Τεχνολογικό Εκπαιδευτικό Ίδρυμα Κρήτης

Σχολή Τεχνολογικών Εφαρμογών
Τμήμα Μηχανικών Πληροφορικής

Πτυχιακή Εργασία

Ανάπτυξη Ηλεκτρονικού Συστήματος Κρατήσεων σε Σύγχρονα Συστήματα Διαχείρισης Περιεχομένου (CMS)

Παπαδάκης Στυλιανός (ΑΜ:1972)

Επιβλέπων καθηγητής : Δρ Ιωάννης Μπαρμπουνάκης

Επιτροπή αξιολόγησης :

Ημερομηνία παρουσίασης:

Ευχαριστίες

Θα ήθελα να ευχαριστήσω ιδιαιτέρως τον κ Ιωάννη Μπαρμπουνάκη για την ανάθεση αυτής της πτυχιακής, καθώς και για την καθοδήγηση και την βοήθεια που μου πρόσφερε. Επίσης θα ήθελα να ευχαριστήσω τον κ Αντώνη Ζερβουδάκη για την βοήθεια και την συνεργασία που είχαμε για την εγκατάσταση της εφαρμογής.

Τέλος θα ήθελα να ευχαριστήσω τους γονείς μου και τους φίλους μου για την βοήθεια και την υποστήριξη που μου έδειξαν καθ' ολη τη διάρκεια της εκπόνησης αυτής της πτυχιακής.

Abstract

At this modern age, the development of websites is becoming more and more automated, as developers try to find ways to automate the development processes. Another thing to keep in mind is the development of dynamic websites so every user, either acquiring programming capabilities or not, can develop and maintain by himself his own website by manipulating the given parameters.

Joomla! is such an increasingly popular tool that is expanding not only at its functionality but also at it's simplicity of use. Its way of use allows anyone to quickly become accustomed to its functions and develop a website without special expertise.

A Joomla! component is an extension that can be installed in a Joomla! Website, extending its functionality.

The ClassRes component, which has been developed for the needs of this thesis provides a means of creating and modifying requests for classroom reservations. The user can see at a glance every available classroom, as well as information regarding it such as the time it is for reservation, send his request for a reservation and the administrators of the website, through the administrator, can modify the classrooms information as well as the reservation requests

In a framework like Joomla! with a standard architecture and automated processes, it is interesting to examine the security that is compensated in exchange for user friendliness and speed.

The evolution of technology helps to gradually overcome all the problems that such an approach presents, creating a framework that can be used without any complicated procedures, offering the utmost simplicity of use in the shortest possible time implementation within a safe and functional environment.

Σύνοψη

Στη σύγχρονη εποχή η κατασκευή ιστοσελίδων γίνεται όλο και πιο αυτοματοποιημένα, καθώς οι προγραμματιστές-αναλυτές ψάχνουν να βρουν τρόπους να αυτοματοποιήσουν τις διαδικασίες κατασκευής. Ένα άλλο φαινόμενο είναι η ανάπτυξη όσο το δυνατόν δυναμικότερων ιστοσελίδων, ώστε ο κάθε χρήστης, είτε έχει προγραμματιστικές γνώσεις είτε όχι, να μπορεί να αναπτύσει και να διαχειρίζεται μόνος του τη δική του ιστοσελίδα αλλάζοντας τις κατά τόπους παραμέτρους.

Το Joomla! είναι ένα ολοένα και δημοφιλέστερο εργαλείο που επεκτείνεται καθημερινά όχι μόνο σε λειτουργίες αλλά και σε απλότητα χρήσης. Ο τρόπος χρήσης του, επιτρέπει σε οποιονδήποτε να αποκτήσει γρήγορα οικειότητα στις λειτουργίες του και να μπορεί να αναπτύξει κάποια ιστοσελίδα χωρίς να χρειάζονται εξειδικευμένες γνώσεις.

Joomla! Components ονομάζονται τα επιπλέον συστατικά που μπορούν να προστίθενται σε κάποια Joomla! ιστοσελίδα, επεκτείνοντας τις λειτουργίες της.

Το Component ClassRes που αναπτύχθηκε για τις ανάγκες της παρούσας πτυχιακής αφορά τη δημιουργία και επεξεργασία αιτήσεων για κρατήσεις αιθουσών. Ο χρήστης μπορεί να βλέπει τις παρεχόμενες αίθουσες και πληροφορίες γι' αυτές καθώς και το πότε είναι διαθέσιμες, να αποστέλει την αίτησή του και οι διαχειριστές, μέσα από κατάλληλο διαχειριστικό περιβάλλον, να επεξεργάζονται τόσο τις αίθουσες όσο και τις αιτήσεις.

Σε ένα τέτοιο περιβάλλον με προτυποποιημένη αρχιτεκτονική και αυτοματοποιημένες διαδικασίες, είναι αρκετά ενδιαφέρον να εξεταστεί η πλευρά της ασφάλειας που θυσιάζεται χάριν της απλότητας και της ταχύτητας.

Η εξέλιξη της τεχνολογίας βοηθάει να ξεπερνιούνται σταδιακά όλα τα προβλήματα που μια τέτοια προσέγγιση παρουσιάζει, συνθέτοντας ένα πλαίσιο που θα μπορεί να χρησιμοποιηθεί χωρίς περίπλοκες διαδικασίες, προσφέροντας τη μέγιστη δυνατή απλότητα χρήσης στον ταχύτερο δυνατό χρόνο υλοποίησης μέσα σε ένα ασφαλές και λειτουργικό περιβάλλον.

Πίνακας περιεχομένων

Πίνακας Εικόνων

Λίστα Πινάκων

1 Εισαγωγή

1.1 Περίληψη

Ο κόσμος μας ολοένα και πιο πολύ δραστηριοποιείται ψηφιακά. Η ανάγκη για προβολή στο Διαδίκτυο είναι πλέον πρωταρχικής σημασίας για κάθε υπηρεσία, οργάνωση ή επιχείρηση. Κάθε σελίδα στο διαδίκτυο εξυπηρετεί κι έναν διαφορετικό σκοπό.

Όσο σημειώνονται εξελίξεις στον τομέα της τεχνολογίας, η ανάγκη για την αξιοποίησή της τόσο σε προσωπικό όσο και σε καθημερινό επίπεδο δημιουργεί περισσότερες απαιτήσεις από την επιστήμη της Πληροφορικής όσον αφορά την απλούστευσή της χρήσης της και την αυτοματοποίηση των επαναλαμβανόμενων διαδικασιών με τρόπο φιλικό προς τον εκάστοτε προγραμματιστή, και εάν είναι δυνατόν, ακόμα και προς τον απλό χρήστη.

Επειδή το χάσμα γνώσης ανάμεσα στον εξειδικευμένο προγραμματιστή και τον απλό χρήστη συνεχώς μειώνεται με την πάροδο του χρόνου και την εξοικείωση των χρηστών στους Η/Υ, είναι φανερό ότι χρειάζεται να αναλάβουν οι απλοί χρήστες αρμοδιότητες που αφορούν τη διαχείριση των εφαρμογών τους.

1.2 Κίνητρο για την Διεξαγωγή της Παρούσας Πτυχιακής Εργασίας

Στο επίπεδο των ιστοσελίδων έχει σημειωθεί σημαντική πρόοδος τα τελευταία χρόνια. Πέραν της προτυποποίησης των τεχνολογιών και της σύγκλισης των διαφόρων περιηγητών που χρησιμοποιούνται από το ευρύ κοινό προς ένα συγκεκριμένο τρόπο ανάλυσης, έχουν αρχίσει και προτυποποιούνται διάφορα συστήματα που παρέχουν συγκεκριμένες ευκολίες στον απλό χρήστη, όπως δυναμική ανανέωση περιεχομένου ή προσθήκη/αφαίρεση προϊόντων, διαχείριση ηλεκτρονικών παραγγελιών π.χ. από ένα ηλεκτρονικό κατάστημα κτλ.

Όπως είναι γνωστό, αρκετές από τις καθιερωμένες χρήσεις διαφόρων ιστοσελίδων αφορούν επαναλαμβανόμενες προγραμματισμένες εφαρμογές. Π.χ. ένα blog με μαγειρικές συνταγές κι ένα με ποδοσφαιρικά νέα έχουν τις ίδιες ανάγκες και λειτουργίες.

Για το λόγο αυτό έχουν αναπτυχθεί συστήματα-πλατφόρμες όπως Joomla!, Drupal, Wordpress κτλ που κάνουν τη χρήση τέτοιων εφαρμογών εύκολη για τον απλό χρήστη. Επί πλέον, επειδή είναι ανοιχτού κώδικα λογισμικά, υποστηρίζονται από μεγάλες κοινότητες στο διαδύκτιο, ενώ παράλληλα βελτιώνονται συνεχώς.

1.3 Σκοπός και Στόχοι Εργασίας

Για την ανάπτυξη της παρούσας εφαρμογής κρατήσεων χρησιμοποιήθηκε μία τέτοια πλατφόρμα ως βάση, συγκεκριμένα η πλατφόρμα "Joomla!". Η συγκεκριμένη εργασία αφορά την ανάπτυξη ενός Joomla! Component το οποίο θα υλοποιεί διαδικτυακή κράτηση των αιθουσών για το Τ.Ε.Ι. Κρήτης σε πραγματικό χρόνο (online) και θα περιλαμβάνει τόσο διαχείριση των αιθουσών όσο και των κρατήσεων, με παράλληλη ενημέρωση των ενδιαφερομένων.

Το Joomla!, όπως και τα υπόλοιπα επονομαζόμενα CMS που κυκλοφορούν (Drupal, Wordpress κτλ), είναι επεκτάσιμο μέσω της εγκατάστασης σε αυτό επεκτάσεων(extensions), δηλαδή πρόσθετων εφαρμογών που επεκτείνουν τη λειτουργικότητα και τις δυνατότητές του. Η παρούσα εργασία αφορά την ανάπτυξη ακριβώς μιας τέτοιας επέκτασης, που θα επεκτείνει τη βασική λειτουργία του Joomla! που αφορά ουσιαστικά διαχείριση άρθρων και θα ενσωματώσει λειτουργικότητα διαχείρισης κρατήσεων και αιθουσών.

Το component κατασκευάστηκε χρησιμοποιώντας διάφορες τεχνολογίες. Έτσι, στα πλαίσια της δημιουργίας του, χρησιμοποιήθηκε η γλώσσα προγραμματισμού PHP, τεχνολογίες XHTML/CSS , XML και η γλώσσα Javascript για τον έλεγχο των πεδίων. Η εφαρμογή στήθηκε και βασίστηκε στην έκδοση 1.5

της πλατφόρμας Joomla!. Δοκιμάστηκε τόσο σε τοπικό επίπεδο (localhost), χρησιμοποιώντας το πακέτο XAMPP, όσο και διαδικτυακά, σε επίπεδο διακομιστή.

1.4 Περιγραφή εργασίας – Κατανομή

Η παρούσα τεκμηρίωση θα ακολουθήσει την παρακάτω δομή:

Στο **κεφάλαιο 1** γίνεται μια εισαγωγή στην εργασία και τη γενική ιδέα και κατεύθυνση που υπάρχει στην ανάπτυξη ιστοσελίδων. Στη συνέχεια περιγράφεται η εργασία και το extension που αναπτύχθηκε, ενώ γίνεται και μια γενική περιγραφή των τεχνολογιών που χρησιμοποιήθηκαν και του CMS στο οποίο βασίστηκε το component.

Στο **κεφάλαιο 2** παρουσιάζεται η δομή και ο τρόπος λειτουργίας του Joomla! καθώς και ενός Joomla! Component, που ουσιαστικά περιγράφει και τη δομή του component που αναπτύχθηκε για το σκοπό της εργασίας.

Στο **κεφάλαιο 3** εστιάζουμε στο component αυτό καθ' εαυτό, περιγράφεται ο τρόπος που λειτουργεί σε επίπεδο κώδικα, με βοήθεια ενός παραδείγματος για κάθε λειτουργία.

Στο **κεφάλαιο 4** ελέγχουμε τη συμβατότητά του με άλλες εκδόσεις του Joomla! ή και άλλες πλατφόρμες CMS και περιγράφουμε τί πιθανές προσαρμογές χρειάζονται για κάτι τέτοιο, ενώ παράλληλα κάνουμε τον απαραίτητο έλεγχο καλής λειτουργίας εξετάζοντας όλες τις πιθανές περιπτώσεις.

Στο **κεφάλαιο 5** αναλύουμε τα συμπεράσματα που βγήκαν από τη μελέτη του component και τη χρήση του σε ένα Content Management System όπως το Joomla!.

Το **κεφάλαιο 6** αφορά τη βιβλιογραφία που χρησιμοποιήθηκε για την ανάπτυξη και συγγραφή της εργασίας.

2 Περιγραφή του Joomla!

Στο παρόν κεφάλαιο εξετάζουμε τη γενική δομή ενός Joomla! Component. Η δομή αυτή ακολουθεί τους ίδιους κανόνες αρχιτεκτονικής με τους οποίους δομείται το ίδιο το Joomla, που ουσιαστικά προκύπτει από την ένωση πολλαπλών components μεταξύ τους. Παρακάτω θα εστιάσουμε στην επεξήγηση του Joomla, της σημασίας του στην ανάπτυξη διαδικτυακού λογισμικού και στον τρόπο με τον οποίο λειτουργεί, ενώ στη συνέχεια επικεντρωνόμαστε στην περιγραφή ενός Joomla! Component.

2.1 Ανάπτυξη Εφαρμογών σε Joomla! και η σημασία του στην ανάπτυξη Διαδικτυακών Εφαρμογών

Το Joomla! CMS (Content Manangment System) με το να χρησιμοποιεί τη γλώσσα PHP που είναι απλή και ευέλικτη μπορεί και ικανοποιεί τέλεια τους χρήστες του. Αυτός είναι κι ένας βασικός λόγος της ανάπτυξής του. Το Joomla! είναι μια δωρεάν πλατφόρμα ανοικτού κώδικα (open-source) η οποία προσφέρει ένα υψηλής ποιότητας σύστημα διαχείρισης μαζί με άλλα πολύ χρήσιμα χαρακτηριστικά. Το Joomla! CMS (Content Management System) είναι εξαιρετικά χρήσιμο για την ανάπτυξη μιας δυναμικής ιστοσελίδας με εξαιρετικά σχέδια και διεπαφές προς το χρήστη. Το Joomla! έχει ένα άλλο θετικό σημείο που φαίνεται στη διαθεσιμότητα ενός μεγάλου συνόλου επεκτάσεων. Υπάρχουν αμέτρητα plug-ins, components και modules διαθέσιμα τα οποία προσφέρουν τεράστιο όφελος στη δημιουργία προηγμένων και πλούσιων σε χαρακτηριστικά ιστοσελίδων. Το δυνατό του σημείο όμως είναι ο τομέας της διαχείρισης περιεχομένου που είναι και η κορυφαία προτεραιότητα στην ανάπτυξη ιστοσελίδων.

Η δημιουργία μιας επιχειρηματικής ιστοσελίδας απαιτεί μεγάλη προσπάθεια, μεγάλη γνώση και χρήση της τελευταίας λέξης της τεχνολογίας. Είναι λοιπόν αναγκαία προϋπόθεση να επιλεγεί μία κατάλληλη, φιλική προς το χρήστη και πλούσια σε χαρακτηριστικά πλατφόρμα ανάπτυξης. Οι περισσότερες από τις ιστοσελίδες των επιχειρήσεων διαθέτουν μια σειρά από λειτουργίες που είναι φιλικές προς τον πελάτη. Η δυνατότητα παροχής ευκολιών προς το χρήστη δεν μπορεί να αγνοηθεί, διότι αυτή είναι η βάση της ανάπτυξης των επιχειρήσεων. Ο χρήστης θέλει να έχει ένα ευχάριστο και λειτουργικό περιβάλλον ενσωματωμένο σε υπερσύγχρονες τεχνολογίες. Ως εκ τούτου, ένας προγραμματιστής οφείλει να κοιτάζει πάντα προς το μέλλον ώστε να εκπληρώνει όλες αυτές τις ανάγκες κατά το στάδιο της ίδιας της ανάπτυξης. Είναι γεγονός ότι, η ανάπτυξη Joomla! εφαρμογών ικανοποιεί τα παραπάνω με άνεση, όπως ακριβώς και η ανάπτυξη εφαρμογών σε PHP που έχει γίνει πολύ δημοφιλής μεταξύ των web developers.

Το Joomla! μπορεί να χρησιμοποιηθεί για σχεδιασμό συνδυάζοντας διάφορα πρότυπα που είναι διαθέσιμα με ενσωμάτωση γραφικών. Πλεονεκτεί δε στην εύκολη παραμετροποίηση, στο χαμηλό κόστος συντήρησης και ταυτόχρονα εξασφαλίζει μεγάλη ασφάλεια για τις PHP εφαρμογές. Ταυτόχρονα, εξασφαλίζεται μεγάλη ασφάλεια για τις PHP εφαρμογές. Επίσης, δεν υπάρχει ασυμβατότητα σχετικά με τα λειτουργικά συστήματα επειδή τρέχει σε διάφορα (Linux, Windows) χωρίς κανένα πρόβλημα. Ένα άλλο αξιοσημείωτο στοιχείο είναι ότι οι ιστοσελίδες που δημιουργούνται με τη βοήθεια του Joomla! παρέχουν υψηλής ποιότητας διαχείριση κινδύνου. Αυτός είναι ο λόγος που έχει γίνει μία από τις πιο χρησιμοποιημένες πλατφόρμες για ιστοσελίδες επιχειρήσεων όπως π.χ. για ιστοσελίδες ηλεκτρονικού εμπορίου.

Τα παραπάνω ασφαλώς δεν σημαίνουν ότι το Joomla! είναι κατάλληλο μόνο για τη δημιουργία επιχειρηματικών web εφαρμογών. Ένας μεγάλος αριθμός δικτυακών τόπων της κοινότητας έχουν αναπτυχθεί σε αυτή την πλατφόρμα. Διάφορες ιστοσελίδες όπως ιστοσελίδες κοινωνικής δικτύωσης, ηλεκτρονικού εμπορίου και επιχειρήσεων, portals ευρέσεως εργασίας, όλα έχουν κατασκευαστεί σε αυτή την τεχνολογία για να αποκομίσουν το καλύτερο αποτέλεσμα. Η δυνατότητα χρήσης των ενσωματωμένων εργαλείων που προσφέρει η τεχνολογία βοηθάει από μόνη της την ανάπτυξη πιο παραγωγικών ιστοσελίδων. Δυνατότητες για χαρακτηριστικά όπως blog, chat, μηχανές αναζήτησης, διαχείριση εγγράφων, παρακολούθηση έργου, διαχείριση επαφών, όλα συμβάλλουν στην επίτευξη του στόχου δημιουργίας του πιο σύγχρονου εικονικού επιχειρηματικού περιβάλλοντος.

Αν λάβει κανείς υπόψιν τα παραπάνω ελπιδοφόρα χαρακτηριστικά, το συμπέρασμα είναι ότι το Joomla! δεν μπορεί να αγνοηθεί όταν αναφερόμαστε σε ανάπτυξη web εφαρμογών καθώς έχει τη δική του σημασία στην ανάπτυξη ιστοσελίδων και την ανάπτυξη web εφαρμογών, όπως και γενικότερα η ανάπτυξη εφαρμογών σε PHP. [1]

2.2 Joomla!: Ορισμός και Περιγραφή

Το Joomla! [2] είναι ένα δωρεάν ανοιχτού κώδικα Σύστημα Διαχείρισης Περιεχομένου (CMS – Content Management System) για δημοσίευση περιεχομένου στο World Wide Web αλλά και σε intranets, ενώ επίσης διαθέτει ένα Model-View-Controller (MVC) Web framework που μπορεί να χρησιμοποιηθεί ανεξάρτητα.

