά τα modules θα είναι πάντα εγκατεστημένα στο σύστημα και θα λειτουργούν σαν βάση περαιτέρω ανάπτυξης της εφαρμογής.

Σε αυτό το σημείο θα παρουσιαστεί και ο κώδικας σε επιλεγμένα modules ώστε να παρουσιαστεί ο τρόπος λειτουργίας του Framework και τον τρόπο με τον οποίο χρησιμοποιείται από τον προγραμματιστή.

Το default module είναι το “Home”. Δεν έχει ουσιαστικά καμιά λειτουργικότητα, και η δουλειά του είναι απλά να εμφανίζει το μενού ώστε ο χρήστης να μπορεί να επιλέξει τι θέλει να κάνει.

Το ποιο είναι το default module μπορεί να το αλλάξει ο χρήστης στο αρχείο config του συστήματος.

5.3.1 – Το module home

5.3.1.1 – Το αρχείο homeadmincontroller.php

Αυτό το αρχείο πρέπει να βρίσκεται στο φάκελο application/modules/home/admin/controllers ώστε να μπορεί το σύστημα να το βρει. Παρακάτω παρουσιάζεται ο κώδικας.

```
<?php

class HomeAdminController extends AdminCmsController

{

    var $modulename = 'home';

    var $controllername = 'home';

    function beforeAction ()

    {

    }

    function index()

    {

        $data = array();

        $data['main'] = 'adminindexview';

        $data['submenu'] = array();
```

```
            $this->load->vars($data);
            $this->load->view('admintemplateview');
        }
        function afterAction()
        {
        }
}΄
```

Αυτό είναι το πιο απλό παράδειγμα module. Έχει μόνο ένα action (registered function που μπορεί να εκτελεστεί από το χρήστη), το index. Οι συναρτήσεις beforeAction και afteraction καλούνται πάντα πριν και μετά την εκτέλεση του κώδικα του action αυτόματα από το σύστημα. Αυτό είναι χρήσιμο όταν να κάνουμε κάποιες εργασίες πριν και μετά την εκτέλεση, σε περιπτώσεις caching και σε πολλές άλλες περιπτώσεις.

Στο function index() φαίνεται η βασική λειτουργικότητα του προγράμματος. Δημιουργούμε ένα νέο array με το όνομα data και θέτουμε διάφορες μεταβλητές. Αυτός ο πίνακας με την εντολή $this->load->vars($data) θα ετοιμαστεί και θα είναι διαθέσιμο σε οποιαδήποτε views φορτώσουμε μετά.

Αμέσως μετά με την εντολή $this->load->view('AdminTemplateView') φορτώνουμε το admintemplateview όπου είναι το βασικό view του συστήματος.

$data['main'] = 'adminindexview'; αυτή η εντολή λέει στο εξωτερικο view ότι για βασικό περιεχόμενο της σελίδας θέλουμε ένα άλλο view, το adminindexview το οποίο βασικά είναι ένα html αρχείο με μερικές πληροφορίες για το σύστημα.

Στην συνέχεια παρουσιάζεται το περιεχόμενο του admintemplateview.php. Αυτό το αρχείο βρίσκεται στο φάκελο application/modules/home/admin/views/default

Το “default” είναι εκεί γιατί τα views μπορεί να χωρίζονται σε διάφορα templates. Το default template ορίζεται και αυτό στο config αρχείο. Μπορούμε δηλαδή με μια αλλαγή στο config αρχείο να αλλάξουμε ολόκληρη την εμφάνιση της εφαρμογής μας.

Ο κώδικας του αρχείου:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"

"http://www.w3.org/TR/html4/loose.dtd">

<html>

<head>

<title>Control Panel</title>

<meta http-equiv="Content-Type" content="text/html; charset=utf-8">

<link href="<?= TEMPLATE_URL ?>css/styles.css" rel="stylesheet" type="text/css">

</head>

<body>

<div id="main-width">

<!-- top start -->

<? $this->load->view("admintopmenuview"); ?>

<!-- top end -->


 <div id="main">

  <? $this->load->view($main); ?>

<div id="side">

   <div class="box-top">Info</div>
```

```
<div class="box">This is the beta version of Hostzone CMS built on Hostzone Framework</div>

    <div class="box-top">Support</div>

<div class="box">Support: <a href="#" target="_blank">nikosglikis@gmail.com</a> or Call: <strong>(+357) 99 804938</strong></div>

</div>

  </div>

 <!-- footer start -->

 <div id="footer">Hostzone Control Panel v.1.0</div>

<!-- footer end -->

</div>

</body>

</html>
```

Με κόκκινα φαίνονται τα σημαντικά σημεία του κώδικα. Στο πρώτο σημείο βλέπουμε χρήση της σταθεράς TEMPLATE_URL. Αυτή η σταθερά δημιουργείται αυτόματα από το σύστημα και λαμβάνει υπόψη ποιο είναι το ενεργό template και αν είμαστε σε admin mode ή public mode.

Με την εντολή $this->load->view($main) φορτώνουμε ένα άλλο view μέσα στο βασικό view. Αυτή η δυνατότητα αυξάνει τη λειτουργικότητα του συστήματος.

5.3.2 – To module budget management

Θέλουμε ένα module που θα διαχειρίζεται ένα budget. Σε κάθε έτος έχουμε ένα συγκεκριμένο διαθέσιμο ποσό. Πρέπει να μοιραστεί στα διάφορα τμήματα. Έστω ότι υπάρχουν επικεφαλής τμημάτων που διεκδικούν ορισμένα ποσά. Αυτοί κάνουν request κάποιο ποσό από το budget μιας χρονολογίας. Για να δεσμευτεί αυτό το ποσό πρέπει να συμφωνήσουν όλοι οι άλλοι επικεφαλής τμημάτων. Αν έστω και ένα τμήμα διαφωνήσει το request θεωρείται denied.

5.3.3 – Το module budget management - υλοποίηση

Το budget management module είναι module ανεξάρτητο και είναι ένα παράδειγμα πλήρους εφαρμογής που βασίζεται πάνω στο CMS. Έχει ένα controller μερικούς Models και μερικά views. Επειδή όμως είναι και module του CMS έχει και ένα επιπλέον αρχείο που περιγράφει το module τα actions περιέχει πληροφορίες για τα menu κ.α. και χρησιμοποιείται από το CMS για να «εγκατασταθεί» το module και να μπορεί να χρησιμοποιηθεί από το σύστημα.

5.3.3.1 – Το αρχείο install.php

Ακολουθεί ο κώδικας του αρχείου και αμέσως μετά η ανάλυσή του:

```
<?

    $module = array(

    'name'=>'budget',

    'public'=>'0',

    'title'=> 'Budget Manager',

    'icon'=> 'budget.png',

    'sort' =>'3',

    'url'=>'budget'

    );

    $module_actions = array();


    $module_actions[] = array('name'=>'index', 'type' => 'admin', 'menu'=> '0', 'icon'=>'', 'title'=>'', 'sort'=>'0');

    $module_actions[] = array('name'=>'addyear', 'type' => 'admin', 'menu'=> '1', 'icon'=>'add.png', 'title'=>'Add Year', 'sort'=>'1');
```

```
$module_actions[] = array('name'=>'listbudgets', 'type' => 'admin', 'menu'=> '1', 'icon'=>'edit.png', 'title'=>'View Edit Years', 'sort'=>'1');

$module_actions[] = array('name'=>'edit', 'type' => 'admin', 'menu'=> '0', 'icon'=>'', 'title'=>'', 'sort'=>'1');

$module_actions[] = array('name'=>'delete', 'type' => 'admin', 'menu'=> '0', 'icon'=>'', 'title'=>'', 'sort'=>'2');

$module_actions[] = array('name'=>'options', 'type' => 'admin', 'menu'=> '1', 'icon'=>'page_white_wrench.png', 'title'=>'Options', 'sort'=>'3');

$module_actions[] = array('name'=>'request', 'type' => 'admin', 'menu'=> '1', 'icon'=>'budget.png', 'title'=>'New Request', 'sort'=>'4');

