ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ ΣΕΡΡΩΝ

ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ & ΕΠΙΚΟΙΝΩΝΙΩΝ

ΔΗΜΙΟΥΡΓΙΑ ΔΙΑΔΙΚΤΥΑΚΟΥ ΣΥΣΤΗΜΑΤΟΣ ΣΥΝΑΝΤΗΣΕΩΝ (APPOINTMENTS) ΜΕ ΧΡΗΣΗ GOOGLE CALENDAR PHP API

Πτυχιακή εργασία του

Αλέξανδρος Τσελεγγίδης (2503)

Επιβλέπων: Δρ. Νικόλαος Πεταλίδης, Επιστημονικός Συνεργάτης

ΣΕΡΡΕΣ, ΝΟΕΜΒΡΙΟΣ 2013

Υπεύθυνη δήλωση

Υπεύθυνη Δήλωση: Βεβαιώνω ότι είμαι συγγραφέας αυτής της πτυχιακής εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της, είναι πλήρως αναγνωρισμένη και αναφέρεται στην πτυχιακή εργασία. Επίσης έχω αναφέρει τις όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών ή λέξεων, είτε αυτές αναφέρονται ακριβώς είτε παραφρασμένες. Επίσης βεβαιώνω ότι αυτή η πτυχιακή εργασία προετοιμάστηκε από εμένα προσωπικά ειδικά για τις απαιτήσεις του προγράμματος σπουδών του Τμήματος Πληροφορικής & Επικοινωνιών του Τ.Ε.Ι. Σερρών.

Σύνοψη

Η εργασία αυτή πραγματεύεται την υλοποίηση ενός διαδικτυακού συστήματος κρατήσεων ραντεβού για επιχειρήσεις, με πλήρη περιβάλλον διαχείρισης και την δυνατότητα συγχρονισμού των ραντεβού με το Google Calendar API. Στόχος είναι να κατασκευαστεί ένα ευέλικτο σύστημα, το οποίο να είναι σε θέση να εξυπηρετήσει τις ανάγκες οποιασδήποτε επιχείρησης, βελτιώνοντας έτσι την μηχανογράφηση και την οργάνωση της και κατ' επέκταση την απόδοση της. Όλα αυτά σαφώς συντελούν στην μείωση του κόστους λειτουργίας, κάτι το οποίο είναι πολύ σημαντικό. Για την υλοποίηση της εφαρμογής επιλέχθηκε η γλώσσα προγραμματισμού PHP και Javascript καθώς και κάποιες εξωτερικές βιβλιοθήκες κώδικα, οι οποίες φάνηκαν πολύ χρήσιμες κατά την υλοποίηση του συστήματος.

Περιεχόμενα

Ορισμοί

Στο έγγραφο αυτό υπάρχουν κάποιες έννοιες οι οποίες χρησιμοποιούνται σε διάφορα σημεία. Παρακάτω παρατίθενται οι περιγραφές τους.

Διαχειριστής Ο διαχειριστής του συστήματος είναι ο χρήστης ο οποίος έχει όλα τα δικαιώματα αλλαγών και ρυθμίσεων του Easy!Appointments. Μπορεί να ορίσει νέες υπηρεσίες και πάροχους υπηρεσίας, να ρυθμίσει το σύστημα ειδοποιήσεων και να εκτελέσει όλες τις δυνατές διαδικασίες διαχείρισης των δεδομένων.

Πάροχος Υπηρεσίας Ο πάροχος υπηρεσίας είναι η οντότητα που εξυπηρετεί μια ή περισσότερες υπηρεσίες. Μπορεί να αντιπροσωπεύει ένα άτομο ή μια ομάδα ατόμων. Σε κάθε περίπτωση όμως διαχειρίζεται από έναν χρήστη του συστήματος.

Πελάτης Ο πελάτης αφού δει τις διαθέσιμες ημερομηνίες και ώρες για τις επιλεγμένες υπηρεσίες και παρόχους, μπορεί να κλείνει ραντεβού με την επιχείρηση. Αν γίνει οποιαδήποτε αλλαγή σε κάποιο ραντεβού του πελάτη τότε αυτός θα ενημερωθεί με σχετικό email.