Είναι γραμμένο στη γλώσσα προγραμματισμού PHP, χρησιμοποιεί τεχνικές αντικειμενοστραφή προγραμματισμού (OOP) από την έκδοση 1.5 και πρότυπα σχεδιασμού λογισμικού, αποθηκεύει δεδομένα σε MySQL ή - από την έκδοση 2.5 - MS SQL βάση δεδομένων και περιλαμβάνει χαρακτηριστικά όπως caching σελίδων, RSS feeds, εκτυπώσιμες εκδόσεις των σελίδων, news flashes, blogs, δημοσκοπήσεις, μηχανές αναζήτησης ενώ υποστηρίζει και πολυγλωσσικότητα.

Πάνω από 6.000 δωρεάν και εμπορικές επεκτάσεις διατίθενται από τον επίσημο κατάλογο επεκτάσεών του ενώ πολλές άλλες είναι διαθέσιμες από άλλες πηγές. Εκτιμάται ότι είναι το δεύτερο συχνότερα χρησιμοποιούμενο CMS στο Διαδίκτυο μετά το WordPress.

Το Joomla! έχει πολλά πλεονεκτήματα σε σχέση με custom CMS ή και άλλα δημοφιλή CMS όπως το Drupal ή το Wordpress. Τα κυριότερα του πλεονεκτήματα είναι[3]:

- **Πολυχρηστικό και Πολυεπίπεδο περιβάλλον** – πολλοί χρήστες μπορούν να αλληλεπιδρούν και να συμβάλλουν στην ανάπτυξη μιας ιστοσελίδας βασισμένης σε Joomla!. Οι χρήστες μπορούν να ανήκουν σε διαφορετικές ομάδες με διαφορετικά δικαιώματα.
- **WYSIWYG (What You See Is What You Get) Editor** - ο διαισθητικός WYSIWYG editor επιτρέπει την εύκολη επεξεργασία online περιεχομένου σε πραγματικό χρόνο.
- **Πρόσθετα components / modules** - το Joomla μπορεί εύκολα να ενισχυθεί με πρόσθετες λειτουργίες, από την ενσωμάτωση ενός φόρουμ μέχρι την εγκατάσταση μιας πλήρους λύσης ηλεκτρονικού εμπορίου.
- **Πρότυπα** - υπάρχουν πάρα πολλά ελεύθερα σχέδια και απεριόριστες επιλογές όσον αφορά την εμφάνιση της ιστοσελίδας.

2.3 Ιστορικό έκδοσης

Το Joomla! 1.0 κυκλοφόρησε στις 22 Σεπτεμβρίου 2005 ως νέο εμπορικό release του Mambo 4.5.2.3 που συνδύαζε διορθώσεις σε bugs και σε επίπεδο ασφαλείας.

Το Joomla! 1.5 κυκλοφόρησε στις 22 Ιανουαρίου 2008. Η τελευταία υποέκδοση αυτής της έκδοσης ήταν η 1.5.26 στις 27 Μαρτίου 2012. Αυτή η έκδοση ήταν η πρώτη για την επίτευξη μακροπρόθεσμης υποστήριξης (LTS – Long Time Support). Οι LTS εκδόσεις δημοσιεύονται πλέον κάθε τρεις μεγάλες ή μικρές εκδόσεις και υποστηρίζονται έως και τρεις μήνες αφού δημοσιευτεί η επόμενη έκδοση LTS.

Το Joomla! 1.6 κυκλοφόρησε στις 10 Ιανουαρίου 2011. Αυτή η έκδοση προσέθεσε λειτουργικότητα λίστας πλήρους ελέγχου πρόσβασης ενώ επίσης μπορεί να καθοριστεί από το χρήστη ιεραρχία στις κατηγορίες άρθρων. Τέλος, υπάρχουν βελτιώσεις στο διαχειριστικό interface.

Το Joomla! 1.7 κυκλοφόρησε στις 19 Ιουλίου 2011, έξι μήνες μετά την 1.6.0. Αυτή η έκδοση προσέθεσε βελτιωμένη ασφάλεια και βελτιωμένα εργαλεία για μελλοντικές αναβαθμίσεις.

Το Joomla! 2.5 κυκλοφόρησε στις 24 Ιανουαρίου 2012, έξι μήνες μετά την 1.7.0. Αυτή η έκδοση είναι μια έκδοση μακροπρόθεσμης υποστήριξης (LTS). Αρχικά αυτή η έκδοση επρόκειτο να ονομαστεί 1.8.0, ωστόσο, οι προγραμματιστές ανακοίνωσαν ότι θα μετονομαστεί ώστε να μετατραπεί σε νέο καθεστός αριθμών έκδοσης στο οποίο κάθε έκδοση LTS θα είναι και μία X.5 έκδοση. Αυτή η έκδοση ήταν η πρώτη που τρέχει και σε άλλες βάσεις δεδομένων εκτός από MySQL (πχ MS SQL).

Το Joomla! 3.0 κυκλοφόρησε στις 27 Σεπτεμβρίου του 2012. Στις 24 Δεκεμβρίου 2012 αποφασίστηκε να προστεθεί μελλοντικά μία ακόμη έκδοση (3.2) στην 3.x σειρά για να βελτιωθεί ο κύκλος ζωής της αναπτυξής της και να επεκταθεί η υποστήριξη των LTS εκδόσεων. Αυτό πρόκειται να εφαρμοστεί επίσης και για τις σειρές 4.x.

Το Joomla! 3.1 κυκλοφόρησε στις 24 Απριλίου 2013. Η έκδοση 3.1 περιλαμβάνει αρκετά νέα χαρακτηριστικά, όπως σύστημα tagging.

2.4 Τρόποι Εγκατάστασης

Όπως πολλές άλλες δημοφιλείς διαδικτυακές εφαρμογές, το Joomla! μπορεί να τρέξει σε μια στοίβα LAMP. Μια στοίβα λογισμικού ανοιχτού κώδικα Linux, Apache HTTP Server, MySQL, PHP. Πολλοί web hosts έχουν συστήματα ελέγχου που επιτρέπουν την αυτόματη εγκατάσταση του Joomla!. Στα Windows το Joomla! μπορεί να εγκατασταθεί χρησιμοποιώντας το Microsoft Web Platform Installer, το οποίο εντοπίζει και εγκαθιστά αυτόματα τις εξαρτήσεις που λείπουν, όπως PHP και MySQL.

Το Joomla! διαθέτει ισχυρή υποστήριξη στο διαδίκτυο από hosts που ειδικεύονται σε αυτό ή το υποστηρίζουν. Πολλές ιστοσελίδες παρέχουν πληροφορίες σχετικά με την εγκατάσταση και τη διατήρηση ιστοσελίδων σε Joomla!.

2.5 Επεκτάσεις του Joomla!

Οι επεκτάσεις (extensions) του Joomla! βοηθάνε στην επέκταση των δυνατοτήτων του για τη δημιουργία ιστοσελίδων και την παροχή νέων λειτουργιών. Υπάρχουν πέντε είδη επεκτάσεων για το Joomla!: Components, Modules, Plugins, Templates και γλώσσες. Κάθε μία από αυτές τις επεκτάσεις χειρίζεται μια συγκεκριμένη λειτουργία.

- **Components (εφαρμογές):** Η μεγαλύτερη και πιο περίπλοκη επέκταση από όλες και αποτελούν αυτόνομες μονάδες. Τα περισσότερα components έχουν δύο μέρη: Το μέρος της ιστοσελίδας και το μέρος της διαχείρισης. Κάθε φορά που μία σελίδα Joomla! φορτώνεται στον περιηγητή, ένα component καλείται να καταστεί ως το κύριο σώμα της σελίδας. Τα components αποτελούν το μεγαλύτερο τμήμα μιας σελίδας, επειδή ένα component οδηγείται από ένα στοιχείο μενού και κάθε στοιχείο μενού τρέχει ένα component. Είναι ουσιαστικά το κύριο θέμα της κάθε σελίδας (μενού).
- **Plugins (πρόσθετα) :** Αυτές είναι πιο προηγμένες επεκτάσεις και αποτελούν στην ουσία χειρισμούς συμβάντων. Κατά την εκτέλεση οποιουδήποτε μέρους του Joomla!, ενός module ή ένος component, ένα συμβάν μπορεί οποιαδήποτε στιγμή και κάτω από συγκεκριμένες συνθήκες να προκληθεί. Στην περίπτωση αυτή, όσα plugins έχουν καταχωρηθεί με την εφαρμογή ώστε να χειριστουν αυτό το γεγονός εκτελούνται. Για παράδειγμα, ένα plugin θα μπορούσε να χρησιμοποιηθεί για να εμποδίσει το χρήστη να υποβάλει άρθρα και να φιλτράρει τις κακές λέξεις.
- **Templates (πρότυπα):** Επεκτάσεις οι οποίες περιγράφουν τον κύριο σχεδιασμό μιας Joomla ιστοσελίδας και επιτρέπουν στους χρήστες να αλλάζουν την εμφάνισή της. Οι χρήστες βλέπουν τα modules και τα components ως μέρος ενός προτύπου. Τα πρότυπα καθορίζουν την οπτική απεικόνιση ενός δικτυακού τόπου.
- **Modules (ενθέματα):** Τα modules είναι επεκτάσεις, οι οποίες συνδέονται με τα Joomla! components για να αναδείξουν νέο περιεχόμενο ή νέες εικόνες. Τα modules μοιάζουν με πλαίσια - όπως η "αναζήτηση" ή η "σύνδεση χρήστη". Ωστόσο, δεν αποτελούν αυτόνομες μονάδες καθώς χρησιμοποιούν ήδη εγκατεστημένους πόρους και λειτουργίες. Μπορούμε να έχουμε ταυτόχρονα ένα ή περισσότερα modules σε μια σελιδα Joomla!.
- **Γλώσσες :** Πολύ απλές επεκτάσεις που χρησιμοποιούνται για αλλαγή γλώσσας και μπορεί να χρησιμοποιηθούν είτε ως βασικό μέρος ή ως επέκταση. Οι πληροφορίες γλώσσας και γραμματοσειράς μπορούν επίσης να χρησιμοποιηθούν κατά τη μετατροπή αρχείων PDF ή PSD σε Joomla! templates.

Για τους σκοπούς της παρούσας εργασίας αναπτύχθηκε ένα Joomla! Component. Παρακάτω θα εστιάσουμε στην εκτενέστερη περιγραφή ενός Component και στο πώς αυτό λειτουργεί σε συνδυασμό με το framework του Joomla!. Επίσης, θα δούμε αναλυτικά τη γενική δομή του και το πώς αυτό συμπεριφέρεται κατά το rendering και την περιήγηση του χρήστη.

2.6 Περιγραφή ενός Joomla! Component

Ένα component (εφαρμογή) είναι ένα είδος Joomla! επέκτασης [4]. Τα components είναι οι κύριες λειτουργικές μονάδες του Joomla!. Μια εύκολη συσχέτιση θα ήταν να πούμε ότι το Joomla! είναι το λειτουργικό σύστημα και τα components είναι desktop εφαρμογές. Τα components παρουσιάζονται συνήθως στο κέντρο της κύριας περιοχής του περιεχομένου ενός προτύπου, ενώ αποτελούνται από δύο κύρια μέρη: ένα διαχειριστικό και ένα δημόσιο μέρος (το μέρος που φαίνεται στο μέρος της ιστοσελίδας που βλέπουν οι απλοί επισκέπτες της). Το δημόσιο μέρος είναι αυτό που χρησιμοποιείται για να αποδώσει τις σελίδες όταν δεν καλούνται κατά την κανονική λειτουργία της ιστοσελίδας. Το τμήμα διαχείρισης παρέχει μια διεπαφή προς τους διαχειριστές ώστε να ρυθμίζουν και να διαχειρίζονται τις διάφορες παραμέτρους του component και είναι προσβάσιμο μέσω της εφαρμογής διαχείρισης του Joomla! (administration). Το Joomla! περιέχει από μόνο του μια σειρά από βασικά components, όπως το σύστημα διαχείρισης περιεχομένου, φόρμες επικοινωνίας και διαχείριση συνδέσμων.

Τα Components του Joomla! είναι οι εφαρμογές εκείνες που ενσωματώνονται στον κορμό του CMS και του δίνουν τη δυνατότητα να εκτελέσει κάποιες εργασίες, ανάλογα με τις δυνατότητες του component. Κάθε component, για να είναι συμβατό με το Joomla! αλλά και για να είναι εύκολα επεκτάσιμο, πρέπει να ακολουθεί κάποιες συμβάσεις (conventions) ώστε να μπορέσει να ενσωματωθεί στο Joomla!.

Στη συνέχεια, ξεκινώντας από τη γενικότερη δομή του και συνοψίζοντας στα ειδικότερα σημεία, θα γίνει μια προσπάθεια να εξηγηθεί η λογική με την οποία λειτουργεί ένα Joomla! Component, αλλά και το Joomla! γενικότερα.

Ο λόγος για την επιτυχία του οφείλεται στις δυνατότητες προσαρμογής που προσφέρει. Ειδικότερα, παρέχει μεγάλη δυνατότητα εξατομίκευσης γραφικών, μέσω του μηχανισμού των προτύπων του συστήματος, καθώς και ένα λεπτομερές σύστημα επέκτασης που επιτρέπουν την τροποποίηση κάθε τμήματος της ιστοσελίδας ή της σελίδας γενικότερα, χωρίς να αλλοιώνεται ο αρχικός κώδικας του CMS.

Το δημόσιο και το διαχειριστικό μέρος του Joomla! κατασκευάζονται με παρόμοιες τεχνολογίες, συνεπώς οι όποιες επεκτάσεις μπορούν να χρησιμοποιηθούν για να ολοκληρωθεί η λειτουργικότητα και των δύο πλευρών.

Το Joomla! Component ακολουθεί τη λογική του MVC (Model – View – Controller). Αυτό σημαίνει πως υπάρχουν Models, Controllers και Views τα οποία είναι κλάσεις/επεκτάσεις του κώδικα του Joomla και ουσιαστικά είναι υπεύθυνες για τη διαχείριση της πληροφορίας, την ανακατεύθυνση του χρήστη και την παράθεση των πληροφοριών σε αυτόν.

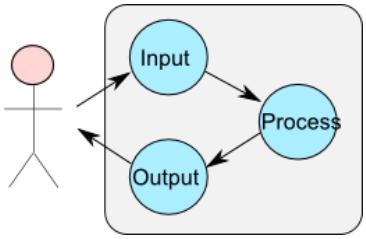
Στη συνέχεια θα εξηγήσουμε τη λογική ενός MVC framework, όπως χρησιμοποιείται από τα components του Joomla!, καθώς όλα τα components ακολουθούν συμβατικά τη λογική αυτή.

2.7 Ανάπτυξη ενός MVC Component

[5] Ένα πλαίσιο λογισμικό είναι η βάση ανάπτυξης μιας εφαρμογής που μπορεί να χρησιμοποιηθεί από έναν προγραμματιστή. Το framework που χρησιμοποιείται στο Joomla! 1.5 παρέχει μεγάλες δυνατότητες. Ο κώδικας του Joomla! έχει σχεδιαστεί λαμβάνοντας υπόψιν τον παράγοντα της επεκτασιμότητας. Το παρόν κεφάλαιο περιγράφει τη διαδικασία ανάπτυξης ενός component (εφαρμογής) χρησιμοποιώντας το MVC framework που χρησιμοποιείται από το Joomla.

2.7.1 Εισαγωγή στο Model-View-Controller

Αν και η ιδέα πίσω από ένα component μπορεί να φαίνεται εξαιρετικά απλή, ο κώδικας μπορεί γρήγορα να γίνει πολύ περίπλοκος, όσο επιπλέον χαρακτηριστικά προστίθενται ή όσο περισσότερο προσαρμόζεται η διεπαφή.



Ως Model-View-Controller (στο οποίο θα αναφερόμαστε στη συνέχεια ως MVC) ορίζουμε ένα πρότυπο πλαίσιο λογισμικού που μπορεί να χρησιμοποιηθεί για την οργάνωση του κώδικα κατά τέτοιο τρόπο ώστε η λογική της σύνθεσης των πληροφοριών και η παρουσίαση των δεδομένων να είναι ξεχωριστά μεταξύ τους. Το σκεπτικό πίσω από αυτή την προσέγγιση είναι ότι εάν η σύνθεση των πληροφοριών είναι αποσπασμένη σε ένα ξεχωριστό τμήμα, τότε η διεπαφή και η αλληλεπίδραση των χρηστών που περιβάλλει τα δεδομένα μπορεί να αναθεωρηθεί και να προσαρμοστεί χωρίς να χρειάζεται να επαναπρογραμματιστεί η λογική της σύνθεσης. Τα δεδομένα και ο τρόπος που αυτά παρουσιάζονται είναι έτσι απεξαρτημένα από τον τρόπο σύνθεσής και διαχείρισής τους. Το MVC αναπτύχθηκε αρχικά για να χαρτογραφήσει παραδοσιακές μεθόδους εισαγωγής, επεξεργασίας και παρουσίασης δεδομένων σε μια αρχιτεκτονική λογικής GUI. Παρακάτω παρουσιάζονται οι τρεις κύριοι ρόλοι που αποτελούν τη βάση του Joomla MVC.

2.7.1.1 Model

Model είναι το μέρος του component που συμπυκνώνει τα δεδομένα της εφαρμογής. Παρέχει συχνά ρουτίνες για τη διαχείριση και το χειρισμό των δεδομένων αυτών με ουσιαστικό τρόπο, ενώ περιέχει επιπλέον ρουτίνες που ανακτούν δεδομένα μέσω αυτού. Για παράδειγμα, το Model μπορεί να περιλαμβάνει μεθόδους για να προσθέτει, να ενημερώνει και να αφαιρεί πληροφορίες από και προς τη βάση δεδομένων. Σε γενικές γραμμές, η υποκείμενη τεχνική πρόσβασης στα δεδομένα θα πρέπει να είναι έγκλειστη στο Model. Με τον τρόπο αυτό, αν μια εφαρμογή πρόκειται να μετακινηθεί από ένα σύστημα που χρησιμοποιεί ένα απλό αρχείο για την αποθήκευση πληροφοριών του σε ένα σύστημα που χρησιμοποιεί μια βάση δεδομένων, το Model είναι το μόνο στοιχείο που πρέπει να αλλάξει και δε χρειάζεται αλλαγή ούτε στο View ούτε στο Controller.

2.7.1.2 View

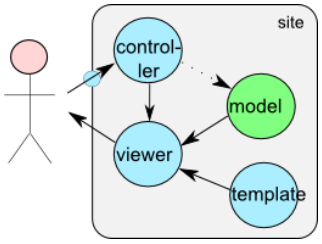
View είναι το τμήμα του component που χρησιμοποιείται για να απεικονίσει και μετατρέψει τα δεδομένα από το μοντέλο με τέτοιον τρόπο ώστε να είναι κατάλληλα για αλληλεπίδραση. Για μια web-based εφαρμογή, το View είναι γενικά μια HTML σελίδα που επιστρέφεται στο χρήστη. Το View δέχεται δεδομένα από το Model (ο οποίος διέρχεται σε αυτό από το Controller) και τα τροφοδοτεί σε ένα πρότυπο το οποίο υπόκειται σε επεξεργασία και παρουσιάζεται στο χρήστη. Το View δεν προκαλεί την επεξεργασία των δεδομένων με οποιονδήποτε τρόπο, παρά μόνο προσαρμόζει και εμφανίζει τα δεδομένα που ανακτώνται από το Model.