$module_actions[] = array('name'=>'listrequests', 'type' => 'admin', 'menu'=> '1', 'icon'=>'budget.png', 'title'=>'Requests', 'sort'=>'5');

$module_actions[] = array('name'=>'vote', 'type' => 'admin', 'menu'=> '', 'icon'=>'', 'title'=>'', 'sort'=>'-1');

//$module_schema = 'schema.sql'; //false if database is not needed

function moduleBeforeInstall()

{

}

function moduleAfterInstall()

{

        //mysql_query("INSERT INTO cms_modules_permissions VALUES ((SELECT id FROM cms_modules_actions WHERE name = 'login' AND type = 'public'),(SELECT id FROM groups WHERE name = 'Visitor'))");

        //echo mysql_error();

}

?>
```

Στην αρχή ένα πίνακα με το όνομα $module που περιέχει τη δομή το module. Πιο συγκεκριμένα περιέχει όνομα, τίτλο και εικονίδιο για το μενού και το url που πρέπει να δοθεί στο σύστημα για να δράσει το module.

Μετά δημιουργείται ο πίνακας $module_actions που περιέχει μέσα όλα τα actions και περιέχει τα χαρακτηριστικά τους. Κάθε γραμμή περιέχει τα χαρακτηριστικά ενός action. Πιο συγκεκριμένα:

- Κάθε action έχει ένα όνομα.
- Ένα τύπο (admin ή public)
- Κάθε action μπορεί να είναι στο μενού ή όχι. Αν στο πεδίο μενού είναι 1 τότε ένα link στο action εμφανίζεται αυτόματα στο μενού.
- Αν το action υπάρχει στο μενού τότε υπάρχουν και τα πεδία “title”, “icon”, “sort”. Title είναι το κείμενο που εμφανίζεται στο μενού, icon η εικόνα που φορτώνεται αυτόματα από το url του template, και το πεδίο sort που αντιπροσωπεύει με ποια σειρά θα εμφανίζονται τα links στο μενού.

Αυτό το αρχείο διαβάζεται από το CMS όταν ο χρήστης προσπαθήσει να εγκαταστήσει το module και όλες οι πληροφορίες εισάγονται αυτόματα στη βάση δεδομένων.

Μετά ο administrator πρέπει να ρυθμίσει την πρόσβαση του κάθε group στο κάθε action. Αυτό γίνεται με εύκολο τρόπο μέσα από το CMS. Περισσότερα στο Παράρτημα 1.

Σε αυτό το σημείο αξίζει να αναφερθεί ότι ο installer περιέχει και δυνατότητα εισαγωγής .sql αρχείου στη βάση δεδομένων, ώστε το νέο module να έχει διαθέσιμα κάποια default δεδομένα.

Επίσης μπορεί να εκτελέσει και συναρτήσεις που εκτελούνται πριν και μετά από το την εγκατάσταση του module.

5.3.3.2 – Το αρχείο admin/controllers/budgetadmincontroller.php

Αυτή η κλάση είναι η κλάση με τη λειτουργικότητα του module. Παρακάτω δίνεται ο κώδικας και αμέσως μετά η ανάλυσή του.

```
<?php

class BudgetAdminController extends AdminCmsController
{
	var $modulename = 'budget';
	var $controllername = 'budget';
	var $data = array();
	function beforeAction ()
	{
		$this->load->model('BudgetAdminYearModel');
		$submenu = $this->adminmodel->getModuleMenu($this->modulename,'admin');
		$this->data['submenu'] = $submenu;
	}
	function vote($id)
	{
		$this->data['main'] = 'budgetadminvoteview';
		$this->data['id'] = $id;
		if ($this->BudgetAdminYearModel->getUserVoteCount($id))
		{
			$this->setFlash('You have already voted for this request! ','error');
			$this->data['main'] = 'budgetadminflashview';
			return;
```

```
            }
            $voteGroup = $this->getKey('voteGroup');
            if (!in_array($voteGroup,$this->auth->groups_ids))
            {
                $this->setFlash('You cannot vote, you are not in the vote Group! ','error');
                $this->data['main'] = 'budgetadminflashview';
                return;
            }
            if (is_numeric($id) && $this->input->post())
            {
                if ($this->BudgetAdminYearModel->insertVote($this->input->post_databaseclean('vote'),$id))
                $this->setFlash('Thank you. Your vote is successfully saved!','success');
                else
                $this->setFlash('There was some error, please try again. Error: '.$this->BudgetAdminYearModel->error,'error');
            }
        }
        function index()
        {
            $this->data['main'] = 'budgetindexview';
        }
        function setFlash($message, $class)
```

```
{
        $this->data['flash_class'] = $class;
        $this->data['flash_message'] = $message;
        return;
}
function addyear()
{
        if ($this->input->post('year'))
        {
                $this->load->library('security');
                $year = $this->input->post('year');
                $this->data['main'] = 'budgetadminaddyearformview';
                if ($this->security->isAlphaNumeric($year))
                {
                        if (strlen($year) != 4)
                        {
                                $this->setFlash('Year format is not corrent.
Year length is > 4.','error');
                                return;
                        }
                        if ($year < 2000)
                        {
                                $this->setFlash('Year is not valid. Must be
> 2000','error');
```

```
                return;
            }
            $money = $this->input->post('money');
            if (!$this->security->isAlphaNumeric($money) || $money <0)
            {
                $this->setFlash('Money has no valid input.','error');
                return;
            }
            $this->load->database();
            $result = $this->BudgetAdminYearModel->insertYear($year,$money);
            if ($result)
            {
                $this->setFlash($result,'error');
                return;
            }
            $this->setFlash('Year Added Succesfully','success');
        }
        else
        {
            $this->setFlash('Year format is not corrent. Year is a 4 digit number.','error');
```

```
                return;
            }

        }
        $this->data['main'] = 'budgetadminaddyearformview';
    }
    function listbudgets()
    {
        $this->data['main'] = 'budgetadminlistview';

        $this->data['budgets'] = $this->BudgetAdminYearModel->getBudgets();
    }
    function delete($id)
    {

        $this->data['main'] = 'budgetadmindeleteview';
        if (!$this->BudgetAdminYearModel->isValidYearId($id))
        {
            $this->setFlash('Not valid year id. Someone is doing something wrong!','error');
            return;
        }
        if ($this->BudgetAdminYearModel->deleteYear($id))
```

```
            {
                $this->setFlash('Year Deleted Successfully','success');
            }
            else
            {
                $this->setFlash('There was some error, please try again','error');
            }

        }
        function options()
        {
            $this->load->model('GroupsAdminModel');
            $this->data['groupid'] = $this->getKey('voteGroup');

            $this->data['groups'] = $this->GroupsAdminModel->getGroups();
            $this->data['main'] = 'budgetadminoptionsformview';
            $groupid = $this->input->post('groupid');
            if ($groupid)
            {
                if (is_numeric($groupid) && $this->GroupsAdminModel->isValidGroupId($groupid))
                {
                    $this->setKey('voteGroup',$groupid);
```

```
                    $this->setFlash('Group Changed
Successfully','success');
                    return;
                }
                else
                {
                    $this->setFlash('Group id invalid, please try again
'.$groupid,'error');
                    return;
                }
            }
        }
        function request()
        {
            $this->data['years'] = $this->BudgetAdminYearModel-
>getYears();
            $this->load->library('security');
            $this->data['main'] = 'budgetadminrequestview';
            if ($this->input->post())
            {

                $title = $this->input->post_databasexxsclean('title');
                $details = $this->input-
>post_databasexxsclean('details');
```

```
$amount = $this->input->post_databasexxsclean('amount');

$year = $this->input->post_databasexxsclean('yearid');

$department = $this->input->post_databasexxsclean('department');

if (!is_numeric($amount))

{

$this->setFlash('Amount input is not valid: '.$amount,'error');

return;

}

if ( is_numeric($year) && $this->BudgetAdminYearModel->insertRequest($title,$amount,$details,$department,$year) )