Γραμματέας Ο γραμματέας είναι ένας χρήστης ο οποίος μπορεί να διαχειριστεί τα ραντεβού και τους πελάτες του συστήματος για συγκεκριμένους πάροχους υπηρεσιών. Το σε ποιους πάροχους αντιστοιχεί ο κάθε χρήστης γραμματέας ορίζεται από τον διαχειριστή στο περιβάλλον ρυθμίσεων της εφαρμογής.

Πλάνο Πάροχου Από την στιγμή που κλείνονται ραντεβού σε έναν πάροχο υπηρεσιών το ημερολογιακό του πλάνο αρχίζει να γεμίζει από χρονικά διαστήματα, τα οποία είναι δεσμευμένα και αντιπροσωπεύουν συναντήσεις με τους πελάτες. Εκτός αυτού υπάρχει και η δυνατότητα να τεθεί ένα ανενεργό χρονικό διάστημα, στο οποίο ο συγκεκριμένος πάροχος δεν θα είναι διαθέσιμος έτσι ώστε να μην μπορούν οι πελάτες να κλείνουν ραντεβού σε αυτό το διάστημα. Αυτό το πλάνο

μπορεί να συγχρονιστεί με το Google Calendar έτσι ώστε να είναι προσβάσιμο και από άλλες υπηρεσίες.

Easy!Appointments Με την ονομασία Easy!Appointments θα γίνεται αναφορά στο σύστημα που υλοποιήθηκε.

Κεφάλαιο 1

Εισαγωγή

Το παρόν κεφάλαιο επεξηγεί τον σκοπό ανάπτυξης του Easy!Appointments καθώς και τις ανάγκες που καλύπτει σε μια επιχείρηση. Επιπρόσθετα αναφέρονται παρόμοια συστήματα και οι διαφορές που έχουν σε σχέση με την εφαρμογή.

1.1 Ποια προβλήματα προσπαθεί να λύσει η εφαρμογή

Οι επιχειρήσεις από την φύση τους χρειάζεται να έρχονται σε επαφή με τους πελάτες, για να μπορέσουν να τους εξυπηρετήσουν και έτσι να πάρουν την αμοιβή τους. Ανάλογα με την μορφή και το είδος της επιχείρησης, η επαφή αυτή διαφέρει. Για παράδειγμα κάποιες επιχειρήσεις έρχονται σε επαφή με περισσότερους πελάτες, άλλες με λιγότερους αλλά η εξυπηρέτηση μπορεί να είναι παθητική (πχ κατάστημα ηλεκτρονικών ειδών) και κάποιες απαιτούν ιδιαίτερη προσοχή στον πελάτη καθώς η εξυπηρέτησή του μπορεί να γίνει μόνο προσωπικώς, από κάποιον υπάλληλο ή επαγγελματία (πάροχος της υπηρεσίας). Η τελευταία κατηγορία περιέχει ένα μεγάλο εύρος επιχειρήσεων το οποίο για να οργανώσει και να διευκολύνει το πελατειακό κοινό του, λειτουργεί κανονίζοντας ραντεβού με τους ενδιαφερόμενους πελάτες.

Η κράτηση ενός ραντεβού είναι μια διαδικασία η οποία γίνεται συνήθως τηλεφωνικώς, είτε μετά από προσωπικό κανονισμό με κάποιον αρμόδιο. Η διαδικασία αυτή αποτελείται συνήθως από τα παρακάτω μέρη:

1. Ο ενδιαφερόμενος πελάτης έρχεται σε επαφή με την επιχείρηση και ζητάει να κάνει κράτηση την επιθυμητή ημερομηνία και ώρα, για μια συγκεκριμένη υπηρεσία.

2. Ο αρμόδιος υπάλληλος ψάχνει σε κάποιο ημερολόγιο ή αρχείο υπολογιστή τα ραντεβού για την συγκεκριμένη ημερομηνία και ανάλογα με την διαθεσιμότητα ανταποκρίνεται στον πελάτη.

3. Αν η συγκεκριμένη χρονική στιγμή δεν είναι διαθέσιμη θα χρειαστεί να γίνει μια αντιπρόταση από τον υπάλληλο ή ο πελάτης να βρει κάποια άλλη στιγμή που θα είναι αυτός διαθέσιμος.