2.7.1.3 Controller

Το Controller είναι το τμήμα του component που είναι υπεύθυνο για την ανταπόκριση στις ενέργειες των χρηστών. Στην περίπτωση μιας εφαρμογής web, μια ενέργεια του χρήστη μπορεί να είναι μια αίτηση σελίδας. Το Controller θα καθορίσει τι αίτημα γίνεται από το χρήστη και θα ανταποκριθεί κατάλληλα προκαλώντας το Model ώστε να χειριστεί τα δεδομένα κατάλληλα και περνώντας το Model στην προβολή (View). Το Controller δεν εμφανίζει τα δεδομένα στο Model, ενεργοποιεί μόνο τις

μεθόδους στο Model που τροποποιούν τα δεδομένα, και στη συνέχεια περνάει το Model στο View, το οποίο εμφανίζει τα δεδομένα.

2.7.2 Η σύνδεση στο MVC Framework



Η απλουστευμένη εικόνα στα αριστερά απεικονίζει τα βασικά συστατικά που χρησιμοποιούνται στο Joomla. Εκτός από το Model, το View και το Controller, ένα σημείο εισόδου έχει προστεθεί το οποίο απεικονίζεται ως ένας μικρός κύκλος. Συνδεδεμένο με το χρήστη έχει προστεθεί ένα πρότυπο (template). Αυτά τα πέντε στοιχεία αποτελούν τις βασικές συνιστώσες για την ανάπτυξη ενός Joomla! MVC component.

Το Joomla! component είναι ένας ειδικός τύπος επέκτασης υπεύθυνος για την εμφάνιση των περιεχομένων μιας ολόκληρης σελίδας. Το αποτέλεσμα που παράγεται από ένα component μπορεί, επομένως, να φανεί από τη δημιουργία ενός στοιχείου μενού.

Ένα component που δεν είναι ακόμα συνδεδεμένο με το στοιχείο ενός μενού μπορεί να το δει κανείς με την κλήση της index.php σελίδας και περνώντας την παράμετρο option με το όνομα της επέκτασης που θέλει να δει. Όταν το Joomla! λάβει αυτή την παράμετρο εκτελεί τα παρακάτω βήματα:

- αναζητά το φάκελο στον οποίο έχει εγκατασταθεί το αιτούμενο extension (από τώρα και στο εξής θα το ονομάσουμε component αφού εστιάζουμε στην επεξήγηση αυτών στο παρόν κεφάλαιο)
- εκτελεί το php αρχείο με το ίδιο όνομα του component
- εφαρμόζει το πρότυπο(template) στο html περιεχόμενο που προκύπτει

Συγκεκριμένα, για το υλοποιημένο component που συνοδεύει την εφαρμογή, μπορούμε να δούμε ένα πρακτικό παράδειγμα. Τα αρχεία που διαχειρίζονται το δημόσιο τμήμα του component αποθηκεύονται στο directory:

```
/components/com_classres
```

και για το διαχειριστικό τμήμα στο:

```
/administrator/components/com_classres
```

Με την κλήση του παρακάτω URL:

```
/index.php?option=com_classres
```

θα εκτελεστούν οι παρακάτω λειτουργίες:

- γίνεται αναζήτηση του extension στη βάση δεδομένων. Αφού το option ξεκινάει από com_, τότε ξέρουμε ότι θα είναι component, αλλά αυτή η πληροφορία, όπως και το όνομα classres είναι αποθηκευμένα στη βάση δεδομένων.
- εκτελείται το αρχείο /components/com_classres/classres.php
- το αποτέλεσμα τοποθετείται στο κύριο μέρος του template της σελίδας
- η πλήρης HTML σελίδα επιστρέφεται στο πρόγραμμα περιήγησης

Παρόμοια λειτουργεί η δρομολόγηση και για το διαχειριστικό μέρος.

Κάθε extension εγκαθίσταται σε μια οποιαδήποτε εφαρμογή Joomla! από τον αντίστοιχο installer της εφαρμογής. Ο installer κατά την εγκατάσταση κατευθύνεται από ένα xml αρχείο που του δίνει πληροφορία ως προς το ποια αρχεία πρέπει να αντιγράψει σε ποιο σημείο.

3 Κώδικας και Τεκμηρίωση

Στο κεφάλαιο αυτό θα γίνει τεκμηρίωση του κώδικα που χρησιμοποιήθηκε για τη δημιουργία του component ClassRes. Για το σκοπό αυτό θα χρησιμοποιήσουμε ένα παράδειγμα για κάθε λειτουργία του component που θα αποτελεί τον οδηγό για την πλοήγησή μας μέσα από τον κώδικά του.

Θα γίνει μια προσπάθεια να καλυφθεί το μεγαλύτερο εύρος του κώδικα που αποτελεί το component. Επειδή το component κάνει χρήση βιβλιοθηκών του Joomla!, οποιαδήποτε εμβάθυνση στον τρόπο λειτουργίας των λειτουργιών αυτών βγάζει εκτός στόχου το σκοπό του παρόντος κεφαλαίου. Έτσι, θα εστιάσουμε στον κώδικα του component αυτού καθ' αυτού θεωρώντας πως οι κλήσεις στις functions του Joomla! μας επιστρέφουν το απαιτούμενο αποτέλεσμα, χωρίς να αναλύσουμε τον τρόπο που αυτό επιτυγχάνεται, εκτός αν απαιτείται για διαλεύκανση σε κάποιο μέρος της εφαρμογής μας.

3. 1 Περιγραφή Λειτουργιών

Όπως αναφέραμε στο προηγούμενο κεφάλαιο, οι ενέργειες που μπορεί να κάνει κάποιος στην εφαρμογή χωρίζονται σε απλές και διαχειριστικές. Οι απλές μπορούν να πραγματοποιηθούν από οποιονδήποτε χρήστη της εφαρμογής, ενώ για την πραγματοποίηση των διαχειριστικών χρειάζονται δικαιώματα διαχειριστή (administrator). Οι λειτουργίες αυτές αφορούν είτε το διαχειριστικό μέρος (administration - backend), είτε το δημόσιο τμήμα της ιστοσελίδας (frontend).
Έχοντας περιγράψει στο προηγούμενο κεφάλαιο τη δομή ενός component, θα συνεχίσουμε με τα παραδείγματα χρήσης και την σχετική επεξήγηση του αντίστοιχου κώδικα, παρουσιάζοντας βήμα βήμα κάθε πιθανή λειτουργία των χρηστών, διαχειριστών ή απλών.

Οι ενέργειες που μπορεί να πραγματοποιήσει κάποιος χρήστης χρησιμοποιώντας την εφαρμογή ClassRes κατηγοριοποιούνται και συνοψίζονται στις παρακάτω:

Απλές

- Προβολή πληροφοριών
    - Προβολή λίστας αιθουσών
    - Προβολή στοιχείων συγκεκριμένης αίθουσας
- Δημιουργία αίτησης κράτησης
    - Αποστολή αίτησης
    - Επιβεβαίωση αίτησης
- Επεξεργασία αίτησης κράτησης
    - Προβολή αίτησης κράτησης
    - Αίτηση αλλαγής στοιχείων κράτησης
    - Ακύρωση κράτησης

Διαχειριστικές

- Προβολή πληροφοριών
    - Προβολή λίστας αιθουσών
    - Προβολή λίστας αιτήσεων κρατήσεων
    - Προβολή λίστας αιτήσεων κρατήσεων αρχείου
- Δημιουργία νέων στοιχείων
    - Δημιουργία νέας αίθουσας
    - Δημιουργία νέας κράτησης
- Επεξεργασία πληροφοριών
    - Επεξεργασία στοιχείων αίθουσας
    - Επεξεργασία στοιχείων αίτησης κράτησης
    - Επεξεργασία στοιχείων αίτησης κράτησης αρχείου

Στη συνέχεια θα αναλύσουμε τον κώδικα για την επίτευξη των παραπάνω λειτουργιών ξεκινώντας από τις απλές λειτουργίες, τις λειτουργίες δηλαδή στις οποίες έχουν πρόσβαση όλοι οι χρήστες της εφαρμογής. Κατά τη διάρκεια της επεξήγησης των λειτουργιών αυτών θα εμβαθύνουμε και στις λεπτομέρειες της υλοποίησής τους.

3.2 Ανάλυση Κώδικα

Παρακάτω ξεκινάμε να αναλύουμε τον κώδικα της εφαρμογής ClassRes σε ό,τι αφορά τις λειτουργίες στις οποίες έχει πρόσβαση ο απλός χρήστης της εφαρμογής. Οι λειτουργίες αυτές πραγματοποιούνται αποκλειστικά στο δημόσιο μέρος της ιστοσελίδας που χρησιμοποιεί την εφαρμογή.

Για τη διευκόλυνση της κατανόησης του τρόπου λειτουργίας της εφαρμογής θα θεωρήσουμε έναν χρήστη ο οποίος έχει μόλις επισκεφτεί την ιστοσελίδα και ενδιαφέρεται να επιλέξει κάποια αίθουσα από τις παρεχόμενες, να αποστείλλει την αίτηση κράτησής του και στη συνέχεια να ελέξγει την κατάσταση της αίτησής του, ή να αποστείλλει αλλαγές για αυτή. Τέλος, ο χρήστης θα αποφασίσει να ακυρώσει την κράτηση.

Με το παραπάνω παράδειγμα θα κάνουμε χρήση όλων των λειτουργιών που είναι διαθέσιμες προς τον χρήστη, καλύπτοντας όλο το φάσμα χρήσης για το δημόσιο μέρος (public frontend) της εφαρμογής, αλλά και για το διαχειριστικό (admin backend), καθώς χρειάζεται και δράση από τον διαχειριστή για την επιβεβαίωση της κράτησης.

Να σημειωθεί πως, χάριν απλότητας και προς αποφυγή επαναλήψεων δεδομένου ότι αρκετά κομμάτια στον κώδικα επαναλαμβάνονται σε κάποιες λειτουργίες, οι συγκεκριμένες εντολές θα εξηγούνται την πρώτη φορά μόνο και στη συνέχεια θα παραλείπεται η επεξήγησή τους, θεωρούμενες ως ήδη εξηγημένες.

3.2.1 Λειτουργίες χρήστη (User functions)

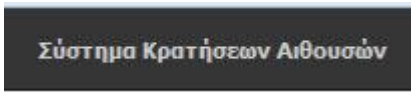
Στο κεφάλαιο αυτό θα περιγράψουμε όλες τις λειτουργίες που μπορεί να εκτελέσει ο χρήστης, επισκεπτόμενος την ιστοσελίδα και το component που αναπτύχθηκε.

3.2.1.1 Προβολή Πληροφοριών

Η πιο βασική λειτουργία, αφορά την παρουσίαση στο χρήστη των πληροφοριών που επιθυμεί, ώστε να ενημερωθεί ανάλογα και να πλοηγηθεί στην ιστοσελίδα. Για την εφαρμογή ClassRes, ο χρήστης μπορεί να αιτηθεί να δει μια λίστα όλων των αιθουσών που του είναι διαθέσιμες, καθώς και κάποια συγκεκριμένη, όταν θελήσει να δει πιο συγκεκριμένες πληροφορίες για κάποια.

3.2.1.1.1 Προβολή λίστας αιθουσών

Ο χρήστης έχει μόλις επισκεφτεί την ιστοσελίδα και ενδιαφέρεται να δει τις παρεχόμενες αίθουσες ώστε να επιλέξει από αυτές εκείνη που καλύπτει καλύτερα τις ανάγκες του. Για το σκοπό αυτό, θα πατήσει στο μενού **"Σύστημα Κρατήσεων Αιθουσών"** από το μενού της ιστοσελίδας.



Εικόνα 1. Επιλογή του μενού λίστας αιθουσών από το χρήστη

Ο σύνδεσμος του μενού είναι ο παρακάτω:

```
/index.php?option=com_classres&view=classrooms&Itemid=105
```

Οι παράμετροι του url έχουν τις παρακάτω τιμές:

option: com_classres
view: classrooms

Η παράμετρος `Itemid` αφορά το αναγνωριστικό id του μενού στο οποίο ο χρήστης έκανε click. Η παράμετρος αυτή χρησιμοποιείται για λόγους που δεν αφορούν τη λειτουργία της εφαρμογής ClassRes και είναι εκτός του πεδίου που περιγράφει το παρόν κεφάλαιο. Για το λόγο αυτό δε θα ασχοληθούμε περαιτέρω με την εν λόγω παράμετρο.

Όπως εξηγήθηκε στο προηγούμενο κεφάλαιο, θέτοντας στο option την τιμή com_classres, ο Controller του Joomla! θα ψάξει στο directory "components" για το φάκελο **com_classres**, και θα εκτελέσει το αρχείο **classres.php**.

Το παραπάνω αρχείο θα εκτελεστεί, και συγκεκριμένα θα εκτελεστούν οι παρακάτω εντολές, οι οποίες και περιγράφονται παρακάτω:

```
defined('_JEXEC') or die('Restricted access');
```

Η εν λόγω εντολή ελέγχει εάν έγινε απευθείας πλοήγηση στο εν λόγω αρχείο και αποτρέπει την εκτέλεσή του σε μια τέτοια περίπτωση.

```
if($controller = JRequest::getVar('controller')) {
     require_once
(JPATH_COMPONENT.DS.'controllers'.DS.$controller.'.php');
}
$classname = 'ClassresController'.$controller;
$controller = new $classname();
```

Με τις παραπάνω γραμμές κώδικα δημιουργείται ο Controller της εφαρμογής μας

```
$controller->execute( JRequest::getCmd('task') );
```

Με την εντολή αυτή εκτελείται από τον Controller η λειτουργία που δηλώνεται από την παράμετρο **task.** Η παράμετρος αυτή δεν εχει δηλωθεί στο σύνδεσμο του μενού, επομένως παίρνει την default τιμή **display**.

Ο Controller δηλώνεται στο αρχείο **controller.php** και καλείται να εκτελέσει τη function **display,** της οποίας ο κώδικας παρουσιάζεται παρακάτω:

```
     public function display($cachable = false, $urlparams = false) {
          JRequest::setVar('view', JRequest::getCmd('view',
'classrooms'));
          parent::display($cachable);
     }
```

Η πρώτη εντολή θέτει στο view την τιμή «classrooms», εάν δεν έχει τεθεί ήδη (έχουμε ήδη τιμή για το view: classrooms), ενώ η δεύτερη καλεί τη function **display** του parent της κλάσης **ClassresController**. Ο Controller της εφαρμογής Classres είναι extension του **JController**, δηλαδή του Controller του Joomla. Η function αυτή καλεί το αντίστοιχο view που επιθυμούμε, το View δηλαδή για τη λίστα των αιθουσών.

Η View class βρίσκεται στο directory "views/**classrooms**/view.html.php". Ο κώδικάς της παρουσιάζεται παρακάτω:

```
defined('_JEXEC') or die('Restricted access');

jimport( 'joomla.application.component.view');
jimport( 'joomla.html.pagination');

class ClassresViewClassrooms extends JView {
      function display($tpl = null)
      {
            $this->items = $this->get('Items');
            $this->pagination = $this->get('Pagination');
            $this->assignRef('items', $this->items);
            $this->assignRef('pagination', $this->pagination);
            parent::display($tpl);
      }

}
```

Οι νέες εντολές αναλύονται μία προς μία στη συνέχεια

```
jimport( 'joomla.html.pagination');
```

Με την εντολή αυτή καλείται η κλάση **Pagination** του Joomla!, που αναλαμβάνει τη σελιδοποίηση των αντικειμένων της σελίδας. Με την κλήση της function **display** της ClassresViewClassrooms, εκτελούνται οι παρακάτω γραμμές κώδικα:

```
$this->items = $this->get('Items');
$this->pagination = $this->get('Pagination');
$this->assignRef('items', $this->items);
$this->assignRef('pagination', $this->pagination);
parent::display($tpl);
```

Η πρώτη εντολή, που είναι και η πιο ενδιαφέρουσα, καλεί τη function **get('Items');** από το Model Classrooms και αναθέτει το αποτέλεσμα στην παράμετρο $this->items του View.

Το αρχείο του Model classrooms βρισκεται στο directory models/**classrooms**.php. Επειδή η function getItems αποτελεί inherited function του ClassresModelClassrooms Model που κληρονομεί από την JModelList, δε θα την αναλύσουμε. Αυτό που μας ενδιαφέρει στο παρόν κεφάλαιο είναι ότι επιστρέφει από τη βάση δεδομένων μας μία **Object List** με όλες τις αίθουσες (Classrooms) που είναι αποθηκευμένες στη βάση και τις αποθηκεύει στην παράμετρο $this->items.

Στη συνέχεια καλείται το αντίστοιχο Model για το Pagination. Και πάλι, η function getPagination ανήκει στην υπερκλάση JModelList και δε θα αναλυθεί.

Τέλος, καλείται η function του JModelList display, η οποία παίρνει μία παράμετρο, την $tpl. Επειδή δεν έχει δωθεί κάποια τιμή στην παράμετρο αυτή, παίρνει την τιμή default. Η function display φορτώνει το html template για την παρουσίαση των πληροφοριών στο χρήστη. Το template βρίσκεται στο directory views/**classrooms**/tmpl/**default**.php.

Το αρχείο default.php περιλαμβάνει την HTML που απαιτείται για την παρουσίαση των πληροφοριών. Στη συγκεκριμένη αναφορά δε θα γίνει ανάλυση του HTML κώδικα, ωστόσο θα παρουσιαστούν τα πιο ενδιαφέροντα σημεία του εν λόγω αρχείου ώστε να βοηθήσουν στην κατανόηση της παρουσίασης των αποτελεσμάτων. Η εντολή

```
echo JTEXT::_('COM_CLASSRES_CLASSROOMS_VIEW_DEFAULT_TITLE');
```

χρησιμοποιεί μια ενσωματωμένη λειτουργία του Joomla! ώστε να χρησιμοποιούνται σταθερές γλωσσών για την ανάπτυξη πολυγλωσσικών συστημάτων. Τα αρχεία που περιέχουν αυτές τις σταθερές, αποθηκεύονται σε αρχεία της μορφής

```
/language/<lang_code>/<lang_code>.com_classres.ini,
```

όπου <lang_code> είναι η κωδική ονομασία της εκάστοτε γλώσσας για την οποία έγινε η μετάφραση (πχ «el-GR» ή «en-GB»). Η εν λόγω λειτουργία ξεφεύγει από τους σκοπούς της αναφοράς και για το λόγο αυτό δε θα γίνει περαιτέρω αναφορά, ωστόσο ενσωματώθηκε στην εφαρμογή για λόγους πληρότητας και επεκτασιμότητας.

Τέλος, όλα τα Classroom Objects έχουν αποθηκευτεί σαν ObjectList στην παράμετρο $this->items, οπότε καλώντας την for loop

```
foreach ($this->items as $item)
```

μπορούμε να αποκτήσουμε πρόσβαση σε κάθε στοιχείο του Object $item, για παράδειγμα $item->title ή $item->intro.
Αφού εκτελεστούν όλα τα παραπάνω, το αρχείο classres.php εκτελεί δύο ακόμα λειτουργίες.