{

$this->setFlash('Request inserted to database','success');

return;

}

else

{

$this->setFlash('There was some error inserting request to database, please try again. Error was: '.$this->BudgetAdminYearModel->error,'error');

return;

}
```

```
            }
        }
        function listrequests()
        {
            $this->data['main'] = 'budgetadminlistrequests';
            $this->data['requests'] = $this->BudgetAdminYearModel-
>getRequests();
        }
        function afterAction()
        {

            $this->load->vars($this->data);
            $this->load->view('admintemplateview');
        }
        function edit($id)
        {
            $this->load->library('security');
            if (!$this->BudgetAdminYearModel->isValidYearId($id))
            {
                die('Not valid year id. Someone is doing something
wrong!');
            }
            $this->data['id']=$id;
            $this->data['main'] = 'budgetadmineditformview';
```

```
            if ($this->input->post('money') && $this->security->isNumeric($this->input->post('money')))
            {
                $money = $this->input->post('money');
                if ($this->BudgetAdminYearModel->updateYear($id,$money))
                {
                    $this->setFlash('Year Updated Successfull','success');
                    return;
                }
                else
                {
                    $this->setFlash('There was some problem, please try again','error');
                    return;
                }
            }
        }
}
```

Το πρώτο πράγμα που παρατηρούμε είναι ότι η κλάση επεκτείνει την κλάση AdminCmsController. Αυτό αυτόματα λέει στο σύστημα ότι πρόκειται για κλάση του CMS και το σύστημα αυτόματα προσθέτει λειτουργικότητα στην κλάση, όπως σύνδεση στη βάση δεδομένων, αυτόματη δημιουργία του μενού κ.α.

Επίσης βλέπουμε ότι στη συνάρτηση beforeaction γίνονται διάφορες εργασίες που είναι ίδιες για όλο το υπόλοιπο του κώδικα, για τα περισσότερα actions. Έτσι γλιτώνουμε χρόνο στην ανάπτυξη της εφαρμογής.

Πιο συγκεκριμένα φορτώνονται κάποια models που στη συνέχεια χρησιμοποιούνται για να διαβάσουν το μενού από τη βάση δεδομένων.

Μετά έχουμε τη συνάρτηση/action index που δεν κάνει τίποτε ιδιαίτερο απλά φορτώνει ένα viewer.

Το action **addyear** προσθέτει μια χρονολογία στο σύστημα. Το input δίνεται με POST φόρμα. Αρχικά φορτώνεται η βιβλιοθήκη security. Μετά ελέγχεται αν όντως το input που δόθηκε για χρονολογία είναι όντως αριθμός. Αν δεν είναι έγκυρο το input τότε το πρόγραμμα τερματίζει με μήνυμα λάθους. Μετά ελέγχεται και το πεδίο money αν είναι σωστό.

Στη συνέχεια φορτώνεται η κλάση της βάσης δεδομένων και με τη βοήθεια του BudgetAdminYearModel εισάγεται το ποσό στη βάση δεδομένων με μία μόνο γραμμή κώδικα.

Η συνάρτηση **request** καλείται όταν κάποιος από τους επικεφαλής των τμημάτων θέλει να δεσμεύσει ένα ποσό. Και πάλι γίνονται οι κατάλληλοι έλεγχοι και εισάγεται το request στη βάση δεδομένων.

Τότε όλοι οι επικεφαλής πρέπει να ψηφίσουν ώστε το request να είναι δεκτό. Η ψήφος του κάθε επικεφαλής γίνεται με τη συνάρτηση **vote**. Πρώτα γίνεται έλεγχος αν ο χρήστης δεν έχει ψηφίσει σωστά για το ίδιο request, μετά ελέγχει αν είναι στο group των voters, αλλιώς δεν μπορεί να ψηφίσει.

Τέλος αν όλοι οι έλεγχοι γίνουν με επιτυχία τότε η ψήφος τους εισάγεται στη σύστημα.

Αξίζει να σημειωθεί πόσο βολικό είναι η αρχιτεκτονική MVC. Στον controller δεν υπάρχει ούτε γραμμή κώδικα html, και κανένα SELECT ή οποιαδήποτε εντολή αλληλεπίδρασης με τη βάση δεδομένων. Αυτό κάνει τη συντήρηση και τροποποίηση του κώδικα πολύ πιο εύκολο από τις κλασικές μεθόδους ανάπτυξης εφαρμογών.

Όλη η αλληλεπίδραση γίνεται μέσα από τους models. Αν για παράδειγμα θέλουμε να αλλάξουμε τη βάση δεδομένων από mysql σε sqlite τότε ο κώδικας του controller μένει απείρακτος και αλλάζει μόνο ο Model.

5.3.3.3 – Το αρχείο admin/models/budgetadminyearmodel.php

```
<?

class BudgetAdminYearModel extends Model

{

	function insertVote($vote,$rid)

	{

		$this->db->insert('cms_budget_votes',array('vote'=>$vote,'rid'=>$rid,'uid'=>$this->auth->getUserId()));

		$this->updateRequestStatus($rid);

		return $this->getReturn();

	}

	function updateRequestStatus($rid)

	{

		$res = mysql_query("SELECT count(*) FROM cms_budget_votes WHERE vote = 'no' AND rid='$rid'");

		//echo mysql_error();

		$res = mysql_fetch_array($res);

		$count = intval($res[0]);

		if ($count)

		{

			//echo "declined";
```

```
                mysql_query("UPDATE cms_budget_requests SET
status = 'declined' WHERE id = '$rid'" );

                return;

        }

        $res = mysql_query("SELECT count(*) FROM
cms_budget_votes WHERE vote = 'yes' AND rid='$rid'");

        echo mysql_error();

        $res = mysql_fetch_array($res);

        $votesCount = intval($res[0]);

        //echo "Positive Votes Count: $votesCount";


        $FR =& get_instance();

        $voteGroup = $FR->getKey('voteGroup');

        echo $voteGroup;

        $res = mysql_query("SELECT count(*) FROM users_groups
WHERE gid = '$voteGroup'");

        $res = mysql_fetch_array($res);

        $groupCount = intval($res[0]);

        //echo "Vote Group Count: $groupCount";

        if ($votesCount == $groupCount)

        {

                mysql_query("UPDATE cms_budget_requests SET
status = 'accepted' WHERE id = '$rid'" );

        }
```

```
	}
	function getUserVote($rid)
	{
		$this->db->where(array('rid'=>$rid,'uid'=>$this->auth->getUserId()));
		$this->db->select('vote');
		$query = $this->db->get('cms_budget_votes');
		echo $this->db->count_all_results();
		if ($this->db->count_all_results()==0)
		{
			return false;
		}
		$vote = $this->db->getOne($query);
		return $vote;
	}
	function getUserVoteCount($rid)
	{
		$this->db->where(array('rid'=>$rid,'uid'=>$this->auth->getUserId()));
		$this->db->select('count(*)');
		$query = $this->db->get('cms_budget_votes');

		$vote = $this->db->getOne($query);
```

```
		$vote = intval($vote);
		return $vote;
		if ($vote=='0')
		{
			return false;
		}
		else
		{
			return true;
		}
}
function insertYear($year,$money)
{
		$this->db->insert('cms_budget_years',array (
'id' => NULL,
		'year' => $year,
		'money'=>$money
)                                                    );
		if (mysql_error()!='')
		{
			if (strpos(mysql_error(),'Duplicate')!== FALSE)
			{
				return "Year already in database";
			}
```

```
                return mysql_error();
        }
        else
        {
                return false;
        }
}
function getYearById($id)
{
        $this->db->where(array('id'=>$id));
        $this->db->select('year');
        $query = $this->db->get('cms_budget_years');
        return $this->db->getOne($query);
}
function getRequests()
{
        $this->db->orderby('id desc');
        $query = $this->db->get('cms_budget_requests');
        return $query->result_array();
}
function getBudgets()
{
        $this->db->order_by('year desc');
        $query = $this->db->get('cms_budget_years');
```