Αν παρατηρήσουμε όμως την παραπάνω διαδικασία, θα δούμε πως έχει κάποια σημαντικά μειονεκτήματα, τα οποία μάλιστα συνεπάγονται την αύξηση του κόστους λειτουργίας μιας επιχείρησης και την μείωση της ποιότητας εξυπηρέτησης των πελατών.

Η ίδια η διαδικασία της κράτησης ενός ραντεβού με τον συγκεκριμένο τρόπο απαιτεί από μόνη της την ύπαρξη ενός υπαλλήλου, ο οποίος θα σπαταλάει αρκετό, αν όχι τον περισσότερο, από τον χρόνο του για να κάνει αυτήν την εργασία. Αυτό πρακτικά σημαίνει δέσμευση ανθρώπινων πόρων της επιχείρησης και συνεπάγεται στην αύξηση των εξόδων λειτουργίας.

Επιπλέον, η ίδια η διαδικασία μπορεί να είναι χρονοβόρα και κουραστική για τους πελάτες, ειδικά στις περιπτώσεις όπου υπάρχει λίγο προσωπικό για να καλύψει μεγάλο κοινό (πχ νοσοκομεία). Στις περιπτώσεις αυτές οι πελάτες περιμένουν στην αναμονή για μεγάλο χρονικό διάστημα και μάλιστα πολλές φορές δεν πιάνουν γραμμή για να μπορέσουν να κρατήσουν κάποιο ραντεβού. Επίσης πρέπει να σημειωθεί ότι όταν ο πελάτης καταφέρει να κλείσει το ραντεβού του, συνήθως δεν έχει επιλογή για το πότε θα γίνει και απλώς ενημερώνεται για την ημερομηνία την οποία έχει ορίσει το προσωπικό, ανάλογα με τις εκάστοτε συνθήκες.

Εκτός αυτών η εκτέλεση αυτής της διαδικασίας είναι αρκετά επιρρεπής στο να έχει ασαφές αποτελέσματα, με την έννοια του ότι δεν υπάρχει κάποιο κοινό σημείο αναφοράς για την συμφωνία που πραγματοποιείται μεταξύ της επιχείρησης και του πελάτη, έτσι ώστε να μπορεί να γίνει εξακρίβωση και επαλήθευση των ιδιοτήτων μιας κράτησης και από τις δύο πλευρές. Αυτό μπορεί να οδηγήσει σε προβλήματα με τους πελάτες, κάτι το οποίο δεν είναι επιθυμητό σε καμία περίπτωση.

Μικρό είναι το μέρος των πληροφοριών που καταγράφεται με το πέρας της κράτησης, καθώς τα μέσα που χρησιμοποιούνται δεν επιτρέπουν ή κάνουν δύσκολη και χρονοβόρα την αποθήκευση όλων των δεδομένων. Αυτό συντελεί στην επιπλέον μεί-

ωση της ποιότητας εξυπηρέτησης και της απόδοσης της επιχείρησης.

Τα δεδομένα αυτά διαχειρίζονται συνήθως δύσκολα. Ακόμα και στην περίπτωση που χρησιμοποιούνται ηλεκτρονικά μέσα για την αποθήκευση των κρατήσεων, η τροποποίηση ή ο έλεγχος μπορούν να είναι δύσκολες και χρονοβόρες διαδικασίες, οι οποίες εξαρτώνται κάθε φορά από το επίπεδο της οργάνωσης της επιχείρησης και τις τεχνολογίες που χρησιμοποιούνται.

Επίσης τα δεδομένα αυτά δεν είναι προσβάσιμα από οποιονδήποτε ανά πάσα στιγμή, αλλά μόνο στον χώρο της επιχείρησης και μόνο από το άτομο το οποίο διαχειρίζεται τα ραντεβού.

Τα παραπάνω προβλήματα διογκώνονται σημαντικά όταν πρόκειται για μεγάλες επιχειρήσεις και οργανισμούς, οι οποίοι εξυπηρετούν μεγάλο αριθμό πελατών.