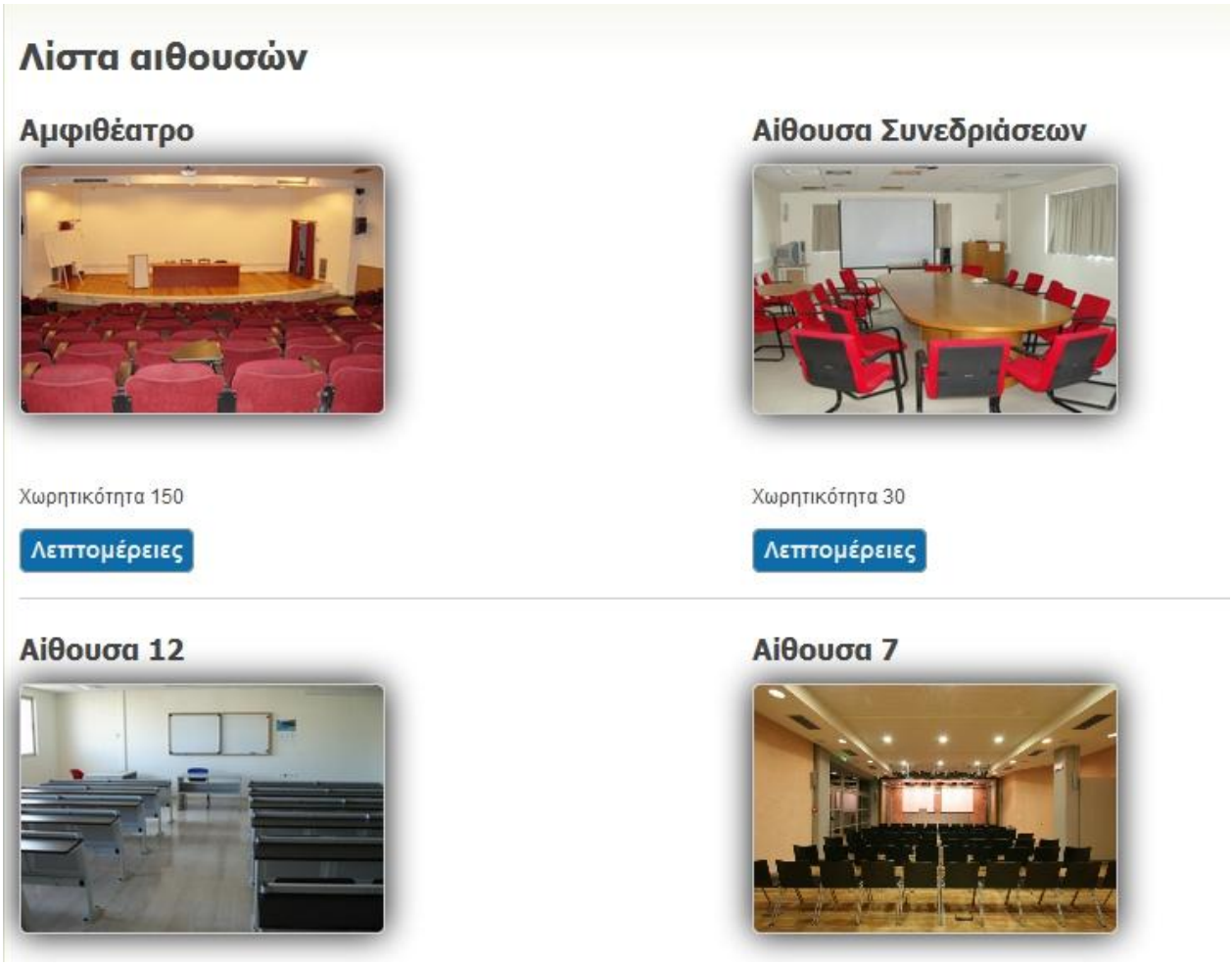
```
$controller->redirect();
```

Με την εντολή αυτή γίνεται ανακατεύθυνση, εάν έχει οριστεί κάτι τέτοιο από τον Controller.

```
$doc =& JFactory::getDocument();
$doc->addStyleSheet(
      JURI::root().'media/com_classres/css/classres.css'
);
```

Με τις δύο αυτές εντολές, αρχικά φορτώνεται το Object Document και στη συνέχεια προστίθεται το stylesheet της εφαρμογής μας σε αυτό.

Μετά από όλα αυτά, ο Controller του Joomla ομαδοποιεί όλες τις πληροφορίες και παρουσιάζει στον χρήστη το τελικό αποτέλεσμα:



Εικόνα 2. Classrooms View: Παρουσίαση Λίστας Αιθουσών

Όπως φαίνεται στην παραπάνω εικόνα, ο χρήστης έχει πρόσβαση σε όλες τις αίθουσες της σχολής, όπου μπορεί να δει τις βασικές πληροφορίες που αφορούν την καθεμία όπως κάποια φωτογραφία, τον αριθμό ατόμων που αυτή χωράει (χωρητικότητα), ένα εισαγωγικό περιγραφικό κείμενο και φυσικά τον τίτλο - ονομασία της αίθουσας.

3.2.1.1.2 Προβολή στοιχείων αίθουσας

Έστω ότι ο χρήστης ενδιαφέρεται να μάθει περισσότερες πληροφορίες για κάποια από τις αίθουσες που του εμφανίστηκαν. Για παράδειγμα, έστω ότι ενδιαφέρεται για την αίθουσα "Αμφιθέατρο", τότε θα πατήσει πάνω στο κουμπί **Λεπτομέρειες**, το οποίο θα ανακατευθύνει τον browser στο url

```
index.php?option=com_classres&view=classroom&id=2
```

Οι παράμετροι του url έχουν τις παρακάτω τιμές:

option: com_classres
view: classroom

id: 2

Έτσι, ακολουθώντας την ίδια δρομολόγηση και διαδικασία με την προηγούμενη ενέργεια, θα εκτελεστεί το αρχείο classres.php το οποίο θα καλέσει με τη σειρά του το αρχείο controller.php. Ο Controller θα ψάξει για κάποιο task και επειδή δε θα βρει κάποιο δηλωμένο θα θέσει

task: display

Η function display που θα εκτελεστεί στη συνέχεια θα ψάξει να βρει για κάποιο δηλωμένο view, και καθώς έχουμε view:classroom, θα καλέσει την **Classroom** View Class (αντίστοιχα, το αρχείο views/classroom/view.html.php).

Ο κώδικας της ClassresViewClassroom παρουσιάζεται παρακάτω:

```
function display($tpl = null) {
	$this->reservations_list = $this->get('ReservationsQuery');
	$this->item = $this->get('Data');
	$this->assignRef('item', $this->item);
	parent::display($tpl);
	$document = JFactory::getDocument();
	$document->addScript(JURI::root().
'/administrator/components/com_classres/models/forms/reservation.js');
	$document->addScript(JURI::root().
 '/administrator/components/com_classres/models/forms/validation.js');
}
```

Αναλύοντας κομμάτι κομμάτι τον κώδικα, έχουμε ότι:

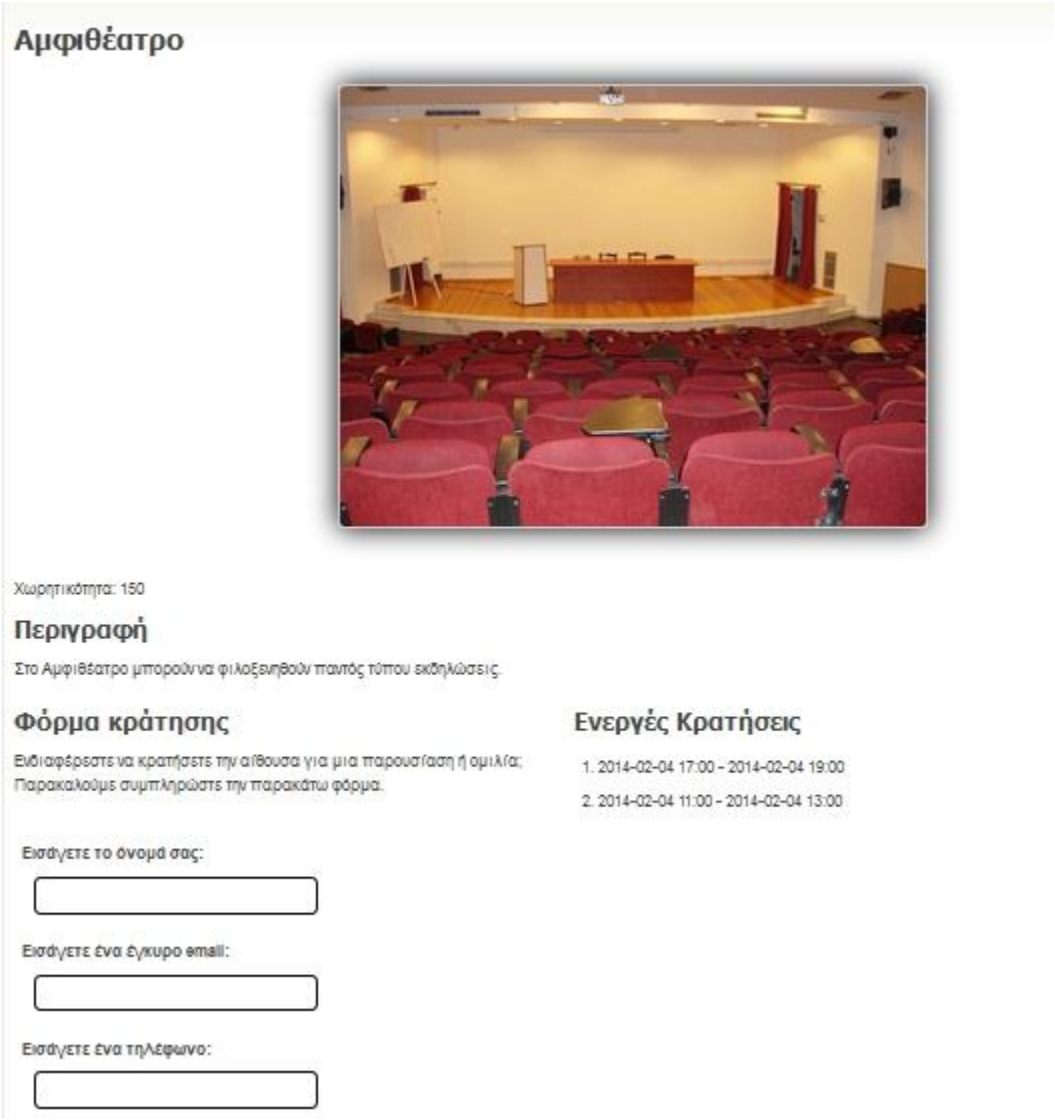
```
$item = $this->get( 'Data' );
```

Η συγκεκριμένη εντολή κάνει ότι κάνει και η `get('Items')`, με τη διαφορά ότι τώρα επιστρέφεται ένα Object και όχι ένα ObjectList, δηλαδή μόνο το αντικείμενο που μας ενδιαφέρει (το οποίο φορτώνεται στην $item).

Το επόμενο κομμάτι κώδικα της `display()` του view.html.php φορτώνει πληροφορία για όλες τις ενεργές κρατήσεις για τη συγκεκριμένη αίθουσα.

Αρχικά καλείται η `getReservationsQuery()`, η οποία επιστρέφει όλες τις ενεργές κρατήσεις. Στη συνέχεια, για κάθε κράτηση εκτελούνται οι παρακάτω εντολές που διαμορφώνουν κατάλληλα την ημερομηνία προς παρουσίαση, και το Object List που τις περιλαμβάνει φορτώνεται στο $this->reservations_list.

Το τελευταίο τμήμα του κώδικα φορτώνει κάποια αρχεία javascript για τον έλεγχο της φόρμας, και εάν δεν υπάρχει κάποιο πρόβλημα, φορτώνει το template. Από τη στιγμή που δεν έχει δηλωθεί κάποιο layout, τίθεται **layout**: default, οπότε και καλείται το αρχείο views/classroom/tmpl/default.php που παρουσιάζει τα δεδομένα όπως τα θέλουμε. Το αποτέλεσμα φαίνεται στην παρακάτω φωτογραφία:



Εικόνα 3. Classrom View: Προβολή αίθουσας

Στην παραπάνω εικόνα φαίνεται ότι βλέπει ο χρήστης πατώντας στον σύνδεσμο κάποιας αίθουσας. Φαίνεται η φωτογραφία της αίθουσας σε μεγάλο μέγεθος, η χωρητικότητα της αίθουσας, η πλήρης περιγραφή της, ενώ παράλληλα υπάρχει η φόρμα για αποστολή αίτησης κράτησης καθώς και ο τομέας που παρουσιάζει όλες τις ενεργές κρατήσεις για τη συγκεκριμένη αίθουσα.

3.2.1.2 Δημιουργία Αίτησης Κράτησης

Έστω ότι ο χρήστης αποφάσισε ότι η αίθουσα τον καλύπτει, και θέλει να αποστείλει μια αίτηση κράτησης. Αυτή την ενέργεια την εκτελεί αποστέλλοντας τη φόρμα που υπάρχει στη σελίδα της κάθε αίθουσας. Ο χρήστης συμπληρώνει υποχρεωτικά τα απαιτούμενα στοιχεία της φόρμας και προαιρετικά τα μη απαιτούμενα. Οι έλεγχοι πραγματοποιούνται ανάλογα με το javascript αρχείο validation.js.

Όταν ο χρήστης πατήσει το κουμπί της υποβολής της φόρμας και έχει συμπληρώσει επιτυχώς τη φόρμα, η φόρμα υποβάλλεται με τα εξής ορίσματα:

option: com_classres
task: classroom.submit

καθώς και μια **jform** array που περιλαμβάνει, εκτός από τις εισαγωγές του χρήστη, το id της αίθουσας (**id**: 2) και μια default τιμή για την υποβληθείσα αίτηση κράτησης (**archived**: 0).

Έτσι, με την ίδια λογική η πληροφορία δρομολογείται μέσω του classres.php στον Controller, όπου πλέον υπάρχει task: classroom.submit. Έτσι, αυτή τη φορά δεν εκτελείται το default task display, αλλά το `submit()` του classroom controller, που περιέχει τον παρακάτω κώδικα:

```
JRequest::checkToken() or jexit(JText::_('JINVALID_TOKEN'));
$app = JFactory::getApplication();
$model = $this->getModel('reservation');
$data = JRequest::getVar('jform', array(), 'post', 'array');
$insertItem = $model->insertItem($data);
if ($insertItem) {
      echo "<h2>".JTEXT::_('COM_CLASSRES_RESERVATION_SENT')."</h2>";
      echo "<p>".JTEXT::_('COM_CLASSRES_RESERVATION_SENT_DESC')."</p>";
} else {
      echo
"<h2>".JTEXT::_('COM_CLASSRES_RESERVATION_FAILED_SENT')."</h2>";
      echo
"<p>".JTEXT::_('COM_CLASSRES_RESERVATION_FAILED_DESC')."</p>";
}
return true;
```

Η πρώτη εντολή αποτελεί μια δικλείδα ασφαλείας του Joomla! για την υποβολή της φόρμας. Η δεύτερη εντολή φορτώνει το Object Application, και η τρίτη το Model Reservation. Στη συνέχεια φορτώνεται η jform array στη μεταβλητή $data, και καλείται η function του Reservation Model insertItem(). Η insertItem εκτελεί 3 βασικές λειτουργίες:

1. Ελέγχει εάν οι ημερομηνίες της κράτησης πέφτουν πάνω σε άλλες έγκυρων κρατήσεων για την ίδια αίθουσα.
2. Εάν δε βρει κάποια, εισάγει την κράτηση στο σύστημα με status: Unconfirmed
3. Στέλνει στον χρήστη ένα email επιβεβαίωσης που περιέχει ένα link που πρέπει να πατήσει ο χρήστης ώστε να επιβεβαιώσει την αίτηση κράτησης

Αφού ο χρήστης ελέγξει τα περιεχόμενα του mailbox του και εντοπίσει το email επιβεβαίωσης και κάνει κλικ στο εν λόγω link μεταφέρεται στο σύνδεσμο

`index.php?option=com_classres&view=reservation&layout=validate&code=<code>`, με τις παράμετρους

option: com_classres
task: display (default τιμή)
view: reservation
layout: validate
code: <code>

Έτσι, αφού η πληροφορία δρομολογηθεί έως την view class reservation, θα εκτελεστεί η function display όπου υπάρχει ο παρακάτω κώδικας:

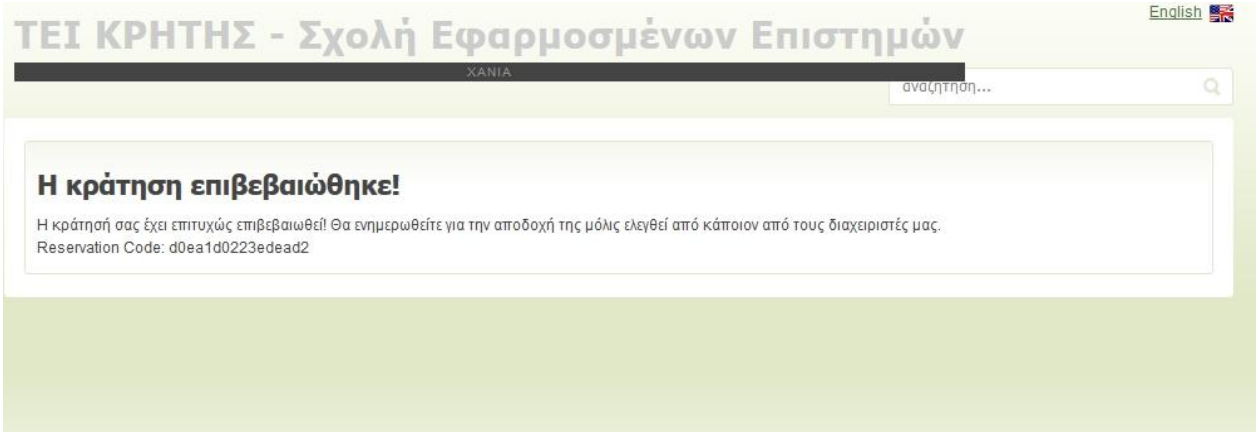
```
$layout = JRequest::getVar('layout');
if ($layout == 'validate') {
      $res_code = $this->get('ReservationConfirm');
```

```
            if (!$res_code) {
                $this->confirm = false;
            } else {
                $this->confirm = true;
                $this->res_code = $res_code;
            }
        }
```

ο οποίος θέτει τις ανάλογες μεταβλητές αναλόγως το αποτέλεσμα της εντολής

```
$res_code = $this->get('ReservationConfirm');
```

Η εντολή αυτή καλεί την `getReservationConfirm()`, η οποία διασταυρώνει εάν η παράμετρος code του url αντιστοιχίζεται σε κάποιο reservation code στη βάση δεδομένων. Εάν αντιστοιχίζεται, τότε το status της αίτησης κράτησης αλλάζει σε Pending και ο κωδικός σβήνεται, ενώ παράγεται ένας νέος κωδικός κράτησης και φορτώνεται στην $res_code, αλλιώς παίρνει την τιμή false. Στη συνέχεια καλείται το template `validate`, όπου προβάλλεται το αντίστοιχο μήνυμα επιτυχίας ή αποτυχίας επιβεβαίωσης της αίτησης κράτησης.



Εικόνα 4. Επιβεβαίωση αποστολής αίτησης κράτησης

3.2.1.3 Επεξεργασία Αίτησης Κράτησης

3.2.1.3.1 Προβολή Αίτησης Κράτησης

Μετά την επιβεβαίωση της επιτυχημένης αποστολής αίτησης κράτησης από τον χρήστη, εκρεμμεί η έγκριση ή η απόρριψή της από κάποιο διαχειριστή. Ωστόσο, από αυτό το σημείο και μετά μπορεί να παρακολουθεί κατ' επιλογήν του την πορεία και τις λεπτομέρειες της αίτησης κράτησής του μέσα από τον κωδικό κράτησης που δημιουργήθηκε στο προηγούμενο βήμα.

Για κάθε επιβεβαιωμένη κράτηση παράγεται ένας μοναδικός αλφαριθμητικός κωδικός που μπορεί να χρησιμοποιήσει ο αιτών για να δει τις λεπτομέρειες της αίτησης κράτησής του.

Αυτό γίνεται από το δεύτερο μενού επιλογής που παρέχει η εφαρμογή. Στην ιστοσελίδα που δημιουργήθηκε για την υλοποίηση της συγκεκριμένης διπλωματικής, το μενού ονομάζεται **Reservation Display**, αλλά αυτό που ουσιαστικά μας ενδιαφέρει είναι το url στο οποίο οδηγείται ο χρήστης όταν κάνει κλικ σε αυτό.