```
        $years = array();
        foreach($query->result() as $year)
        {
                $years[] = array(
                                        'id' =>$year->id,
                                        'year' => $year->year,
                                        'money' => $year->money
                                        );
        }
        return $years;
}

function getYears()
{
        $query = $this->db->get('cms_budget_years');
        $years = array();
        foreach($query->result() as $result)
        {
                $year['id'] = $result->id;
                $year['year'] = $result->year;
                $year['money'] = $result->money;
                $years[] = $year;
        }
        return $years;
```

```
	}
	function updateYear($id,$money)
	{
		$this->db->where(array('id'=>$id));
		$this->db->update('cms_budget_years',array('money'=>$money));

		return true;
	}

	function deleteYear($id)
	{
		$this->db->delete('cms_budget_years',array('id'=>$id));
		return $this->getReturn();
	}
	function insertRequest($title,$amount,$details,$department,$year)
	{
		$data = array('id'=>NULL,
'title'=>$title,
'amount'=>$amount,

	'department'=>$department,
'details'=>$details,
'year'=>$year
```

```
                                        );
            $this->db->insert('cms_budget_requests',$data);
            return $this->getReturn();
        }
        function isValidYearId($id)
        {
            $this->load->library('security');
            if (!$this->security->isNumeric($id))
            {
                return false;
            }
            $id = $this->security->databaseclean($id);
            $this->db->where(array('id'=>$id));
            $query = $this->db->get('cms_budget_years');
            if ($query->count_all_results()>0)
            {
                return true;
            }
            else
            {
                return false;
            }
        }
}
```

Όπως βλέπουμε είναι απλή σύνδεση με βάση δεδομένων και αλληλεπίδραση με αυτή, είτε με τη χρήση της λειτουργικότητας Active Record είτε με τις κλασικές php εντολές.

Μερικά παραδείγματα χρήστης του ActiveRecord:

```
$data = array('id'=>NULL, 'title'=>$title, 'amount'=>$amount, 'department'=>$department, 'details'=>$details, 'year'=>$year);

$this->db->insert('cms_budget_requests',$data);
```

Εισαγωγή στη βάση δεδομένων με ουσιαστικά 2 γραμμές.

Παράδειγμα select:

```
$this->db->where(array('id'=>$id));