1.2 Γιατί είναι σημαντικά τα προβλήματα αυτά

Οι απαιτήσεις και η ανταγωνιστικότητα που υπάρχει μεταξύ των επιχειρήσεων στην εποχή μας, απαιτεί την γρήγορη και άμεση διεκπεραίωση διεργασιών και την όσο το δυνατόν καλύτερη οργάνωση τους, για να μπορούν να παρέχουν υπηρεσίες υψηλού επιπέδου με το χαμηλότερο δυνατό κόστος και προσωπικό. Για να επιτύχουν τον σκοπό αυτό οι επιχειρήσεις πρέπει να επιλέξουν τα κατάλληλα εργαλεία οργάνωσης και εξυπηρέτησης των πελατών τους.

Βλέποντας τα προβλήματα που αναφέρθηκαν προηγουμένως είναι κατανοητό ότι με την χρήση της έως τώρα μεθόδου κράτησης ραντεβού, επέρχεται μείωση της ποιότητας και της απόδοσης της επιχείρησης. Αυτό σημαίνει ότι το επίπεδο εξυπηρέτησης είναι χαμηλότερο και έτσι η επιχείρηση αδυνατεί να είναι ανταγωνιστική προς τις άλλες καθώς γίνεται σπατάλη πόρων για την υλοποίηση αυτής της διαδικασίας.

Η μείωση αυτή επιφέρει αύξηση του κόστους λειτουργίας το οποίο αποτελεί ένα επιπρόσθετο εμπόδιο στην προσπάθεια για ανάπτυξη και επέκταση. Πολλές φορές μάλιστα αυτή η αύξηση του κόστους σε συνδιασμό με άλλους παράγοντες μπορεί να συντελέσουν στην μη βιωσιμότητα και το κλείσιμο της επιχείρησης, εφόσον αυτή δεν μπορεί να παράγει κέρδη.

Παρατηρούμε λοιπόν ότι η σημερινή οργάνωση των επιχειρήσεων που λειτουργούν με ραντεβού, θα μπορούσε να βελτιωθεί με την χρήση ενός ηλεκτρονικού συ-

στήματος που θα επίλυε τα προαναφερθέντα προβλήματα και θα πρόσδιδε μεγαλύτερη ευκολία στην εξυπηρέτηση του κοινού. Η προτεινόμενη λύση αποσκοπεί στο να εκπληρώσει αυτά τα κενά και να εντάξει στο ενεργητικό της επιχείρησης ένα δυνατό εργαλείο οργάνωσης.

1.3 Παρόμοιες λύσεις που υπάρχουν ήδη

1.3.1 Genbook

Το Genbook είναι μια online υπηρεσία που προσφέρει στις επιχειρήσεις την δυνατότητα να εγγραφούν (πληρώνοντας το αντίτιμο) και να χρησιμοποιήσουν την εφαρμογή που τους επιτρέπει να διαχειρίζονται τα ραντεβού. Παρέχει αρκετά φιλικό περιβάλλον, είναι παραμετροποιήσιμο και περιέχει την δυνατότητα παραγωγής στατιστικών στοιχείων για τις υπηρεσίες που είναι διαθέσιμες προς το κοινό.

Αυτό που δεν υποστηρίζει είναι η δημιουργία πολλαπλών πλάνων που να αντιπροσωπεύουν διαφορετικούς τομείς ή υπαλλήλους (όλα τα ραντεβού φαίνονται σε ένα ημερολόγιο).

Παρατηρήσεις : επί πληρωμή, ικανοποιητικά παραμετροποιήσιμο, δημιουργία στατιστικών

www.genbook.com

1.3.2 Web Appointment Scheduling System (Open Source)

Το WASS είναι μια λύση ανοιχτού κώδικα, η οποία περιέχει τις βασικότερες λειτουργίες διαχείρισης ραντεβού για μια επιχείρηση. Από την εφαρμογή αυτή λείπουν κάποια στοιχεία διαχείρισης και παραμετροποίησης και το γραφικό περιβάλλον του χρήστη χρειάζεται επιπλέον δουλειά. Παρ' όλα αυτά είναι δωρεάν και προτείνεται για οποιαδήποτε μικρή επιχείρηση. Η εφαρμογή υποστηρίζει το iCal της Apple και την δημιουργία πολλών πλάνων.