Έτσι λοιπόν, όταν ο χρήστης κάνει κλικ στο μενού Reservation Display, οδηγείται στο παρακάτω url:

```
index.php?option=com_classres&view=reservation&Itemid=160
```

Σύμφωνα με τα όσα έχουμε αναφέρει ως τώρα, έχουμε τις παρακάτω παραμέτρους από το url:

option: com_classres
task: display
view: reservation
layout: default

Σύμφωνα με τα παραπάνω, ο Controller της ClassRes θα καλέσει την View Class του Reservation, η οποία θα εκτελέσει την display function. Ο κώδικας της `display()` παρουσιάζεται παρακάτω και αναλύεται στη συνέχεια.

```
$layout = JRequest::getVar('layout');
if ($layout == 'validate') {
      $res_code = $this->get('ReservationConfirm');
      if (!$res_code) {
            $this->confirm = false;
      } else {
            $this->confirm = true;
            $this->res_code = $res_code;
      }
} else {
      $reservation = $this->get('ReservationByCode');
      if ($reservation) {
            $this->found = true;
            $this->assignRef('reservation', $reservation);
      } else {
            $this->found = false;
      }
}

if (count($errors = $this->get('Errors'))) {
      JError::raiseError(500, implode('<br />', $errors));
      return false;
}
parent::display($tpl);
$document = JFactory::getDocument();
$document->addScript(JURI::root().
'/administrator/components/com_classres/models/forms/reservation.js');
$document->addScript(JURI::root().
'/administrator/components/com_classres/models/forms/validation.js');
```

Η πρώτη εντολή

```
            $layout = JRequest::getVar('layout');
```

αποθηκεύει την τιμή του layout (στη συγκεκριμένη περίπτωση **default**) στην $layout.
Στη συνέχεια έχουμε τη συνθήκη που παρουσιάστηκε στο προηγούμενο βήμα

```
if ($layout == 'validate')
```

η οποία δεν ισχύει αυτή τη φορά, καθώς $layout == default. Έτσι θα εκτελεστεί το else της συνθήκης. Η πρώτη εντολή στο else block

```
$reservation = $this->get('ReservationByCode');
```

καλεί τη function `getReservationByCode()` από το model Reservation, η οποία παίρνει από το url την παράμετρο code, εάν υπάρχει, και ελέγχει εάν υπάρχει κάποια κράτηση που να συνδέεται με τον κωδικό αυτόν. Στη συγκεκριμένη περίπτωση δεν υπάρχει, επομένως η variable $reservation τίθεται **false**, κάτι που μας οδηγεί στο else και της επόμενης συνθήκης

```
if ($reservation)
```

δηλαδή στο

```
$this->found = false;
```

Στο τελευταίο κομμάτι του κώδικα απλά ελέγχεται εάν έχει συμβεί κάποιο σφάλμα και παρουσιάζεται το template που βρίσκεται στο directory views/reservation/default.php.

Το συγκεκριμένο template εμφανίζει διαφορετική πληροφορία ανάλογα με την τιμή του $this->found. Στη συγκεκριμένη περίπτωση έχει την τιμή **false**, επομένως παρουσιάζεται το αντίστοιχο κομμάτι από το template. Αυτό είναι μια απλή φόρμα με ένα μόνο πεδίο input, 2 hidden πεδία και ένα submit κουμπί. Η πληροφορία της φόρμας είναι η παρακάτω:

code: <input χρήστη>
option: com_classres
view: reservation

Στο πεδίο code, ο χρήστης καλείται να εισάγει τον κωδικό της κράτησής του ώστε να τον υποβάλλει και να επιστραφεί πληροφορία για την αίτηση κράτησής του. Παρατηρώντας τα στοιχεία, φαίνεται καθαρά ότι ο χρήστης, εισάγωντας στο πεδίο τον κωδικό και πατώντας το κουμπί υποβολής, θα οδηγηθεί ξανά στην ίδια σελίδα, καθώς οι παράμετροι είναι ίδιοι. Ωστόσο, υπάρχει μία επιπλέον παράμετρος και αυτή είναι η παράμετρος code. Αν παρατηρήσουμε τον προηγούμενο κώδικα, εντοπίζουμε το πρώτο σημείο που αυτή η παράμετρος επηρεάζει το αποτέλεσμα στον κώδικα της `display()` της Reservation View Class, και συγκεκριμένα στην εντολή

```
$reservation = $this->get('ReservationByCode');
```

Η function `getReservationByCode()` βρίσκεται στο model Reservation στο directory models/reservation.php, και ο κώδικάς της είναι ο παρακάτω:

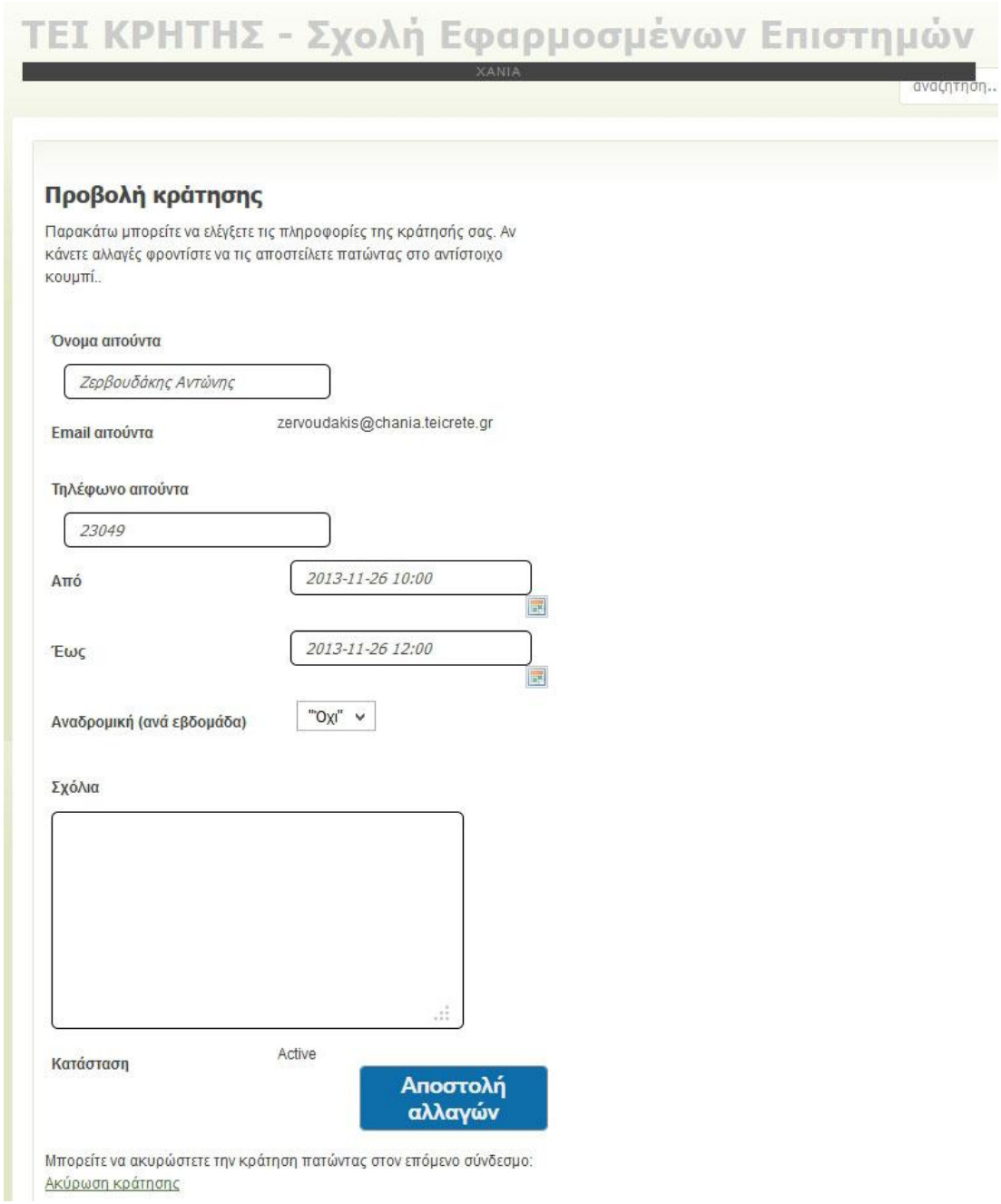
```
$code = JRequest::getVar('code');
$query = " SELECT id, classroom_id, from_timestamp, to_timestamp,
reserver_name, reserver_email, reserver_contact, reserver_comments,
status, recurse, recurse_day, recurse_from, recurse_to, res_code "
                . " FROM #__classres_reservations "
                . " WHERE res_code'".$code."'";
```

```
$reservation = $this->_getList( $query );
return $reservation;
```

Φαίνεται από την πρώτη εντολή ότι ο κώδικας ψάχνει για την παράμετρο code. Αυτή τη στιγμή η παράμετρος είναι διαθέσιμη καθώς την έχει πληκτρολογήσει ο χρήστης στο πεδίο, επομένως στα επόμενα βήματα γίνεται ένα query στη βάση δεδομένων και επιστρέφονται τα στοιχεία της κράτησης στην οποία αντιστοιχεί ο κωδικός που πληκτρολογήθηκε. Εάν βρεθεί κράτηση που αντιστοιχίζεται με τον κωδικό, αποθηκεύεται σαν Object στο $reservation, αλλιώς αυτό παίρνει την τιμή false (όπως έγινε στο προηγούμενο βήμα).

Τα επόμενα βήματα του κώδικα της `display()` φορτώνουν την αντίστοιχη φόρμα του Reservation Model στο $this->form, θέτουν $this->found = true και απλά μεταφέρουν τα στοιχεία του reservation στην κλάση του μοντέλου για πιο εύκολη χρήση.

Στη συνέχεια, αφού ελεγχθεί εάν έχει συμβεί κάποιο σφάλμα, παρουσιάζεται το template, όπου αυτή τη φορά, επειδή έχουμε $this->found == true, θα παρουσιαστεί το άλλο κομμάτι του, το οποίο είναι η φόρμα που παρουσιάζεται στην παρακάτω εικόνα.



Εικόνα 5. Προβολή στοιχείων κράτησης

Από την παραπάνω εικόνα φαίνεται ότι ο χρήστης μπορεί να ελέγξει τα στοιχεία της αίτησης κράτησής του, να αλλάξει τα περισσότερα από αυτά και να στείλει αίτηση αλλαγής των στοιχείων, να παρακολουθήσει σε ποιά κατάσταση βρίσκεται η αίτηση ή να την ακυρώσει. Το μόνο πεδίο που δεν μπορεί να αιτηθεί αλλαγή είναι το email, καθώς μετά την επιβεβαίωση του αυτό κλειδώνει με την κράτηση και αντιστοιχίζεται με αυτήν. Τις ενέργειες που μπορεί να κάνει ο χρήστης από αυτή τη σελίδα θα τις εξετάσουμε στα επόμενα δύο υποκεφάλαια.

3.2.1.3.2 Αίτηση Αλλαγής Στοιχείων Κράτησης

Έστω ότι ο χρήστης αποφασίζει να αλλάξει κάποιο ή κάποια από τα στοιχεία που συνοδεύουν την αίτηση κράτησής του. Θα μπορούσε για παράδειγμα να αλλάξει το τηλέφωνο επικοινωνίας, ή ακόμα και την ώρα ή ημέρα για την οποία χρειάζεται την αίθουσα. Τέλος, μπορεί να θέλει απλά να στείλει κάποιο επιπλέον σχόλιο στο διαχειριστή σχετικά με την κράτηση αυτή.

Αυτό μπορεί να το κάνει από τη συγκεκριμένη φόρμα. Ωστόσο, οποιαδήποτε αίτηση για αλλαγή πρέπει να εξεταστεί και να εγκριθεί από κάποιον διαχειριστή, καθώς η κράτηση είναι επιβεβαιωμένη και πιθανώς να έχει εγκριθεί από τη Διαχείριση. Έτσι, ο διαχειριστής λαμβάνει το αίτημα αλλαγής του χρήστη σε email, το οποίο περιέχει τα στοιχεία της φόρμας.

Εάν τα στοιχεία άλλαζαν κατευθείαν από το χρήστη, θα μπορούσε πολυ εύκολα να χρησιμοποιήσει ψευδή στοιχεία αρχικά για να εγκριθεί η αίτηση, και στη συνέχεια να τα αλλάξει. Ωστόσο κάτι τέτοιο τώρα αποφεύγεται. Έτσι, πατώντας στο κουμπί αποστολής της φόρμας, η φόρμα υποβάλεται με τις παρακάτω παραμέτρους:

option: com_classres
task: reservation.submit
view: reservation
layout: default
jform[] array

Η φόρμα θα υποβληθεί και θα περάσει από τη function `submit()` του Reservation Controller. Ο κώδικας της `submit()` χωρίζεται σε 3 τμήματα. Στο πρώτο

```
JRequest::checkToken() or jexit(JText::_('JINVALID_TOKEN'));
$app = JFactory::getApplication();
$model = $this->getModel('reservation');
$data = JRequest::getVar('jform', array(), 'post', 'array');
```

κάθε εντολή εκτελεί αρχικοποιήσεις. Η πρώτη ελέγχει εάν η φόρμα υποβλήθηκε μέσω του Joomla! (για λόγους ασφαλείας), η δεύτερη αρχικοποιεί το object Application που χρειάζεται στη συνέχεια, η τρίτη αρχικοποιεί το Reservation Model και η τέταρτη φορτώνει στο $data την $jform array, τα στοιχεία δηλαδή της φόρμας.

Το δεύτερο κομμάτι

```
$mailer =& JFactory::getMailer();
$config =& JFactory::getConfig();
$sender = array(
     $config->getValue( 'config.mailfrom' ),
     $config->getValue( 'config.fromname' )
);
$mailer->setSender($sender);
$mailer->addRecipient($config->getValue( 'config.mailfrom' ));
$days =
array("Κυριακή","Δευτέρα","Τρίτη","Τετάρτη","Πέμπτη","Παρασκευή","Σάββα
το");
$body   = "Έχετε μία αίτηση αλλαγής στοιχείων κράτησης!\n\n";
$body   .= "ID κράτησης: " . $data['id'] . "\n\n";
$body   .= "Στοιχεία Αίτησης Αλλαγής\n";
$body   .= "Όνομα: " . $data['reserver_name'] . "\n";
```

```
$body   .= "Τηλέφωνο: " . $data['reserver_contact'] . "\n";
$body   .= "Από: " . $data['from_timestamp'] . "\n";
$body   .= "Εως: " . $data['to_timestamp'] . "\n";
$body   .= "Επαναληπτική (ανά εβδομάδα): " . $data['recurse'] . "\n";
$body   .= "Ημέρα (σε περίπτωση επαναληπτικής): " .
$days[$data['days']] . "\n";
$body   .= "Από (σε περίπτωση επαναληπτικής): " . $data['recurse_from']
. ":00\n";
$body   .= "Εως (σε περίπτωση επαναληπτικής): " . $data['recurse_to'] .
":00\n";
$body   .= "Σχόλια: " . $data['reserver_comments'] . "\n";
$mailer->setSubject('Αίτηση αλλαγής κράτησης!');
$mailer->setBody($body);
$mailer->Encoding = 'base64';
$send =& $mailer->Send();
```

κατασκευάζει και στέλνει το email στο διαχειριστή. Οι εντολές

```
$mailer =& JFactory::getMailer();
$config =& JFactory::getConfig();
```

αρχικοποιούν την mail εφαρμογή και το Config Object που περιέχει χρήσιμη πληροφορία. Με τις εντολές

```
$sender = array(
     $config->getValue( 'config.mailfrom' ),
     $config->getValue( 'config.fromname' )
);
$mailer->setSender($sender);
```

θέτουμε τον sender με το mail που έχει δηλωθεί στο configuration του Joomla (ώστε το email να φτάνει από το site), ενώ με την εντολή

```
$mailer->addRecipient($config->getValue('config.mailfrom'));
```

θέτουμε τον παραλήπτη, που στη συγκεκριμένη περίπτωση είναι το mail της Διαχείρισης. Οι επόμενες εντολές συνθέτουν το σώμα του κειμένου με τις μεταβλητές της φόρμας στην μεταβλητή $body, ενώ με τις παρακάτω εντολές

```
$mailer->setSubject('Αίτηση αλλαγής κράτησης!');
$mailer->setBody($body);
$mailer->Encoding = 'base64';
```

ορίζεται το θέμα του mail, ενσωματώνεται η $body στο κείμενο του email και καθορίζεται το encoding. Τέλος, με την εντολή

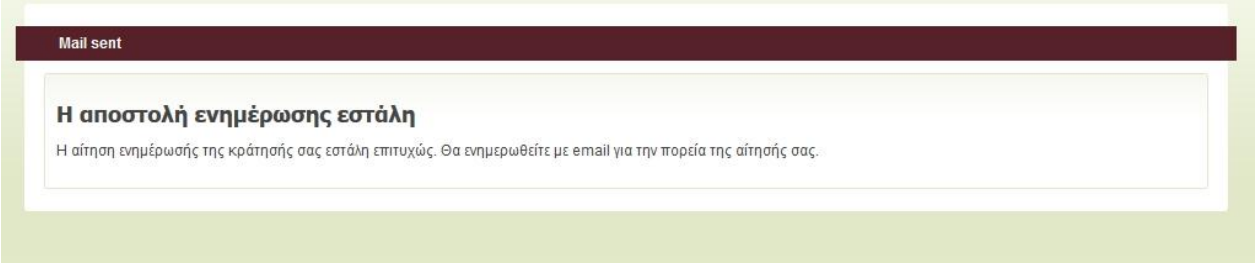
```
$send =& $mailer->Send();
```

το mail αποστέλεται.

Το τελευταίο κομμάτι αφορά την επιτυχή ή μη αποστολή του email. Όπως φαίνεται στις παρακάτω εντολές

```
	if ( $send !== true ) {
		$app->enqueueMessage('Error sending email');
		echo
"<h2>".JTEXT::_('COM_CLASSRES_RESERVATION_UPDATE_FAILED').”</h2>";
		echo
"<p>".JTEXT::_('COM_CLASSRES_RESERVATION_UPDATE_FAILED_DESC') . "</p>";
	} else {
		$app->enqueueMessage('Mail sent');
		echo
"<h2>".JTEXT::_('COM_CLASSRES_RESERVATION_UPDATE_SENT').”</h2>";
		echo
"<p>".JTEXT::_('COM_CLASSRES_RESERVATION_UPDATE_SENT_DESC')."</p>";
	}
	 return true;
```

εάν το email αποτύχει να σταλεί, περνάει το αντίστοιχο μήνυμα στην εφαρμογή και ειδοποιείται ο χρήστης. Εάν το email σταλεί με επιτυχία, επίσης ο χρήστης ειδοποιείται αναλόγως.