$query = $this->db->get('cms_budget_years');
```

Τα αποτελέσματα είναι διαθέσιμα σε πολλές μορφές, είτε κλασικά σε associative array.

5.3.4 – Το module “modulesmanager”

Το module αυτό είναι από τα βασικά modules του συστήματος και αναλαμβάνει την εγκατάσταση άλλων modules. Διαβάζει το αρχείο install.php μέσα σε κάθε module και περνά τις πληροφορίες στη βάση δεδομένων, εκτελεί οποιαδήποτε .slq αρχεία που χρειάζονται και τον κώδικα στις συναρτήσεις beforeInstall και afterInstall αν υπάρχουν.

Περνά όλα registered actions (functions) στο σύστημα και ρυθμίζει την πρόσβαση του κάθε group πάνω στο κάθε action με πολύ εύκολο και γρήγορο τρόπο (περισσότερα στο παράρτημα Α’).

Μια βασική διαφορά των controllers του MVC (PublicCmsController, AdminCmsController) από τους απλούς Controllers του framework είναι ότι οι δεύτεροι δεν τρέχουν αν τα actions δεν είναι registered στο σύστημα και αν ο χρήστης δεν ανήκει σε Group που να έχει πρόσβαση στο συγκεκριμένο action).

5.3.5– To module “users”

Αυτό το module δουλεύει μαζί με την auth βιβλιοθήκη του Framework και επιτρέπει δημιουργία, διαγραφή και τροποποίηση χρηστών στο σύστημα. Επίσης επιτρέπει δημιουργία groups και επιτρέπει την εισαγωγή των χρηστών σε ένα ή περισσότερα groups.

Κεφάλαιο 6

Συμπεράσματα και ανακεφαλαίωση

Συμπερασματικά μπορούμε να αναφέρουμε ότι επιτεύχθηκε ο στόχος για την ανάπτυξη ενός ελαφρού framework για ανάπτυξη εφαρμογών για τον παγκόσμιο ιστό.