Παρατηρήσεις : δωρεάν, βασικές λειτουργίες, ανοιχτός κώδικας, υποστήριξη iCal

www.wass.princeton.edu | www.sourceforge.net

1.3.3 Appointment-plus

Το Appointment-plus είναι από τις πιο οργανωμένες και εμπλουτισμένες εφαρμογές που υπάρχουν σε αυτόν τον τομέα. Έχει εκδόσεις για οποιαδήποτε συσκευή (pc, tablets, smartphones), όμορφο περιβάλλον, υποστήριξη συγχρονισμού δεδομένων με άλλες υπηρεσίες και εφαρμογές (Google, Outlook, iCal κτλ), χρήση από πολλούς υπαλλήλους και πολλά πλάνα. Παρέχει λειτουργία αναμονής για τους πελάτες σε περίπτωση που η επιθυμητή ώρα είναι πιασμένη. Υπάρχει σύστημα email στα οποία ο πελάτης μπορεί να κάνει επιλογές και να τις αποστείλει πίσω στο σύστημα. Δυνατότητα για τροποποίηση της σελίδας που βλέπει ο χρήστης και πώλησης προϊόντων μέσω της εφαρμογής. Η εταιρεία προσφέρει σε κάθε πελάτη έναν βοηθό στον οποίο θα μπορεί να απευθυνθεί για να ρυθμίσει την εφαρμογή.

Παρατηρήσεις : επί πληρωμή, πλήρως παραμετροποιήσιμο, λειτουργία σε διαφορετικές συσκευές, πολλαπλά πλάνα, υποστήριξη (τηλέφωνο - email), το χρησιμοποιούν μεγάλες εταιρείες, οργανισμοί και πανεπιστημιακά ιδρύματα

www.appointment-plus.com

1.3.4 Acuity Scheduling

Η εφαρμογή αυτή αν και πιο απλή από την Appointment-plus περιέχει όλες τις βασικές λειτουργίες που θα χρειαστεί μια επιχείρηση για την υλοποίηση ενός συστήματος ραντεβού. Υποστηρίζει την δυνατότητα τροποποίησης της εμφάνισης, δημιουργία πολλών υπαλλήλων και πλάνων, ηλεκτρονικών πληρωμών με πιστωτική κάρτα, εξαγωγή ημερολογίου σε άλλες εφαρμογές (Facebook, Google και Outlook), ιστορικό, διαχείριση πελατών και λειτουργία σε iPhone. Επίσης δίνει την δυνατότητα πώλησης προϊόντων (eshop) για ηλεκτρονικές αγορές.

Παρατηρήσεις : επί πληρωμή (δωρεάν για έναν χρήστη αλλά με περιορισμούς), υποστήριξη προϊόντων, iPhone, εξαγωγή σε Google, Outlook και Facebook

www.acuityscheduling.com

1.3.5 SetMore

Το SetMore έχει απλή και όμορφη εμφάνιση και παρέχει ένα πλήρως παραμετροποιήσιμο περιβάλλον. Είναι το μόνο που υποστηρίζει SMS ειδοποιήσεις και αυτό σε

beta στάδιο. Η διαδικασία κράτησης ενός ραντεβού χωρίζεται όπως και με τα υπόλοιπα συστήματα, σε έναν οδηγό με 4-5 βήματα στα οποία ο χρήστης επιλέγει σε ποια υπηρεσία και σε ποιόν υπάλληλο θέλει να κλείσει ραντεβού. Είναι πολύ εύκολο στην χρήση και υποστηρίζει plugins για το WordPress και το Facebook.

Παρατηρήσεις : επί πληρωμή, υποστήριξη SMS, λειτουργία με WordPress και Facebook, δεν μπορεί να εξάγει τα δεδομένα του σε άλλη εφαρμογή (Google Calendar, Outlook)

www.setmore.com

1.3.6 Υπόλοιπα συστήματα

Εκτός των αναφερθέντων υπάρχουν πολλές άλλες εφαρμογές διαθέσιμες προς το κοινό. Μερικές από αυτές αναφέρονται στην παρακάτω λίστα:

1. SnapAppointments
2. Doodle
3. Bookeo
4. ScheduleOnce
5. BookingBug
6. SetSter
7. Agreedo
8. BookedIn
9. Book’d
10. Schedulista

1.4 Σε τι διαφέρει από τις υπόλοιπες η προτεινόμενη λύση

Το Easy!Appointments έχει ως σκοπό να αυτοματοποιήσει την διαδικασία της κράτησης και διαχείρισης ραντεβού για οποιαδήποτε επιχείρηση. Χρησιμοποιώντας τις δυ-