Εικόνα 6. Μήνυμα επιτυχούς αποστολής

3.2.1.3.3 Ακύρωση Κράτησης

Από την οθόνη διαχείρισης της αίτησης κράτησης, ο χρήστης έχει επίσης τη δυνατότητα να ακυρώσει εάν επιθυμεί την κράτησή του. Αυτό γίνεται κάνοντας κλικ στο σύνδεσμο "Cancel Reservation". Αυτό θα δρομολογήσει στην εφαρμογή τις ακόλουθες παραμέτρους:

option: com_classres
view: reservation
task: reservation.cancel
code: <reservation_code>

Έτσι, αρχικά θα εκτελεστεί η function `cancel()` του Reservation Controller. Αρχικά με τις παρακάτω εντολές

```
	$app = JFactory::getApplication();
	$code = JRequest::getVar('code');
```

φορτώνεται το Object της εφαρμογής και αποθηκεύεται η παράμετρος code. Στη συνέχεια, στο παρακάτω κομμάτι κώδικα

```
	$db = JFactory::getDbo();
	$query = "SELECT id FROM #__classres_reservations WHERE
res_code='".$code."'";
	$db->setQuery($query);
```

```
	$result = $db->query();
	$query = "UPDATE #__classres_reservations SET
status='Canceled',res_code='0' WHERE res_code='".$code."'";
	$db->setQuery($query);
```

εκτελείται ένα sql query το οποίο αλλάζει τις τιμές status και res_code της κράτησης (κατάσταση και κωδικός κράτησης αντίστοιχα) σε Canceled και 0 αντίστοιχα. Η κράτηση για την οποία αλλάζουν αυτά τα στοιχεία είναι η ίδια με το reservation code που πήραμε από την παράμετρο. Το τελευταίο κομμάτι αφορά και πάλι την ενημέρωση του χρήστη ως προς την επιτυχημένη ή αποτυχημένη ακύρωση της αίτησης κράτησής του.

```
if ( !$result ) {
	$app->enqueueMessage('Error canceling');
	echo
"<h2>".JTEXT::_('COM_CLASSRES_RESERVATION_CANCEL_FAILED')."</h2>";
	echo
"<p>".JTEXT::_('COM_CLASSRES_RESERVATION_CANCEL_FAILED_DESC')."</p>";
} else {
	echo
"<h2>".JTEXT::_('COM_CLASSRES_RESERVATION_CANCEL_SUCEEDED')."</h2>";
	echo
"<p>".JTEXT::_('COM_CLASSRES_RESERVATION_CANCEL_SUCEEDED_DESC')."</p>";
}
return true;
```

3.2.2 Λειτουργίες διαχειριστή (Admin functions)

Στην προηγούμενη ενότητα αναλύθηκε ο κώδικας του component `ClassRes`, σε ό,τι αφορά τις λειτουργίες στις οποίες έχει πρόσβαση ο απλός χρήστης της εφαρμογής, αναλύσαμε δηλαδή όλες τις λειτουργίες που εκτελούνται στο δημόσιο μέρος της ιστοσελίδας (frontend) για τον απλό χρήστη. Ωστόσο, η ουσιαστική διαχείριση και επεξεργασία της πληροφορίας των κρατήσεων και των αιθουσών γίνεται από το διαχειριστικό μέρος. Στην παρούσα ενότητα θα αναλύσουμε όλες τις πρόσθετες λειτουργίες στις οποίες έχει πρόσβαση ο διαχειριστής από το διαχειριστικό τμήμα (administration / back-end).

Η ανάλυση θα ακολουθήσει την ίδια λογική με την ανάλυση των λειτουργιών του απλού χρήστη. Αρχικά θα αναλύσουμε πως συμβαίνει στο διαχειριστικό μέρος η απλή παρουσίαση και προβολή των πληροφοριών και στη συνέχεια θα επεκταθούμε στον τρόπο που γίνεται η επεξεργασία αυτών, δηλαδή η δημιουργία, τροποποίηση και διαγραφή τους. Αυτό θα αναλυθεί τόσο για τις αίθουσες, όσο και για τις κρατήσεις.

3.2.2.1 Διαχείριση Αιθουσών

Ο διαχειριστής έχει πρόσβαση στην εφαρμογή και στις επιμέρους λειτουργίες της από τη διαχείριση, και συγκεκριμένα ακολουθώντας τη διαδρομή Components -> ClassRes.

Εικόνα 7. Διαχειριστικά μενού του component ClassRes

Το μενού ClassRes διαθέτει 3 υπομενού:

- Classrooms
- Reservations
- Archived Reservations

Πατώντας σε κάποιο από αυτά, ο διαχειριστής μεταφέρεται στην αντίστοιχη κατηγορία της εφαρμογής. Το πρώτο μενού που θα εξετάσουμε είναι το μενού Classrooms, το μενού δηλαδή που του παρουσιάζει τη λίστα των αιθουσών της εφαρμογής.

3.2.2.1.1 Προβολή λίστας αιθουσών

Ο διαχειριστής έχει πλήρη πρόσβαση στα δεδομένα που συνοδεύουν τις αίθουσες και τις κρατήσεις. Έτσι, μέσω της διαχείρισης μπορεί να προβάλει όλες τις διαθέσιμες αίθουσες ώστε να επιλέξει κάποια από αυτές για επεξεργασία. Πατώντας πάνω στο μενού "Classrooms", ο διαχειριστής οδηγείται στο link

`/`**`administrator`**`/index.php?`**`option`**`=com_classres&`**`view`**`=classrooms,`

με παραμέτρους

directory: administrator
option: com_classres
view: classrooms

Η διαφορά με τις αντίστοιχες λειτουργίες στο frontend είναι ότι τώρα οι κλήσεις γίνονται από τη Διαχείριση (**administrator**), επομένως χρησιμοποιούνται τα αντίστοιχα αρχεία που βρίσκονται κάτω από το directory **administrator** στο file structure του Joomla!. Έχοντας λάβει το παραπάνω υπόψιν, επισκεπτόμενοι το παραπάνω url, θα κληθεί το αρχείο classres.php που βρίσκεται κάτω από το directory administrator/components/com_classres, και όχι components/com_classres όπως γίνεται με τις αντίστοιχες κλήσεις στο frontend. Ωστόσο, η δομή αυτού του αρχείου είναι ίδια με την αντίστοιχη του frontend, και αυτό που κάνει είναι να αρχικοποιεί τον Controller ClassRes και να ελέγχει την παράμετρο view, θέτοντας `view = classrooms` εάν δεν υπάρχει κάποια τιμή.

Στην παρούσα περίπτωση υπάρχει αν και τυχαίνει να είναι ίδια με την default τιμή οπότε, από τη στιγμή που δεν έχει τιμή η παράμετρος **task,** καλείται η function `display()` της View **ClassResViewClassrooms**. Ο κώδικας της function είναι ο παρακάτω:

```
JToolBarHelper::title(
     JText::_('COM_CLASSRES_CLASSROOMS'),
     'generic.png'
);
JToolBarHelper::addNewX();
```

```
JToolBarHelper::editListX();
JToolBarHelper::deleteList();
$items      = & $this->get( 'Data');
$this->assignRef('items', $items);
$state = & $this->get('State');
$this->assignRef('state', $state);
parent::display($tpl);
```

Οι τέσσερεις πρώτες εντολές δημιουργούν σύμφωνα με το layout της Διαχείρισης του Joomla! ένα γενικό τίτλο για τη σελίδα (συνοδευόμενο από κάποιο εικονίδιο) και κάποια κουμπιά επεξεργασίας, στη συγκεκριμένη περίπτωση το κουμπί Προσθήκης (New), Επεξεργασίας (Edit) και Διαγραφής (Delete). Τις επόμενες εντολές τις εξετάσαμε στα προηγούμενα κεφάλαια. Η

```
$items      = & $this->get( 'Data');
```

φορτώνει τη λίστα αιθουσών χρησιμοποιώντας την in-built function του Joomla! στην αντίστοιχη μεταβλητή, η

```
$this->state = $this->get('State');
```

φορτώνει πληροφορία για τα properties του Model Classrooms και η

```
parent::display($tpl);
```

χρησιμοποιεί το HTML template (που τώρα είναι το **default**) για να δομήσει και να παρουσιάσει την πληροφορία. Στη συνέχεια καλείται το template default.php που βρίσκεται στο directory administrator/components/com_classres/views/classrooms/tmpl/default.php. Το τελικό αποτέλεσμα φαίνεται στην παρακάτω εικόνα:

Εικόνα 8. Προβολή αιθουσών στη διαχείριση

3.2.2.1.2 Διαγραφή αίθουσας

Για τη διαγραφή κάποιας αίθουσας, η διαδικασία είναι απλή. Ο Διαχειριστής επιλέγει την αίθουσα που θέλει να διαγράψει ενεργοποιώντας το αντίστοιχο checkbox στην πρώτη στήλη της σειράς της αίθουσας. Ο Διαχειριστής μπορεί να επιλέξει με τον τρόπο αυτό και παραπάνω από μία αίθουσες, εάν επιθυμεί να διαγράψει περισσότερες.

Αφού επιλέξει τις αίθουσες που επιθυμεί, τότε πατάει το κουμπί Delete που εμφανίζεται στην πάνω δεξιά γωνία της Διαχείρισης. Το Joomla! χρησιμοποιεί εσωτερικές λειτουργίες που εκτελούν την ενέργεια της διαγραφής, χωρίς να χρειαστεί να υλοποιηθεί κάτι εκ νέου στο component.

3.2.2.1.3 Δημιουργία αίθουσας

Για τη δημιουργία μιας νέας αίθουσας, ο Διαχειριστής κάνει κλικ στο κουμπί New, στο μενού που βρίσκεται πάνω δεξιά στη Διαχείριση. Κάνοντας κλικ στο κουμπί, οδηγείται στο παρακάτω link

`/administrator/index.php?option=com_classres&view=classroom&layout=edit`

Έτσι, έχουμε τις παραμέτρους

directory: administrator
option: com_classres
view: classroom
layout: edit

Αυτό σημαίνει πως οι εσωτερικές διαδικασίες θα κινηθούν, μέσω του directory **administrator**, στο component **com_classres** καλωντας τον κεντρικό controller, ο οποίος με τη σειρά του θα καλέσει την View **Classroom** που θα εκτελέσει την `display()`, ο κώδικας της οποίας παρουσιάζεται και αναλύεται παρακάτω:

```
$item = $this->get( 'Data' );
$isNew = ($item->id < 1);
JRequest::setVar('hidemainmenu', true);
if ($isNew) {
      JToolBarHelper::title(
            JText::_('COM_CLASSRES_CLASSROOMS_NEW'),
            'generic.png'
      );
} else {
      JToolBarHelper::title(
            JText::_('COM_CLASSRES_CLASSROOMS_EDIT'),
            'generic.png'
      );
}
JToolBarHelper::save();
if ($isNew)  {
      JToolBarHelper::cancel();
} else {
      // for existing items the button is renamed `close`
      JToolBarHelper::cancel( 'cancel', 'Close' );
}
$this->assignRef('item', $item);
parent::display($tpl);
$document = JFactory::getDocument();
$document->addScript(
JURI::root().'/administrator/components/com_classres/models/forms
/classroom.js');
      $document->addScript(
JURI::root().'/administrator/components/com_classres/models/forms/valid
ation.js');
      JText::script('COM_CLASSRES_VALIDATION_ERROR');
```

Οι πρώτες δύο γραμμές του κώδικα είναι γνωστές από προηγούμενες αναλύσεις. Με την εντολή

```
$isNew = ($item->id < 1);
```

θέτουμε την boolean μεταβλητή $isNew ανάλογα με το εάν υπάρχει ήδη η αίθουσα (σε περίπτωση επεξεργασίας που θα αναλυθεί στη συνέχεια) ή εάν είναι καινούρια εγγραφή.
Με την εντολή

```
JRequest::setVar('hidemainmenu', true);
```

κρύβουμε το κεντρικό μενού του Joomla!, και στη συνέχεια, ελέγχοντας τη μεταβλητή $isNew, παρουσιάζουμε τον αντίστοιχο τίτλο στη σελίδα:

```
if ($isNew) {
      JToolBarHelper::title(
            JText::_('COM_CLASSRES_CLASSROOMS_NEW'),
            'generic.png'
      );
} else {
      JToolBarHelper::title(
            JText::_('COM_CLASSRES_CLASSROOMS_EDIT'),
            'generic.png'
      );
}
```

Αντίστοιχα και με τις επόμενες εντολές

```
JToolBarHelper::save();
if ($isNew)  {
      JToolBarHelper::cancel();
} else {
      // for existing items the button is renamed `close`
      JToolBarHelper::cancel( 'cancel', 'Close' );
}
```

που προσθέτουν στην μπάρα πάνω δεξιά τα αντίστοιχα κουμπιά που ορίζουμε. Τέλος, με τις εντολές

```
$document = JFactory::getDocument();
$document->addScript(
JURI::root().'/administrator/components/com_classres/models/forms
/classroom.js');
$document->addScript(
JURI::root().'/administrator/components/com_classres/models/forms/valid
ation.js');
JText::script('COM_CLASSRES_VALIDATION_ERROR');
```

προσθέτουμε στη σελίδα ($document) javascript που θα ελέγχει τη φόρμα κατά την αποθήκευσή της. Στη συνέχεια καλείται το layout **edit**, το οποίο παρουσιάζει τη φόρμα της αίθουσας όπως φαίνεται στην παρακάτω εικόνα:

Εικόνα 9. Προσθήκη νέας αίθουσας

Στη συνέχεια και μετά τη συμπλήρωση των στοιχείων, ο Διαχειριστής έχει τρεις επιλογές, όπως αυτές ορίστηκαν προηγουμένως: μπορεί να αποθηκεύσει την εγγραφή και να παραμείνει στην ίδια σελίδα (Save), να αποθηκεύσει την εγγραφή και να βγει από τη σελίδα (Save & Close) ή να ακυρώσει την οποιαδήποτε ενέργεια (Cancel). Στην τελευταία περίπτωση, απλά οδηγείται στη σελίδα της λίστας αιθουσών χωρίς να αποθηκευτούν οι πληροφορίες και η εγγραφή.

Στις υπόλοιπες δύο περιπτώσεις, εκτελούνται αρχικά οι έλεγχοι της javascript και, εάν όλα είναι σωστά, η αίθουσα αποθηκεύεται χρησιμοποιώντας και πάλι τις εσωτερικές διαδικασίες του Joomla!. Για τον λόγο αυτό δε θα γίνει ανάλυση των συγκεκριμένων ενεργειών στο παρόν κείμενο.

3.2.2.1.4 Προβολή στοιχείων αίθουσας

Η προβολή των στοιχείων κάποιας αίθουσας γίνεται κάνοντας κλικ σε κάποια από τις αίθουσες στη λίστα αιθουσών. Ο Διαχειριστής τότε οδηγείται στον ίδιο σύνδεσμο όπως προηγουμένως, που ακολουθεί

```
/administrator/index.php?option=com_classres&view=classroom&layout=edit&id=2
```

με τη διαφορά πως αυτή τη φορά υπάρχει και το id της αίθουσας στο link. Έτσι, στον έλεγχο της συνθήκης $isNew (βλ. 3.2.2.1.3 Δημιουργία αίθουσας), η μεταβλητή παίρνει την τιμή false. Από τη στιγμή που υπάρχει id, η φόρμα της αίθουσας συμπληρώνεται με τα αντίστοιχα στοιχεία της αίθουσας

στην οποία αντιστοιχίζεται το id από τη βάση δεδομένων. Έτσι, ο Διαχειριστής, κάνοντας κλικ στο όνομα μιας αίθουσας, μπορεί να δει τα στοιχεία της και ενδεχομένως να τα τροποποιήσει.

3.2.2.1.5 Επεξεργασία στοιχείων αίθουσας

Η αλλαγή των στοιχείων της αίθουσας γίνεται σε ακολουθία της προβολής των στοιχείων της και ακολουθεί την ίδια διαδικασία με τη δημιουργία νέας αίθουσας. Για μια ακόμα φορά, οι λειτουργίες εμπεριέχονται στις εσωτερικές διεργασίες του Joomla!, οπότε αρκεί να πατήσει ο Διαχειριστής σε κάποιο από τα πλήκτρα αποθήκευσης (Save, Save & Close, Save & New) για να εκτελεστούν οι απαραίτητοι έλεγχοι της javascript και, εάν όλα είναι ορθά, να γίνουν οι αντίστοιχες ενημερώσεις.

3.2.2.2 Διαχείριση Αιτήσεων Κρατήσεων

Το δεύτερο μενού είναι το μενού ClassRes - Reservations, το μενού δηλαδή που του παρουσιάζει τη λίστα των αιτήσεων κρατήσεων της εφαρμογής και που θα αναλύσουμε στη συνέχεια.

3.2.2.2.1 Προβολή Λίστας Αιτήσεων Κρατήσεων

Ο Διαχειριστής έχει πλήρη πρόσβαση στις αιτήσεις κρατήσεων που αποστέλλονται ή είναι αποθηκευμένες στο σύστημα. Μπορεί μέσω της εφαρμογής να ελέγξει τα στοιχεία κάποιας κράτησης, να τα αλλάξει, να διαγράψει κρατήσεις κτλ. Η προβολή της λίστας Αιτήσεων Κρατήσεων γίνεται κάνοντας κλικ στο παραπάνω μενού. Ο Διαχειριστής τότε οδηγείται στο link

administrator/index.php?**option**=com_classres&**view**=reservations

Έτσι, σύμφωνα με όσα έχουν εξηγηθεί παραπάνω, καλείται η function `display()` της View **ClassResViewReservations**. Ο κώδικας της function είναι ο παρακάτω:

```
$this->get('ReservationsCheck');
JToolBarHelper::title(
     JText::_( 'COM_CLASSRES_RESERVATIONS' ),
     'generic.png'
);
JToolBarHelper::addNewX('reservation.add');
JToolBarHelper::editListX('reservation.edit');
JToolBarHelper::deleteList();
$items = & $this->get( 'Data');
$this->assignRef('items', $items);
$state = & $this->get('State');
$this->assignRef('state', $state);
parent::display($tpl);
```

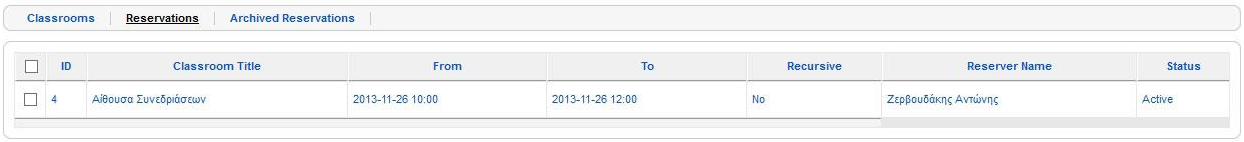
Ο παραπάνω κώδικας δεν έχει τίποτα καινούριο σε σχέση με αυτά που έχουν αναλυθεί προηγουμένως, εκτός από την εντολή

```
$this->get('ReservationsCheck');
```

Η εντολή αυτή καλεί την function `getReservationsCheck()` από το model Reservations, με στόχο να ελεγχθούν οι κρατήσεις και να μεταφερθούν στις Archived (αρχείο) όσες έχουν περάσει ημερολογιακά και δεν ισχύουν πλέον. Ο κώδικας της getReservationsCheck() είναι ο παρακάτω:

```
	$db = JFactory::getDBO();
	$query = " UPDATE ".$db->nameQuote('#__classres_reservations')."
SET archived = 1 WHERE to_timestamp < NOW() AND recurse = 0";
	$db->setQuery($query);
	$db->query();
```

Το παραπάνω είναι ένα απλό query στη βάση δεδομένων το οποίο θέτει την παράμετρο archived = 1 όπου το to_timestamp, δηλαδή το τέλος της κράτησης, είναι προγενέστερο της τωρινής ημερομηνίας και ώρας. Στη συνέχεια παρουσιάζονται όλες οι αιτήσεις για τις οποίες ισχύει archived = 0, όλες οι αιτήσεις δηλαδή που αφορούν μελλοντικές ημερομηνίες. Το τελικό αποτέλεσμα που φαίνεται στον Διαχειριστή είναι το παρακάτω:



Εικόνα 10. Προβολή λίστας κρατήσεων

3.2.2.2.2 Διαγραφή Αίτησης Κράτησης

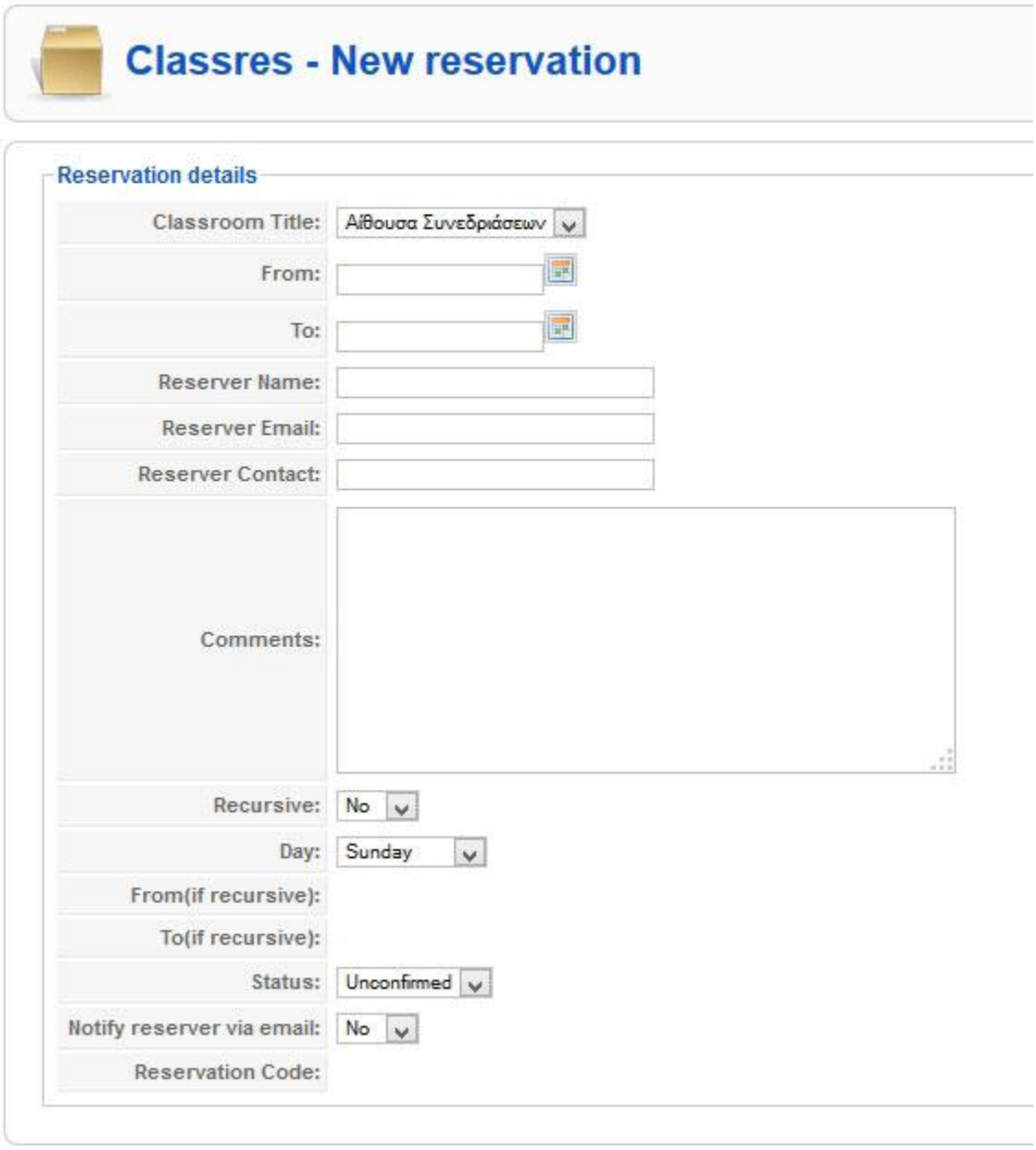
Η διαγραφή μίας ή περισσότερων αιτήσεων κρατήσεων γίνεται με τον ίδιο ακριβώς τρόπο όπως με τη διαγραφή των αιθουσών, χρησιμοποιώντας και πάλι εσωτερικές λειτουργίες του Joomla!

3.2.2.2.3 Δημιουργία Αίτησης Κράτησης

Ο Διαχειριστής, πέραν της προβολής, επεξεργασίας ή/και διαγραφής των αιτήσεων, έχει επίσης τη δυνατότητα δημιουργίας μιας αίτησης κράτησης, σε περίπτωση που δε γίνει μέσω του site αλλά με κάποιον άλλον τρόπο, παραδείγματος χάριν τηλεφωνικά. Η διαδικασία είναι και πάλι παρόμοια με τη διαδικασία δημιουργίας μιας νέας αίθουσας: ο διαχειριστής κάνει κλικ στο κουμπί New και μεταφέρεται στο link

`/administrator/index.php?option=com_classres&view=reservation&layout=edit`

όπου αυτή τη φορά θα εκτελεστεί η function `display()` της View Reservation, η οποία όμως έχει πανομοιότυπο κώδικα με την αντίστοιχη function της View Classroom που αναλύθηκε προηγουμένως. Έτσι, χρησιμοποιώντας στη συνέχεια το template administrator/components/com_classres/views/reservation/tmpl/edit.php, εμφανίζεται στο Διαχειριστή η παρακάτω οθόνη:



Εικόνα 5. Προσθήκη νέας κράτησης

Από εδώ ο Διαχειριστής μπορεί να συμπληρώσει όλα τα στοιχεία μιας νέας κράτησης και στη συνέχεια να την αποθηκεύσει με παρόμοιες διαδικασίες με εκείνες της αποθήκευσης μιας νέας αίθουσας και μετά τον αντίστοιχο έλεγχο της ορθότητας των πεδίων από την javascript.

3.2.2.2.4 Προβολή Στοιχείων Αίτησης Κράτησης

Για να προβάλει ο Διαχειριστής τα στοιχεία μιας αίτησης κράτησης, αρκεί να κάνει κλικ στο όνομα της κράτησης. Μετά το κλικ μεταφέρεται στο link

administrator/index.php?**option**=com_classres&**view**=reservation&**layout**=edit&**id**=2

Όπως αναλύθηκε και στην προβολή στοιχείων μιας αίθουσας, από τη στιγμή που υπάρχει id στο url, η φόρμα θα γεμίσει με τα στοιχεία της αντίστοιχης αίτησης.

3.2.2.2.5 Επεξεργασία Στοιχείων Αίτησης Κράτησης

Ο Διαχειριστής μπορεί στη συνέχεια να αλλάξει όσα πεδία επιθυμεί και στη συνέχεια να την αποθηκεύσει με τον ίδιο τρόπο που αποθήκευσε μια νέα αίτηση κράτησης. Οι διαδικασίες είναι και πάλι μέρος των κεντρικού κώδικα του Joomla! και παρόμοιες με τις διαδικασίες ενημέρωσης μίας υπάρχουσας αίθουσας.

4. Έλεγχος καλής λειτουργίας – Συμβατότητα

Πέρα από την κύρια χρήση του component είναι σημαντικό να εξετάσουμε όλες τις πιθανές περιπτώσεις χρήσης του ώστε να εξαντλήσουμε όλες τις πιθανότητες λαθών. Αυτό είναι πολύ σημαντικό ώστε η εφαρμογή να λειτουργεί σωστά, οτιδήποτε και αν κάνει ο χρήστης. Όπως είδαμε στο προηγούμενο κεφάλαιο, ο χρήστης έχει τις παρακάτω δυνατότητες, είτε σαν διαχειριστής (admin) είτε σαν απλός χρήστης (user) :

Απλές

- Προβολή πληροφοριών
  - Προβολή λίστας αιθουσών
  - Προβολή στοιχείων συγκεκριμένης αίθουσας
- Δημιουργία αίτησης κράτησης
  - Αποστολή αίτησης
  - Επιβεβαίωση αίτησης
- Επεξεργασία αίτησης κράτησης
  - Προβολή αίτησης κράτησης
  - Αίτηση αλλαγής στοιχείων κράτησης
  - Ακύρωση κράτησης

Διαχειριστικές

- Προβολή πληροφοριών
  - Προβολή λίστας αιθουσών
  - Προβολή λίστας αιτήσεων κρατήσεων
  - Προβολή λίστας αιτήσεων κρατήσεων αρχείου
- Δημιουργία νέων στοιχείων
  - Δημιουργία νέας αίθουσας
  - Δημιουργία νέας κράτησης
- Επεξεργασία πληροφοριών
  - Επεξεργασία στοιχείων αίθουσας
  - Επεξεργασία στοιχείων αίτησης κράτησης
  - Επεξεργασία στοιχείων αίτησης κράτησης αρχείου
- Διαγραφή πληροφοριών
  - Διαγραφή αίθουσας
  - Διαγραφή αίτησης κράτησης

Παρακάτω θα εξετάσουμε για όλες τις παραπάνω ενέργειες όλες τις πιθανές υποπεριπτώσεις, εάν υπάρχουν, καθώς και το πώς αυτές αντιμετωπίστηκαν είτε χρησιμοποιώντας τις παρεχόμενες βιβλιοθήκες του Joomla, είτε μέσω εφαρμογής custom κώδικα.

4.1 Λειτουργίες χρήστη

4.1.1 Προβολή πληροφοριών

Για την προβολή πληροφοριών, είτε για τη λίστα αιθουσών, είτε για συγκεκριμένη αίθουσα, δεν χρειάστηκε να εκτελέσουμε κάποια επιπλέον λειτουργία για να καλύψουμε τις πιθανότητες εκτέλεσης κακόβουλων ενεργειών. Ειδικά στην περίπτωση της προβολής συγκεκριμένης αίθουσας, σε περίπτωση

που κάποιος χρήστης εισάγει χειροκίνητα επεξεργάζοντας το url κάποιο ID που δεν υπάρχει στη βάση δεδομένων, το Joomla επιστρέφει σχετικό σφάλμα που τον ενημερώνει σχετικά.

4.1.2 Δημιουργία αίτησης κράτησης

Η δημιουργία και αποστολή μιας νέας αίτησης κράτησης γίνεται από την σελίδα της προβολής αίθουσας. Ο χρήστης μπορεί να αποστείλλει μέσω της φόρμας κράτησης μια νέα κράτηση, ενώ θα πρέπει στη συνέχεια να την επιβεβαιώσει από το email του.

4.1.2.1 Αποστολή αίτησης

Η μοναδική ενέργεια που μπορεί να εκτελέσει ο απλός χρήστης, πέραν της προβολής των αιθουσών και των κρατήσεων που τις συνοδεύουν, είναι να αποστείλλει μια αίτηση κράτησης. Ο χρήστης καλείται να συμπληρώσει κάποια στοιχεία ώστε η υποβολή της αίτησης να δημιουργήσει μια κράτηση για τη συγκεκριμένη αίθουσα με κατάσταση Pending. Στη συνέχεια κάποιος διαχειριστής θα αναλάβει να εξετάσει τα στοιχεία και να επιβεβαιώσει ή να απορρίψει την κράτηση.

Για το σκοπό αυτό ο χρήστης συμπληρώνει και αποστέλλει μια φόρμα. Η φόρμα αυτή περιέχει πεδία σχετικά με το όνομα του χρήστη, ένα έγκυρο email του, έναν αριθμό τηλεφώνου καθώς και την περίοδο για την οποία επιθυμεί την κράτηση. Προαιρετικά μπορεί να συμπληρώσει και κάποια σχόλια.

Επιπλέον, υπάρχει η δυνατότητα ο χρήστης να επιλέξει η κράτηση να λειτουργεί κάθε εβδομάδα την ίδια ημέρα και ώρα. Για το σκοπό αυτό μπορεί να ελέγξει την επιλογή Recursive, οπότε και εμφανίζονται άλλα 3 πεδία: ημέρα κράτησης και ώρα (από – έως).

Όλα τα πεδία εκτός από το πεδίο των σχολίων και της αναδρομικής κράτησης είναι υποχρεωτικά στη συμπλήρωση. Το email πρέπει να είναι της μορφής "xxx@yyy.zzz", ενώ τα πεδία " από " και " έως " πρέπει να έχουν τη μορφή ημερομηνίας-ώρας, πχ "13-02-2013 18:00". Σε περίπτωση που ο χρήστης επιλέξει αναδρομική κράτηση, τότε τα πεδία από / έως δε λαμβάνονται υπόψιν (απενεργοποιούνται) και ο χρήστης καλείται να επιλέξει ημέρα και ώρα, αντί για ημερομηνία.

Εάν τα πεδία είναι σωστά, τότε η φόρμα αποστέλλεται με επιτυχία και ο χρήστης-αιτών της κράτησης ενημερώνεται στο email που έδωσε για την κατάσταση της αίτησής του. Οι έλεγχοι γίνονται με τη χρήση javascript, η οποία επιβεβαιώνει ότι τα εισαγόμενα στοιχεία πληρούν τις προϋποθέσεις εισαγωγής της κράτησης και βρίσκονται σε σωστή μορφή.

4.1.2.2 Επιβεβαίωση αίτησης

Μετά την αποστολή της κράτησης, ο χρήστης καλείται να επιβεβαιώσει το email του κάνοντας κλικ σε έναν σύνδεσμο που αποστέλλεται στο email που δήλωσε. Με τον τρόπο αυτό υπάρχει μια ταυτοποίηση του email με την κράτηση και αποφεύγεται η περίπτωση δήλωσης ψευδούς ή ανενεργού email. Επίσης, με την ενέργεια αυτή αποτρέπεται η πιθανότητα να δήλωσε κάποιος διαφορετικό email από το δικό του. Η επιβεβαίωση γίνεται κάνοντας κλικ σε ένα σύνδεσμο της μορφής

```
/index.php?option=com_classres&view=reservation&layout=validate&code=<code>,
```

όπου <code> είναι μία τυχαία παραγόμενη αλφαριθμητική συμβολοσειρά 16 χαρακτήρων. Με τον τρόπο αυτό αποφεύγεται η πιθανότητα κάποιος να αποκωδικοποιήσει τον τρόπο που παράγεται ο κωδικός αυτός για να επιβεβαιώσει την κράτησή του χρησιμοποιώντας τον παραπάνω σύνδεσμο χωρίς να χρειάζεται να έχει πρόσβαση στο email που δήλωσε, καθώς η παραγωγή της συμβολοσειράς είναι τυχαία. Αυτό επιβεβαιώνει ότι το email που ορίστηκε όντως ανήκει στο άτομο που έκανε την κράτηση.

4.1.3 Επεξεργασία αίτησης κράτησης

4.1.3.1 Προβολή αίτησης κράτησης

Με την επιβεβαίωση από τους διαχειριστές της κράτησης, ο χρήστης που την έκανε λαμβάνει έναν κωδικό με τον οποίο μπορεί να ελέγξει τα στοιχεία της κράτησής του. Ο κωδικός αυτός παράγεται επίσης με τυχαίο τρόπο, ώστε να είναι μοναδικός για την κάθε κράτηση, αλλά και απίθανο να βρεθεί χρησιμοποιώντας οποιοδήποτε ευριστικό μηχανισμό.

4.1.3.2 Αίτηση αλλαγών στοιχείων κράτησης

Αφού ο χρήστης εισάγει τον κωδικό και δει τα στοιχεία της κράτησής του, είναι δυνατόν να αποστείλει ένα αίτημα αλλαγής κάποιων στοιχείων της κράτησής του. Εδώ ισχύουν οι ίδιοι μηχανισμοί που ισχύουν και κατά την αποστολή της αρχικής αίτησης κράτησης. Ο χρήστης δεν μπορεί με τον τρόπο αυτό να αλλάξει τα στοιχεία της κράτησής του άμεσα, παρά μόνο να στείλει μέσω της φόρμας ένα σχετικό email στους διαχειριστές. Σε διαφορετική περίπτωση, ο χρήστης θα μπορούσε, αφού η κράτησή του θεωρείται επιβεβαιωμένη και ενεργή, να αλλάξει τα στοιχεία σε άλλες ώρες ή/και ημέρες, δημιουργώντας σύγχυση με τις άλλες ενεργές κρατήσεις. Είναι λοιπόν στη δικαιοδοσία και την κρίση των διαχειριστών εάν θα προβούν στις αλλαγές που αιτήθηκε ο χρήστης.

4.1.3.3 Ακύρωση αίτησης κράτησης

Ο χρήστης, επιλέγοντας το σχετικό link, μπορεί να ακυρώσει την κράτησή του. Και πάλι, με τη χρήση του κωδικού κράτησης στο link, αποφεύγεται η ακύρωση της κράτησης από κάποιον με κακόβουλο σκοπό.

4.2 Λειτουργίες Διαχειριστή

Οι παρακάτω λειτουργίες αφορούν τις λειτουργίες που μπορεί να εκτελέσει ο διαχειριστής της ιστοσελίδας, χρησιμοποιώντας το διαχειριστικό μέρος (administration). Η πρόσβαση στο διαχειριστικό μέρος γίνεται με την εισαγωγή του ονόματος χρήστη και του κωδικού του Διαχειριστή, κάτι που φυσικά λειτουργεί σαν δικλείδα ασφαλείας.

4.2.1 Προβολή πληροφοριών

Για την απλή προβολή των πληροφοριών, είτε αιθουσών, είτε κρατήσεων, δε χρειάζεται κάποια ασφάλεια, καθώς ο διαχειριστής έχει πρόσβαση σε κάθε πληροφορία.

4.2.2 Δημιουργία νέων στοιχείων

4.2.2.1 Δημιουργία νέας αίθουσας

Για τη δημιουργία μίας νέας αίθουσας, είναι απαραίτητες οι πληροφορίες για τον Τίτλο (ονομασία) της και τη Χωρητικότητά της. Εάν τα στοιχεία αυτά δε δοθούν κατά τη δημιουργία, θα πρέπει να μην μπορεί να προχωρήσει η διαδικασία. Αυτό επιτεύχθηκε με τη χρήση Javascript στον αντίστοιχο τομέα του διαχειριστικού μέρους, ώστε να ενημερώνεται ο διαχειριστής ότι πρέπει να δώσει τις παραπάνω πληροφορίες, σε περίπτωση που δεν τις έχει συμπληρώσει, πριν επιλέξει να αποθηκεύσει τη νέα αίθουσα.

Καθώς η εφαρμογή χρησιμοποιεί και φωτογραφίες, θα μπορούσε να γίνει το ίδιο και για αυτές. Ωστόσο, όσον αφορά τις φωτογραφίες προτιμήθηκε να υπάρχει εξ' ορισμού μία εικόνα-δείγμα, σε περίπτωση που δεν υπάρχει φωτογραφία για την αίθουσα. Έτσι, το πεδίο της φωτογραφίας δεν είναι μεν υποχρεωτικό, ωστόσο εάν δε δωθεί κάποια φωτογραφία χρησιμοποιείται η default εικόνα που έχει οριστεί μέχρι να οριστεί κάποια.

Κάτι άλλο που έπρεπε να προσεχτεί είναι το είδος πληροφορίας στο πεδίο Χωρητικότητας, καθώς μας ενδιαφέρει να υπάρχει ένας αριθμός. Έτσι, το πεδίο είναι αριθμητικό και δε δέχεται άλλους χαρακτήρες εκτός από αριθμούς.

4.2.2.2 Δημιουργία νέας κράτησης

Κατά παρόμοιο τρόπο αντίστοιχοι έλεγχοι συμβαίνουν κατά τη δημιουργία κρατήσεων. Τα απαιτούμενα πεδία είναι ο τίτλος της αίθουσας, η περίοδος κράτησης, το όνομα του αιτούντα την κράτησης, το email επικοινωνίας του, κάποιος τρόπος επικοινωνίας (πχ τηλέφωνο) καθώς και η κατάσταση της κράτησης.

Κανένα από τα παραπάνω πεδία δε μπορεί να είναι κενό. Για τα πεδία του τίτλου της αίθουσας και της κατάστασης κράτησης δεν μπορεί και πρακτικά να είναι κενά λόγω χρήσης dropdown select. Τέλος, ισχύουν οι παρακάτω επιμέρους έλεγχοι:

- Τα πεδία "Από" και "Έως" παίρνουν μόνο τύπους ημερομηνίας (π.χ. "13-02-2013 18:00")
- Το πεδίο email πρέπει να είναι της μορφής "xxx@yyy.zzz"

Τα παραπάνω και πάλι επιτυγχάνονται με τη χρήση Javascript κώδικα.