Είναι φτιαγμένο ακολουθώντας πιστά τις καλύτερες αρχιτεκτονικές που υπάρχουν και για αυτό το λόγο είναι εξαιρετικά ευέλικτο επιτρέπει γρήγορη ανάπτυξη εφαρμογών.

Έχει όλη την απαραίτητη λειτουργικότητα και αρχιτεκτονική που επιτρέπει τη γρήγορη ανάπτυξη εφαρμογών και αναπτύχθηκε και ένα βασικό MVC για να αναπτύξει ο προγραμματιστής πάνω του την εφαρμογή του.

Από πλευράς απόδοσης είναι πολύ κοντά στο CodeIgniter στο καλύτερο του είδους και είναι πολύ πιο γρήγορο από όλα τα άλλα υπάρχοντα Frameworks, έστω και αν αποτελεί μια αρχική έκδοση που μπορεί να δεχθεί πολλές βελτιστοποιήσεις και αλλαγές.

Βιβλιογραφία

[1] Write your own PHP MVC Framework (Part 1)
http://anantgarg.com/2009/03/13/write-your-own-php-mvc-framework-part-1/

[2] How to create a simple MVC Framework in PHP
http://www.henriquebarroso.com/how-to-create-a-simple-mvc-framework-in-php/

[3] A Role-Based Access Control (RBAC) system for PHP
http://www.tonymarston.net/php-mysql/role-based-access-control.html

[5] Model-View-Controller
http://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller

[6] David Lane, Hugh E. Williams - O'Reilly - Web Database Applications with PHP and MySQL 2nd Edition

[7] SQL Injections http://en.wikipedia.org/wiki/SQL_injection

[8] David Sklar, Learning PHP 5

Παράρτημα 1

Εγκατάσταση και χρήση ενός module στο Framework