νατότητες που μας παρέχει το διαδίκτυο, μπορεί να υλοποιηθεί ένα σύστημα το οποίο να έχει την δυνατότητα να οργανώσει τα επαγγελματικά πλάνα πολλών υπαλλήλων ταυτόχρονα, επιφέροντας έτσι όχι μόνο την μείωση του χρόνου που απαιτούσαν οι παλιές μέθοδοι διαχείρισης ραντεβού, αλλά και την αύξηση της παραγωγικότητας της επιχείρησης. Οι πελάτες δεν θα χρειάζεται πλέον να τηλεφωνούν ή να πηγαίνουν στο κατάστημα, αλλά θα μπορούν να βλέπουν τις διαθέσιμες ώρες της επιχείρησης και να κλείνουν το ραντεβού τους την επιθυμητή ημερομηνία και ώρα, μέσω του υπολογιστή και του internet. Αυτό έχει ως αποτέλεσμα την ποιοτικότερη αλλά και αποδοτικότερη εξυπηρέτηση τους. Επιπρόσθετα βελτιώνεται η επικοινωνία και η οργάνωση των συντελεστών της επιχείρησης, παρέχοντας δυνατότητες αρχειοθέτησης και διαχείρισης των δεδομένων που αποθηκεύονται στο σύστημα ανά πάσα στιγμή και σε οποιοδήποτε μέρος. Σε αντίθεση με τα άλλα συστήματα, προσφέρει επιπλέον τα εξής:

1. **Αυτόνομη Εγκατάσταση :** Η επιχείρηση που θέλει να χρησιμοποιήσει την εφαρμογή θα μπορεί να την εγκαταστήσει στον server της και να την τρέξει μαζί με κάποιο άλλο site, έχοντας έτσι πλήρη πρόσβαση στα δεδομένα και τον κώδικα. Η διαδικασία της εγκατάστασης και παραμετροποίησης θα είναι παρόμοια με άλλα συστήματα (Joomla, WordPress κτλ) και όσο πιο αυτοματοποιημένη γίνεται.

2. **Διαμόρφωση Πρότυπου Πλάνου :** Το σύστημα θα έχει ενσωματωμένη δυνατότητα δημιουργίας πρότυπου πλάνου για τον κάθε πάροχο υπηρεσιών, το οποίο θα αποτελεί την βάση της κάθε εβδομάδας και από εκεί και πέρα ο διαχειριστής θα μπορεί να κάνει αλλαγές. Η επανάληψη του πλάνου καθώς και το από ποιά προτυπα πλάνα θα αποτελείται ένας μήνας θα συμπεριλαμβάνονται στην ρύθμιση της εφαρμογής.

3. **Ρύθμιση Δικαιωμάτων Πάροχων :** Ο διαχειριστής θα έχει την δυνατότητα να ορίζει τα δικαιώματα αλλαγών και ρυθμίσεων που θα έχει στην διάθεσή του ο κάθε πάροχος υπηρεσιών. Έτσι μπορεί να διασφαλιστεί η ασφάλεια των δεδομένων όπου κρίνεται αυτό χρήσιμο, είτε να διευκολυνθεί η διαχείριση των ραντεβού έτσι ώστε να μπορεί ο κάθε πάροχος να διαχειρίζεται τα δικά του.

4. **Υποστήριξη Γραμματείας :** Αν παρόλα αυτά η εταιρεία ορίσει κάποια γραμματέα ως υπεύθυνση των ραντεβού, τότε είναι απαραίτητο να μπορεί να διαχειρίζεται μόνο αυτά και τους πελάτες που είναι καταχωρημένοι στο σύστημα. Το

Easy!Appointments υποστηρίζει την δημιουργία χρηστών που αντιπροσωπεύουν αυτόν τον σκοπό.

5. **Αμφίδρομος Συγχρονισμός με το Google Calendar :** Το σύστημα θα υποστηρίζει τον αμφίδρομο συγχρονισμό ραντεβού με το Google Calendar, κάνοντας χρήση του Google Calendar API. Με αυτόν τον τρόπο η διαχείριση των ραντεβού μπορεί να γίνει ακόμα πιο εύκολη, λαμβάνοντας υπόψιν το πόσο δημοφιλής είναι η συγκεκριμένη υπηρεσία της Google.