4.2.3 Επεξεργασία πληροφοριών

4.2.3.1 Επεξεργασία στοιχείων αίθουσας

Οι έλεγχοι κατά την επεξεργασία μιας αίθουσας παραμένουν οι ίδιοι που υφίστανται και κατά τη δημιουργία. Εάν κάποιο από τα απαιτούμενα πεδία μείνει κενό ή δεν περιέχει το σωστό τύπο πληροφορίας, τότε πριν την αποθήκευση βγαίνει σχετικό μήνυμα και η εκτέλεση της αποθήκευσης διακόπτεται μέχρι να συμπληρωθούν σωστά τα σχετικά πεδία.

4.2.3.2 Επεξεργασία στοιχείων αίτησης κράτησης

Κατά την επεξεργασία μιας υφιστάμενης κράτησης ισχύουν οι ίδιοι αντίστοιχοι έλεγχοι. Ο διαχειριστής έχει την επιλογή να ενημερώσει ή όχι στο email της κράτησης για την αλλαγή των στοιχείων της κράτησης, π.χ. σε περίπτωση αλλαγής της κατάστασης της κράτησης από Pending σε Confrimed. Εδώ η default επιλογή είναι να μην ενημερώνεται, για να αποφεύγεται πλεοναστική ενημέρωση σε περίπτωση απλής διόρθωσης στοιχείων, παραδείγματος χάριν τυπογραφικών λαθών.

4.2.3.3 Επεξεργασία στοιχείων αίτησης κράτησης αρχείου

Εδώ ισχύουν ακριβώς οι ίδιοι έλεγχοι με την επεξεργασία των στοιχείων των υπολοίπων αιτήσεων κρατήσεων.

4.2.4 Διαγραφή πληροφοριών

4.2.4.1 Διαγραφή αιθουσών

Επειδή η διαγραφή μιας εγγραφής θα διαγράψει την πληροφορία αυτή μόνιμα από τη βάση δεδομένων, πριν την εκτέλεση της διαγραφής βγαίνει ένα σχετικό μήνυμα επιβεβαίωσης της ενέργειας, ώστε να μην εκτελεστεί διαγραφή σε περίπτωση λανθασμένης επιλογής ή κεκτημένης ταχύτητας.

4.2.4.2 Διαγραφή κρατήσεων

Ομοίως, κατά τη διαγραφή των κρατήσεων εμφανίζεται και εδώ σχετικό μήνυμα επιβεβαίωσης ώστε να αποφευχθεί οποιαδήποτε λάθος διαγραφή πληροφοριών που δε θα μπορούν στη συνέχεια να επανακτηθούν.

4.3 Συμβατότητα

Το παρόν component, όσον αφορά τα όποια ζητήματα συμβατότητας, ελέγχθηκε ως προς τρεις κύριους τομείς:

1. Συμβατότητα με διαφορετικές εκδόσεις Joomla!
2. Συμβατότητα με διαφορετικά CMS
3. Συμβατότητα με browsers

Επίσης, στα πλαίσια του ελέγχου, έγινε έλεγχος καλής λειτουργίας του component για όλες τις περιπτώσεις, καθώς και για περιπτώσεις κακόβουλης επίθεσης από χρήστες ή προγράμματα.

4.3.1 Συμβατότητα με διαφορετικές εκδόσεις Joomla!

Το παρόν component αναπτύχθηκε για την έκδοση Joomla! 1.5. Καθώς το Joomla! είναι μια ζωντανή κοινότητα και συνεχώς εξελίσσσεται, πιθανές μετατροπές στον κώδικα του σε μελλοντικές εκδόσεις ίσως να καταστήσουν αδύνατη τη χρήση του συγκεκριμένου component από μια μελλοντική έκδοση. Αυτό είναι κάτι αποδεκτό στα πλαίσια της εξέλιξης του κώδικα, και για κάθε μελλοντική έκδοση Joomla! θα πρέπει να δοκιμάζονται όλα τα extensions από τους κατασκευαστές τους και να γίνονται οι απαραίτητες ρυθμίσεις, εάν χρειάζεται, ώστε να κατασκευαστούν οι αντίστοιχες εκδόσεις των components.

4.3.2 Συμβατότητα με διαφορετικά CMS

Καθώς το συγκεκριμένο component αποτελεί extension του Joomla! και χρησιμοποιεί συμβάσεις (conventions) που η αρχιτεκτονική του Joomla! έχει ορίσει, είναι αδύνατον να εγκατασταθεί σε διαφορετικά Συστήματα Διαχείρισης Περιεχομένου (CMS), όπως παραδείγματος χάριν σε Drupal ή Wordpress.

Κάτι τέτοιο θα προϋπέθετε την ανακατασκευή των τμημάτων που το αποτελούν ώστε να είναι συμβατά για το CMS που μας ενδιαφέρει. Όμως, σε περίπτωση μιας τέτοιας ενέργειας, το component θα ήταν τότε extension εκείνου του CMS, και δε θα μπορούσε με την ίδια λογική να εγκατασταθεί σε κάποιο Joomla!

4.3.3 Συμβατότητα με browsers

Ένας άλλος τομέας στον οποίο έπρεπε να δωθεί προσοχή είναι η συμβατότητα του component με τους διάφορους browsers που κυκλοφορούν, ώστε η εμπειρία του χρήστη να είναι η ίδια, άσχετα με τον browser τον οποίο χρησιμοποιει για την περιήγησή του, ενώ επίσης χρειαζόταν να μην υπάρχει κανένα

σφάλμα λειτουργίας που να επηρεάζεται από τον browser (όπως, για παράδειγμα, ο τρόπος που αναγνωρίζει την Javascript ο Internet Explorer σε σχέση με άλλους browsers).

Διαφορές υπάρχουν και ως προς τις εκδόσεις των διαφόρων browser, παραδείγματος χάριν υπάρχει πολύ μεγάλη διαφορά στον τρόπο που αντιλαμβάνεται μια σελίδα ο IE7 και ο IE9. Όλα αυτά χρειάζονται προσαρμογές και δοκιμές ώστε να αποθευχθούν πιθανές διαφορές τόσο στην εμφάνιση όσο και στη χρήση της εφαρμογής.

Λαμβάνοντας υπ' όψιν τα σταστιστικά χρήσης των browsers[1], εξείχθηκαν οι παρακάτω παρατηρήσεις[8]:

- Ο **Internet Explorer**, αργά αλλά σταθερά, περιορίζεται σημαντικά συγκριτικά με τους υπόλοιπους browsers, έχοντας πέσει κατά 4,4% από τον Οκτώβρη του 2012[2] μέχρι τον Οκτώβρη του 2013 με ποσοστό 11,7%.
- Ο **Firefox** έχει επίσης χάσει ένα ποσοστό των χρηστών του κατά την ίδια περίοδο (πτώση 4,6%), ωστόσο παραμένει σχετικά σταθεροποιημένος σε υψηλά επίπεδα με ποσοστό 27,2%.
- Ο **Chrome** ανεβαίνει σταθερά με αύξηση 8,3% από το Οκτώβρη του 2012, καταλαμβάνοντας πλέον το μεγαλύτερο μερίδιο χρηστών με ποσοστό 54,1%.
- **Safari** και **Opera** καταλαμβάνουν πολύ μικρό ποσοστό της μερίδας χρηστών, αν κ η σημασία του Safari δε θα έπρεπε να θεωρηθεί αμελητέα λόγω της έντονης χρήσης του από κινητά τηλέφωνα iPhone. Ωστόσο, το ποσοστό του Safari παραμένει στάσιμο σε χαμηλά επίπεδα (3.8% σε αντιστοιχία με 4,3% κατά τον Οκτώβρη του 2012), ενώ ο Opera κινείται σε πολύ χαμηλά επίπεδα για να ληφθεί υπ' όψιν (1,7%).

Στα πλαίσια της εξέλιξης των browsers αλλά και της συνολικής κατεύθυνσης των χρηστών, τέθηκε ένα κατώτατο όριο για το οποίο θα γινόταν έλεγχος της εφαρμογής στο 5%. Αυτό πρακτικά σημαίνει πως εάν ένας browser χρησιμοποιείται από λίγοτερο από 5% των χρηστών δε θα λαμβανόταν υπόψιν κατά τις δοκιμές.

Έτσι, σύμφωνα με τις παραπάνω παρατηρήσεις, η εφαρμογή ελέγχθηκε και σταθεροποιήθηκε για τους παρακάτω browsers και τις αντίστοιχες εκδόσεις τους[9]:

1	Chrome 30	22.98%
2	Firefox 24	10.90%
3	Internet Explorer 10	10.07%
4	Internet Explorer 8	7.77%
5	Safari 7	5.38%
6	Internet Explorer 9	5.09%

Πίνακας 0-1. Στατιστικά χρήσης browsers (Νοέμβρης 2013)

4.4 Ασφάλεια εφαρμογής

Όσον αφορά την ασφάλεια της εφαρμογής έγιναν όλες οι απαραίτητες ενέργειες για την προστασία των δεδομένων και της εφαρμογής αυτής καθ' εαυτής από κακόβουλες ενέργειες. Έτσι, σε πρώτη φάση χρησιμοποιώντας το inbuilt function system του Joomla! εκμεταλλευόμαστε τις δυνατότητές του ώστε να αποφύγουμε SQL injection και XSS επιθέσεις.

Για λόγους πληρότητας, και επειδή κάποιοι servers ίσως να μην έχουν κάνει την απαραίτητη ρύθμιση, τοποθετήθηκαν σε κάθε φάκελο κενά αρχεία index.html. Με αυτόν τον τρόπο αποφεύχθηκε η

[1] Τα στατιστικά χρήσης των διαφόρων browsers λήφθηκαν υπ' όψιν από τις ιστοσελίδες w3schools.com και w3counter.com. Για ενημερωμένα στατιστικά ο όποιος ενδιαφερόμενος μπορεί να επισκεφτεί τα παρακάτω links:
α) w3schools.com: http://www.w3schools.com/browsers/browsers_stats.asp
Β) w3counter.com: http://www.w3counter.com/globalstats.php
[2] Λόγω αδυναμίας συνεχούς ενημέρωσης με τα σύγχρονα δεδομένα και στατιστικά χρήσης, τα στατιστικά που παρουσιάζονται είναι όπως αυτά έχουν προκύψει μέχρι τον Ιούνιο του 2013.

πλοήγηση σε συγκεκριμένο directory με σκοπό την εμφάνιση όλων των περιεχομένων του στον χρήστη. Αντ' αυτού, ο χρήστης θα βρεθεί αντιμέτωπος με μια κενή λευκή σελίδα, αποτέλεσμα του κενού αρχείου index.html.

Αυτό γίνεται καθώς, εάν υπάρχει κάποιο αρχείο με το συγκεκριμένο όνομα σε ένα directory, ο χρήστης κατευθύνεται εξ' ορισμού σε αυτό. Σε περίπτωση παραδείγματος χάριν που πληκτρολογήσει το παρακάτω link

```
http://www.somepage.gr/some-directory/
```

και στο φάκελο `some-directory` υπάρχει ένα αρχείο index.html, στην ουσία θα πλοηγηθεί αυτόματα στη διεύθυνση

```
http://www.somepage.gr/some-directory/index.html
```

Εάν όμως δεν υπάρχει κάποιο αρχείο με αυτή την ονομασία, υπάρχει η πιθανότητα (αναλόγως των ρυθμίσεων του web server) να μπορέσει να δει και να ανακτήσει όλα τα αρχεία του directory.

Τέλος, κάθε αρχείο στον κώδικά μας καλείται μέσα από τη δρομολόγηση συγκεκριμένων ενεργειών του Joomla, η οποία κατευθύνεται μέσα από το αρχείο index.php που βρίσκεται στο ανώτερο επίπεδο directory της ιστοσελίδας (root). Οποιαδήποτε άλλη πρόσβαση σε κάποιο εκτελέσιμο php αρχείο θεωρείται κακόβουλη και θα πρέπει να απαγορεύεται.

Για να αποφύγουμε την άμεση πρόσβαση σε ένα εκτελέσιμο αρχείο του κώδικά μας, και σύμφωνα με τα πρότυπα κατασκευής και τις συμβάσεις (conventions) του Joomla!, τοποθετήθηκε στην πρώτη εκτελέσιμη σειρά κάθε εκτελέσιμου αρχείου php η γραμμή

```
defined('_JEXEC') or die('Restricted access');
```

Με αυτόν τον τρόπο απαγορεύτηκε οποιαδήποτε άμεση πρόσβαση στα αρχεία του κώδικά μας.

Φυσικά, ο κώδικας δεν είναι απροσπέλαστος για κάποιον επίδοξο hacker, και δεν μπορεί να προστατευτεί ποτέ πλήρως. Μεγαλύτερο μειονέκτημα του κώδικα είναι αυτό ακριβώς που του δίνει τόση ελαστικότητα και εξελιξιμότητα: ο κώδικας έχει γραφτεί με συγκεκριμένες συμβάσεις (conventions), καθώς χρησιμοποιείται από το Joomla!, οι οποίες είναι φανερές σε όλους. Έτσι, ο οποιοσδήποτε γνώστης της αρχιτεκτονικής ενός οποιουδήποτε συστήματος Joomla! γνωρίζει και την αρχιτεκτονική του component που παρουσιάζεται, χωρίς μάλιστα να χρειάζεται καν να το εξετάσει.

Αυτό είναι ένα σημαντικό μειονέκτημα ασφάλειας σε όλα τα ευρέως χρησιμοποιούμενα CMS (όπως Joomla!, Drupal, Wordpress κτλ) και όποιος αποφασίζει να χρησιμοποιήσει ένα τέτοιο CMS για την ιστοσελίδα του θα πρέπει να γνωρίζει πως η επεκτασιμότητα και η υποστήριξη που θα έχει από την τεράστια κοινότητα που το υποστηρίζει έρχεται με κάποιο κόστος της ασφάλειας του συστήματος, λόγω της συγκεκριμένης αρχιτεκτονικής με τον οποία αυτά δομούνται.

5 Συμπεράσματα

Η εφαρμογή ClassRes επεκτείνει τις δυνατότητες του Joomla! δίνοντάς του δυνατότητα δημιουργίας και διαχείρισης αιθουσών και κρατήσεων. Όπως κάθε component, βοηθάει και αυτό στην περαιτέρω ανάπτυξη ενός ήδη διάσημου και ευρέως χρησιμοποιούμενου CMS, συμβάλλοντας στην εξέλιξη της κοινότητας των CMS που φιλοδοξούν να κάνουν τη διαχείριση των δυναμικών ιστοσελίδων προσβάσιμη σε όλους τους χρήστες, είτε έχουν προγραμματιστική γνώση, είτε όχι.

Σε ένα ολοένα εξελίξιμο περιβάλλον, οποιαδήποτε εφαρμογή θα πρέπει να είναι συγχρονισμένη με τις εξελίξεις και να διατηρείται σε όσο το δυνατόν πιο εκσυγχρονισμένα επίπεδα, ώστε να εκμεταλλεύεται στο έπαρκο όλες τις δυνατότητες που η τεχνολογία μπορεί να προσφέρει, αλλά και της ασφάλειας ενός πιο μοντέρνου συστήματος.

Η συγκεκριμένη εφαρμογή σε ένα χρόνο από τώρα θα είναι ξεπερασμένη. Η εξέλιξη του Joomla!, που επιβάλλει τη δημοσίευση μιας νέας έκδοσης κάθε 6 μήνες, θα καταστήσει την εφαρμογή ξεπερασμένη σε αρκετά μικρό διάστημα. Για τη βιωσιμότητα της εφαρμογής, είναι απαραίτητο να γίνονται οι αντίστοιχες αναβαθμίσεις στον κώδικά της, ώστε να είναι συμβατή με τις νεότερες εκδόσεις του Joomla! που θα προκύψουν.

Τέλος, τίθεται το μεγάλο ζήτημα της ασφάλειας και το κατά πόσο μπορεί ένα CMS σαν το Joomla!, όσο μεγάλη λειτουργικότητα κι αν έχει, να υποστηρίξει την κατασκευή σημαντικών έργων με τρόπο που να μην κλονίζεται η ασφάλεια του συστήματος λόγω της συγκεκριμένης δομής που πρέπει να έχει το Joomla!.

Σύμφωνα με τη λογική με την οποία "ασφαλίζεται" μια ιστοσελίδα, ο χρήστης ή το κακόβουλο πρόγραμμα γνωρίζει πάντα την αρχιτεκτονική της ιστοσελίδας μας, Πρακτικές "απόκρυψης" είναι κακές και δεν οδηγούν σε πραγματική και ουσιαστική ασφάλεια, καθώς είναι απλά θέμα ανακάλυψης η κατάρρευση των συστημάτων που στηρίζονται σε αυτή τη λογική.

Από τα παραπάνω μπορούμε να υποστηρίξουμε πως, με την ανάπτυξη της κοινότητας του Joomla! και των συνεχών αναβαθμίσεων αυτού θα οδηγηθούμε τελικά σε ένα όσο το δυνατόν ασφαλέστερο σύστημα το οποίο, ακόμα και με εμφανή αρχιτεκτονική δομή, θα είναι σε θέση να αμυνθεί απέναντι σε οποιοδήποτε είδος επίθεσης.

Το Joomla! είναι σαν ένας ζωντανός οργανισμός και εξελίσσεται συνεχώς από μια τεράστια κοινότητα χρηστών που το χρησιμοποιούν καθημερινά αλλά και προγραμματιστών που το επεκτείνουν. Αν και χρειάζεται αρκετή δουλειά ακόμα, οι ενδείξεις και η εξέλιξη που υπάρχει από παλαιότερες σε νεότερες εκδόσεις είναι ενθαρρυντικές της επιτυχημένης πορείας του δημοφιλούς αυτού CMS.

Βιβλιογραφία

[1] «Joomla Application Development and Its Significance in Web Development,» [Ηλεκτρονικό]. Available: http://www.groundreport.com/Business/Joomla-Application-Development-and-Its-Significanc/2953237.

[2] «Wikipedia,» [Ηλεκτρονικό]. Available: http://en.wikipedia.org/wiki/Joomla.

[3] «What is Joomla!™ CMS,» [Ηλεκτρονικό]. Available: http://www.siteground.com/tutorials/joomla/.

[4] "J2.5: Developing a MVC Component/Developing a Basic Component," [Online]. Available: http://docs.joomla.org/J2.5:Developing_a_MVC_Component/Developing_a_Basic_Component.

[5] «Developing a MVC Component,» [Ηλεκτρονικό]. Available: http://docs.joomla.org/J2.5:Developing_a_MVC_Component/Introduction.

[6] «Joomla Documentation,» [Ηλεκτρονικό]. Available: http://docs.joomla.org/Component.

[7] L. Library, «Developing Joomla 2.5 MVC Components,» [Ηλεκτρονικό]. Available: http://library.logicsistemi.it/en/joomla/developing-joomla-25-mvc-components.

[8] «W3Schools.com,» [Ηλεκτρονικό]. Available: http://www.w3schools.com/browsers/browsers_stats.asp.

[9] «W3Counter.com,» [Ηλεκτρονικό]. Available: http://www.w3counter.com/globalstats.php.

[10] «Joomla 2.5 Custom Component Development,» [Ηλεκτρονικό]. Available: http://www.biswarupadhikari.com/2013/05/joomla-25-custom-component-development_5911.html.

[11] «Top Ten Joomla Security Issues,» [Ηλεκτρονικό]. Available: http://www.deanmarshall.co.uk/joomla-services/joomla-security/joomla-security-issues.html.

[12] «The Most Commonly Known Joomla Security Issues,» [Ηλεκτρονικό]. Available: http://www.webdevelopmentconsultancy.com/joomla-website-security/joomla-security-issues.html.