Εγκατάσταση του module “Budget”

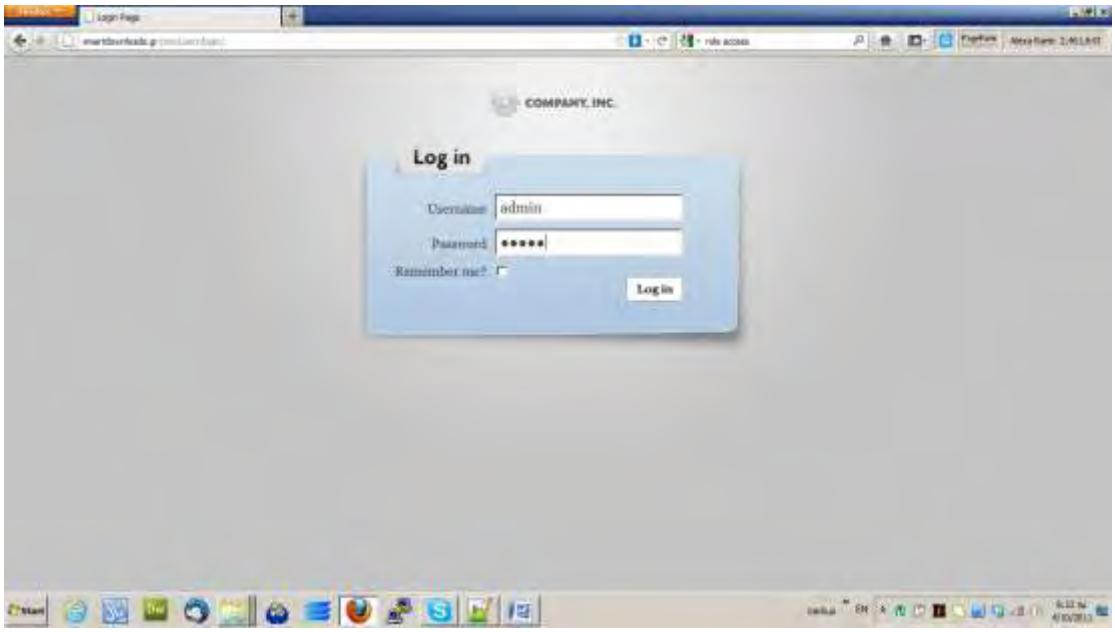
Εισαγωγή

Για τη λειτουργία του budget module επιλέγετε από τον administrator ένα group από users (ή δημιουργείτε ένα καινούργιο), οι οποίοι πρέπει να ψηφίσουν όλοι θετικά ώστε να θεωρείται το request ότι έγινε “accepted”.

Μετά πρέπει να δοθεί σε αυτό το group πρόσβαση στις λειτουργίες (actions) του Module. Η διαδικασία εξηγείτε με screenshots παρακάτω.

Url Εφαρμογής: http://smartdownloads.gr/cms/admin/
Username: admin
Password: admin

Το http://smartdownloads.gr/cms/ είναι το βασικό URL της εφαρμογής (προσωρινό url που χρησιμοποιήθηκε για ανάπτυξη της πτυχιακής) και http://smartdownloads.gr/cms/admin/ για να δούμε το admin μέρος του CMS (Content Management System)



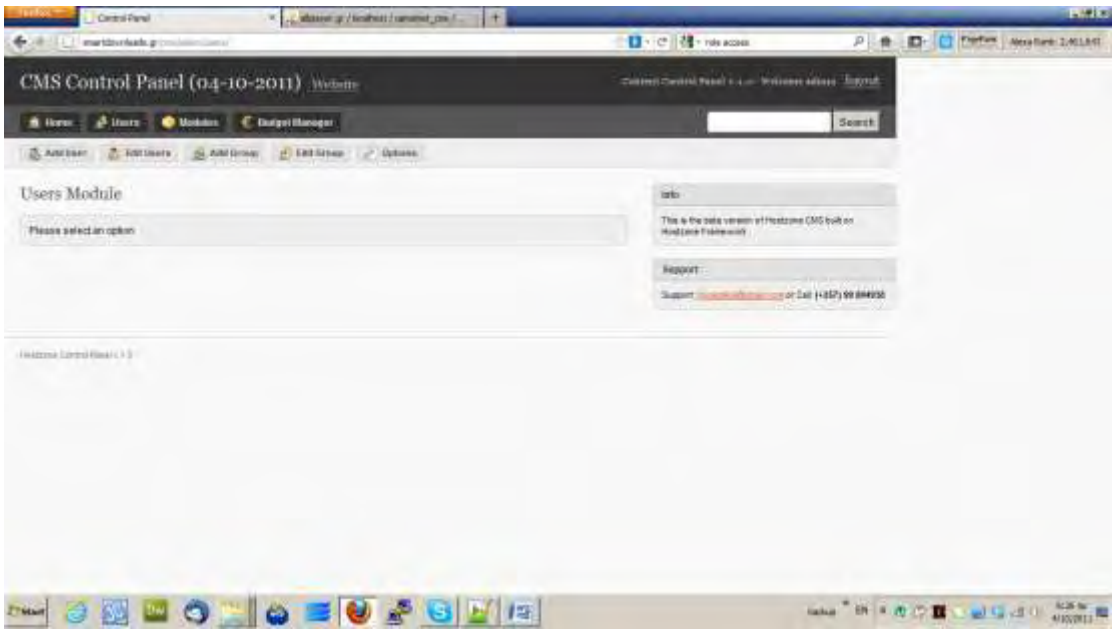
http://smartdownloads.gr/cms/admin/

Username: admin

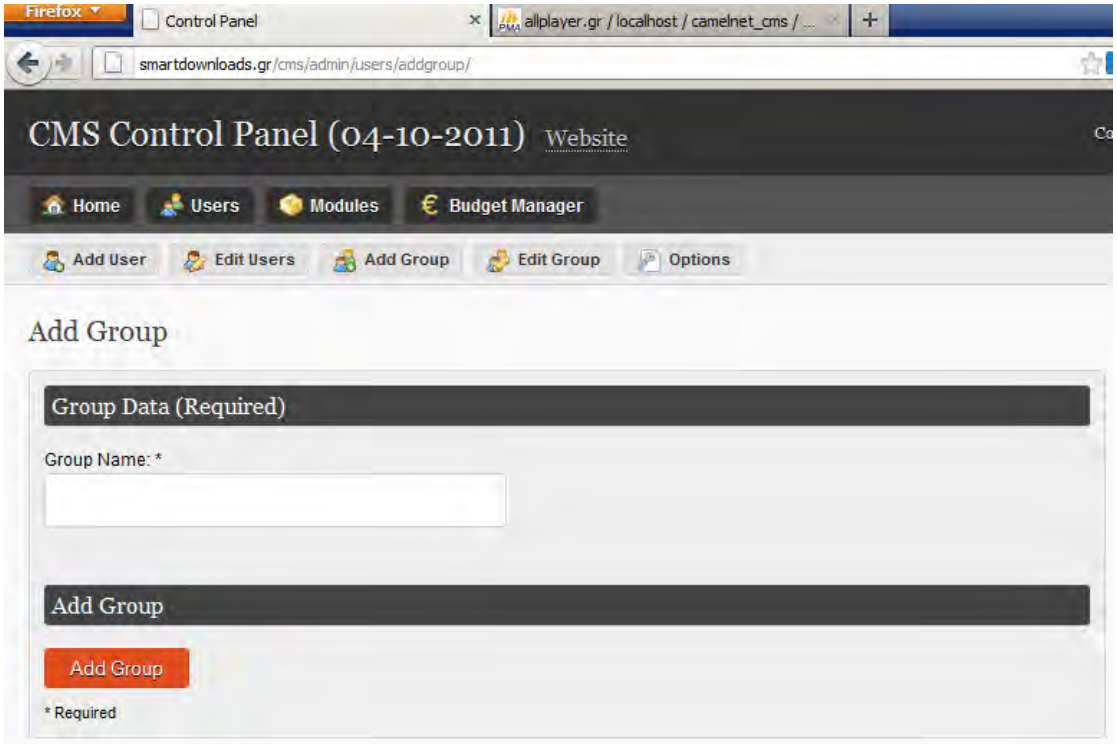
Password: admin

Προετοιμασία User Module για το Budget Module

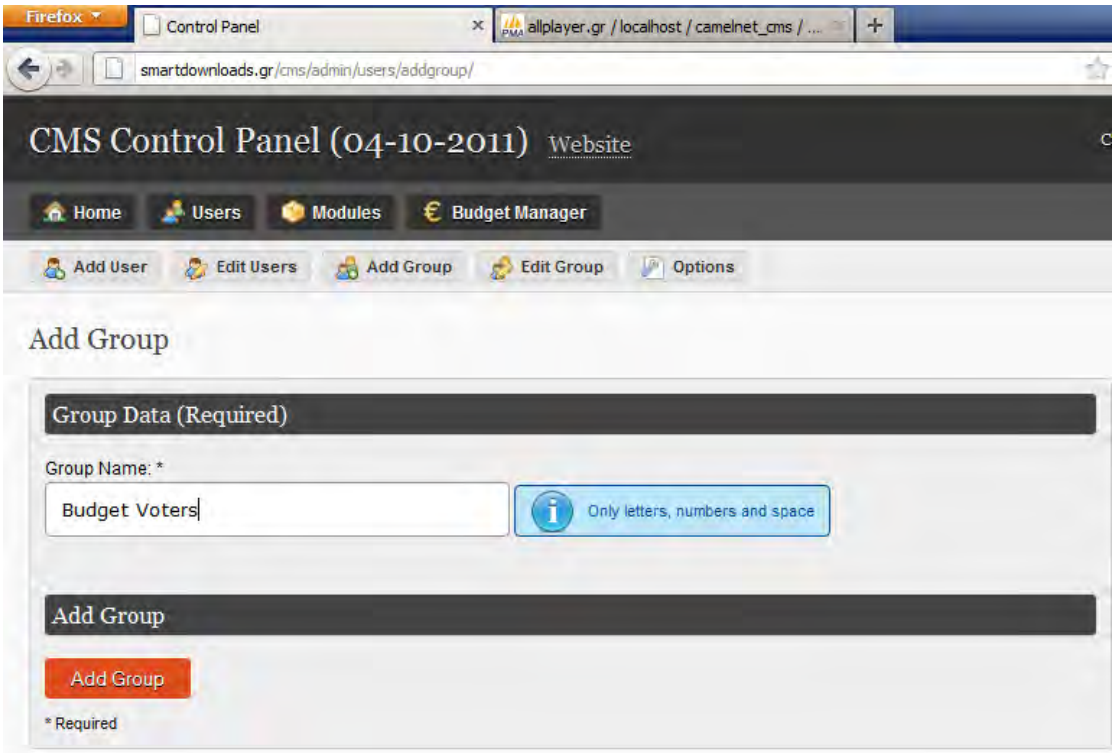
Επιλέγω την επιλογή Users



Add Group



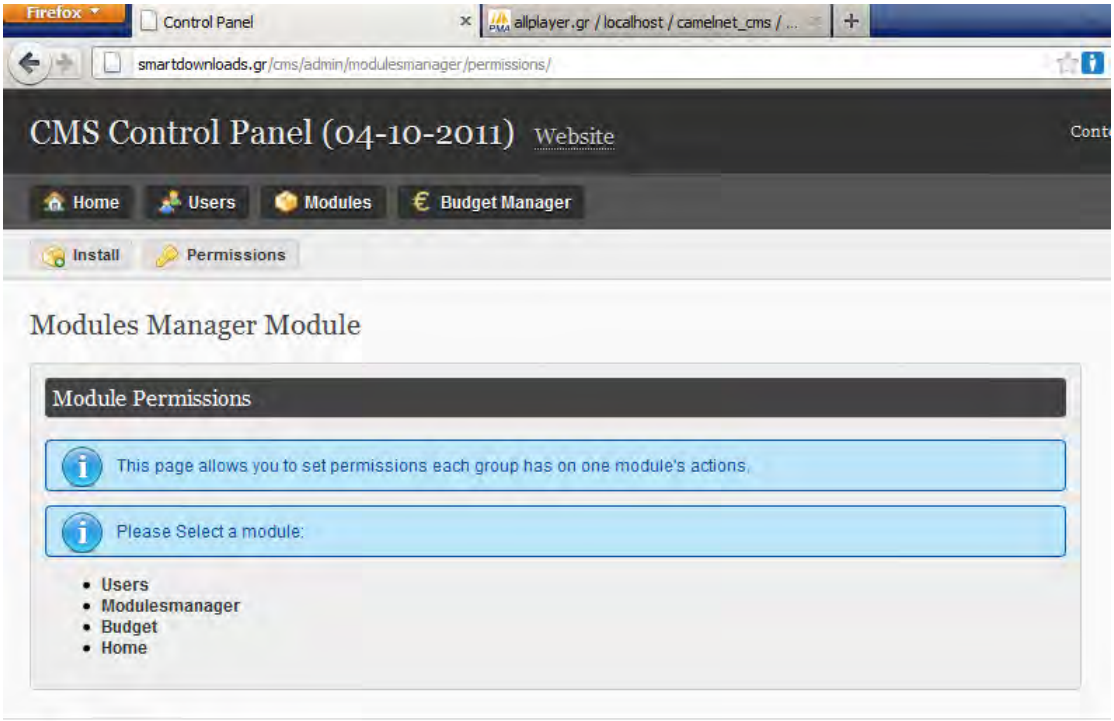
Συμπληρώνω το όνομα του group



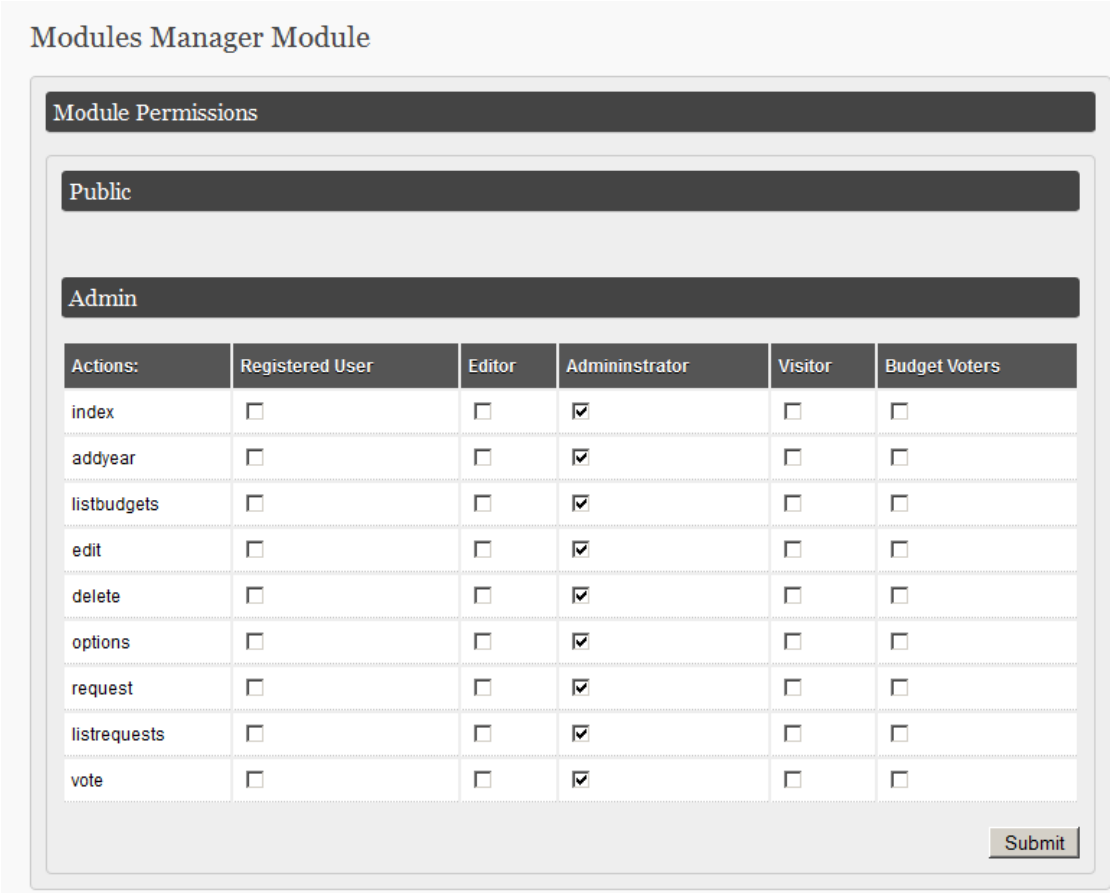
Click στο κουμπί Add Group

Τώρα πρέπει να δώσουμε πρόσβαση στις λειτουργίες του Budget Module στο νέο Group.

Click στο Modules και μετά στο Permissions. Εδώ εμφανίζεται μία λίστα με τα registred modules.

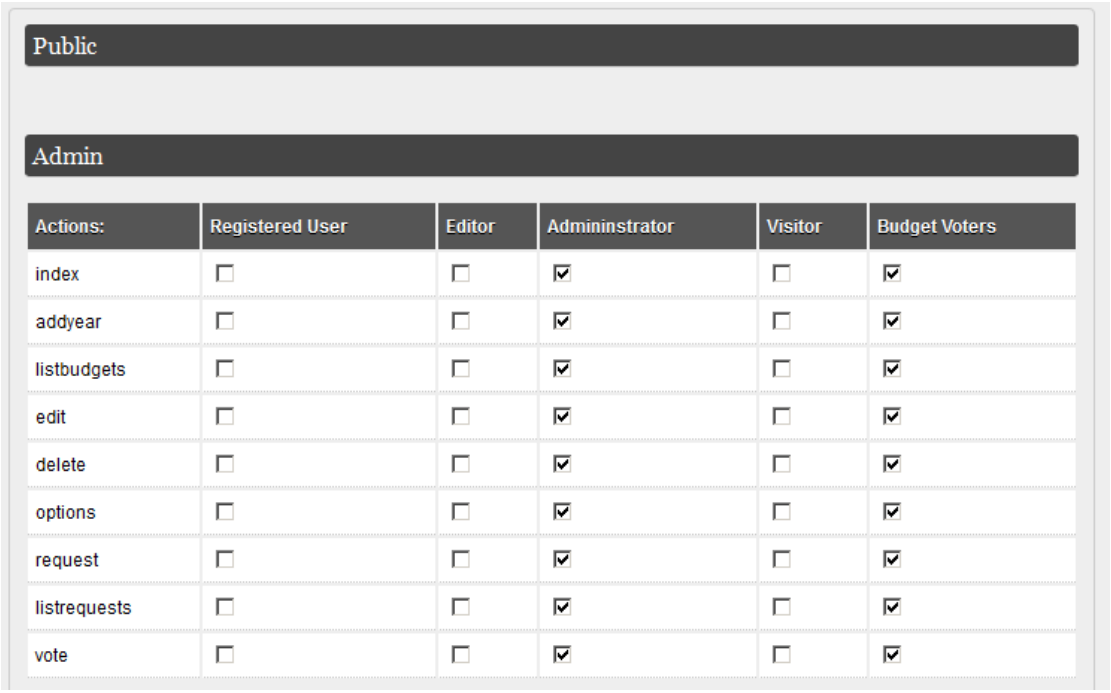


Επιλέγουμε το Budget



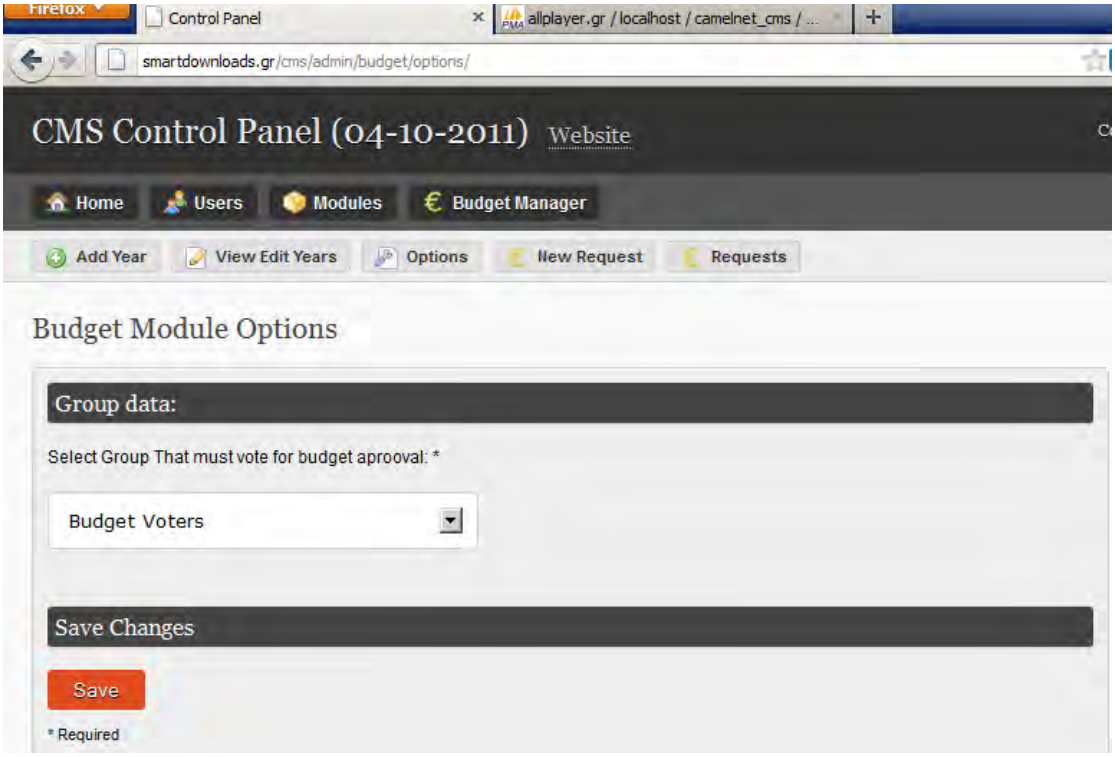
Οριζόντια εμφανίζονται τα διάφορα Groups και κάθετα τα actions (μέθοδοι) του Module. Όπως βλέπουμε το Group “Administrator” έχει

πρόσβαση σε όλα τα actions Του Module. Πρέπει να κάνουμε το ίδιο για το Group “Budget Voters”.



Με τον ίδιο τρόπο μπορούμε να δημιουργήσουμε ένα Group “Παρατηρητών” που θα έχουν πρόσβαση μόνο σε ορισμένες λειτουργίες, πχ εμφάνιση των Requests.

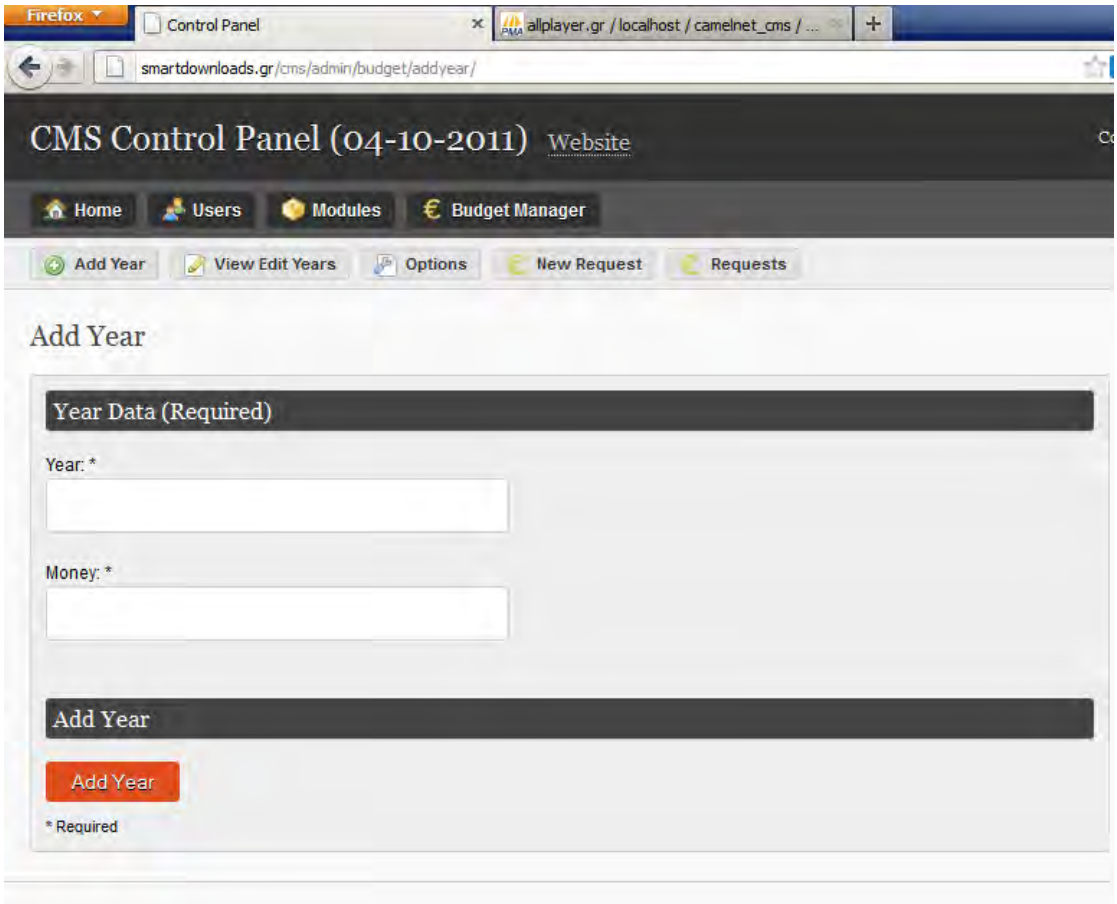
Options Budget Group – Επιλέγουμε Budget Manager – Options.



Εδώ επιλέγω πιο από τα groups πρέπει να ψηφίσει θετικά ώστε το Request να θεωρείται αποδεκτό.

Χρήση Budget Module

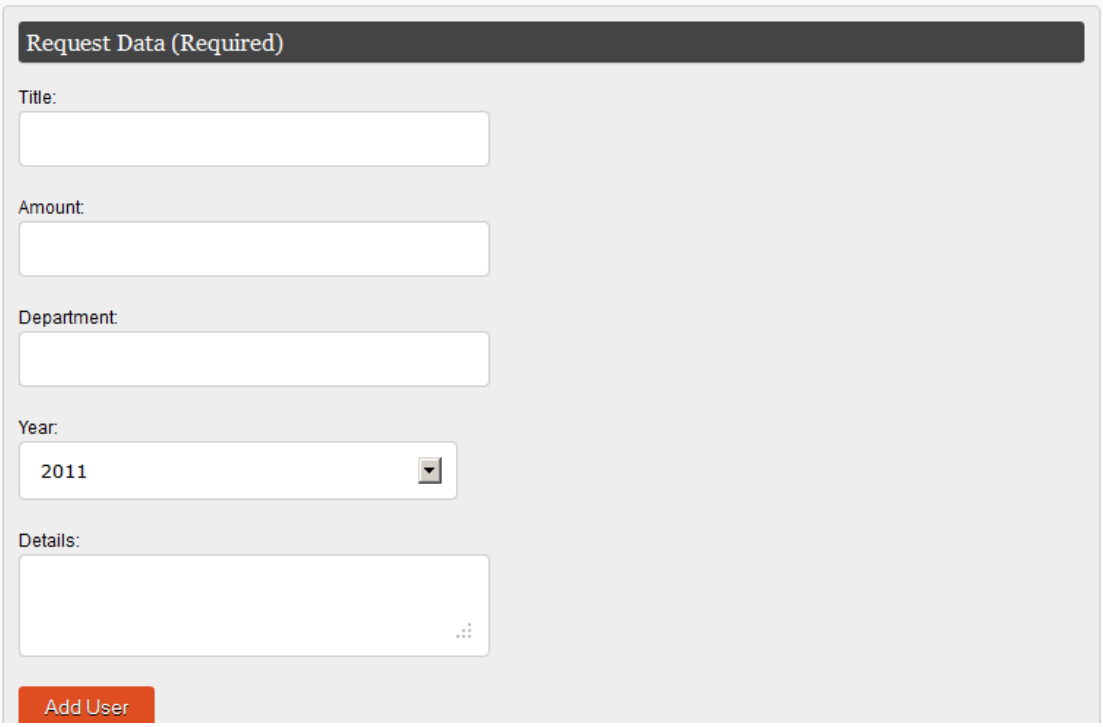
Επιλογή Budget Module – Add Year



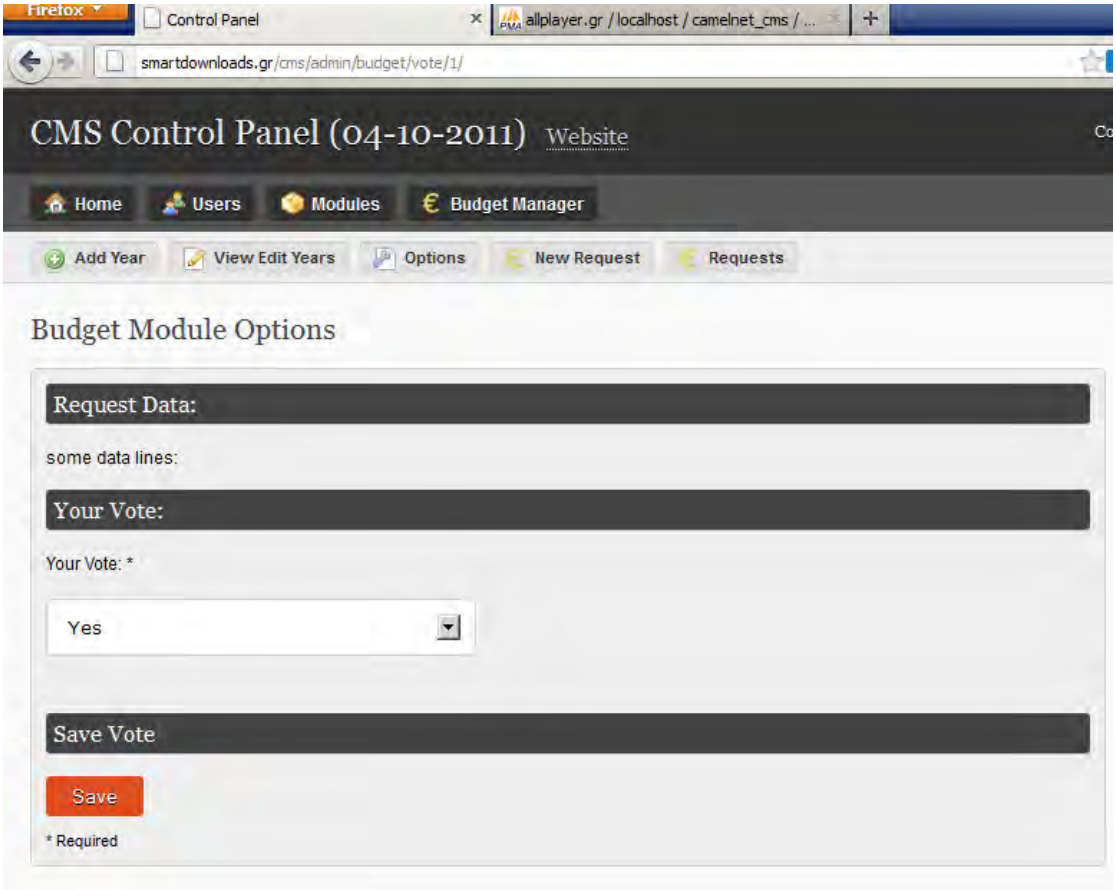
Συμπληρώνω τη χρονολογία και το διαθέσιμο ποσό και κάνω click το Add Year.

Κάνω click στο "View Edit Years" για να επιβεβαιώσω την εισαγωγή του ποσού.

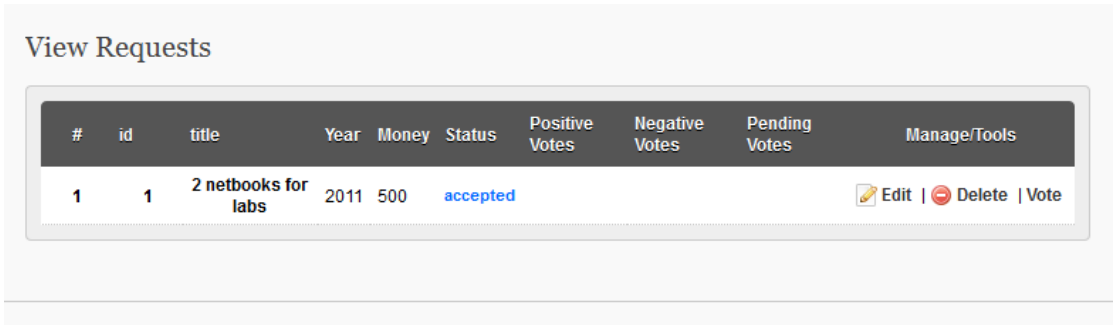
Επιλέγω New Request και συμπληρώνω τα στοιχεία ενός νέου request.



Vote για το request μου – Requests – Vote στο request που θέλω.



Μόλις όλοι οι χρήστες του Group ψηφίσουν θετικά το request μαρκάρεται σαν Accepted.


View Requests
id title Year Money Status Positive Votes Negative Votes Pending Votes Manage/Tools
1 1 2 netbooks for labs 2011 500 accepted Edit | Delete | Vote