Κεφάλαιο 2

Google Calendar API

Το Ημερολόγιο της Google είναι μια διαδικτυακή εφαρμογή που επιτρέπει στους χρήστες της να αποθηκεύουν και να διαχειρίζονται τα ραντεβού και τα συμβάντα τους σε ένα όμορφο περιβάλλον. Εκτός αυτού, υπάρχουν πολλές πρόσθετες δυνατότητες όπως ο συγχρονισμός ημερολογίου με κάποια άλλη εφαρμογή ή η κοινή χρήση ενός ημερολογίου από πολλά άτομα.

Όπως και με τα περισσότερα προϊόντα της Google παρέχονται εργαλεία με τα οποία μπορούν οι προγραμματιστές να επικοινωνήσουν και να λάβουν δεδομένα από τις υπηρεσίες της εταιρείας. Με αυτόν τον τρόπο είναι δυνατή η ανάπτυξη εφαρμογών οι οποίες διαχειρίζονται αυτά τα δεδομένα, διευκολύνοντας έτσι τον χρήστη.

Το Google Calendar API (Application Programming Interface) είναι μια πλατφόρμα διαχείρισης συμβάντων ενός ημερολογίου από την Google. Επιτρέπει στον προγραμματιστή να πραγματοποιήσει λειτουργίες προσθήκης, επεξεργασίας, διαγραφής και αναζήτησης συμβάντων μέσω ενός RESTful στυλ κλήσεων προς τον server.

Με την έννοια RESTful (Representional State Transfer) εννοείται ένας από τους πιο δημοφιλής τρόπους επικοινωνίας στον παγκόσμιο ιστό. Η επικοινωνία γίνεται με την χρήση ειδικών αιτήσεων προς τους servers, οι οποίοι με την σειρά τους είναι σε θέση να τις επεξεργαστούν και να επιστρέψουν δεδομένα πίσω στους clients. Οι μέθοδοι αιτήσεων που είναι διαθέσιμες είναι:

1. GET
2. POST
3. PUT

4. DELETE

Πρακτικά η μέθοδος επικοινωνίας RESTFul μπορεί να χρησιμοποιηθεί από οποιοδήποτε σύστημα υποστηρίζει το πρωτόκολλο HTTP. Για να διευκολύνει όμως η Google τους προγραμματιστές, έχει αναπτύξει βιβλιοθήκες κώδικα σε διάφορες γλώσσες προγραμματισμού (PHP, Java, .NET, Ruby κτλ) οι οποίες περιέχουν έτοιμες μεθόδους επικοινωνίας με τις υπηρεσίες της. Έτσι διευκολύνεται πολύ η διαδικασία ανάπτυξης μιας εφαρμογής που βασίζεται πάνω στα δεδομένα των χρηστών της Google.

Για να αποτραπεί η υπερβολική χρήση της υπηρεσίας Calendar, η εταιρεία έχει θέσει ένα υπέρτατο όριο 10.000 request την ημέρα. Αν κάποια εταιρεία ξεπεράσει αυτό το όριο τότε θα χρειαστεί να πληρώσει κάποιο αντίτιμο για να μπορέσει να συνεχίσει κανονικά την χρήση. Για αυτό τον λόγο είναι και απαραίτητο οποιοσδήποτε client χρησιμοποιεί το Calendar API, να έχει πρώτα δημιουργήσει ένα API Key μέσω της σελίδας API Console που προσφέρει η Google.

2.1 Περιγραφή του Calendar API

Το Ημερολόγιο της Google είναι ένα πολύ δυνατό και ευέλικτο εργαλείο. Οι χρήστες μπορούν να βλέπουν το ίδιο ημερολόγιο σε οποιαδήποτε συσκευή βρίσκονται έχοντας απλώς σύνδεση με το διαδίκτυο, για να μπορέσουν να ληφθούν τα δεδομένα από την υπηρεσία. Όλες οι εφαρμογές αυτές χρησιμοποιούν το API για να υλοποιήσουν τις βασικές λειτουργίες ενός ημερολογίου, δηλαδή την διαχείριση και την εύκολη εύρεση συμβάντων που είναι καταχωρημένα στο Google Calendar. Αφού γίνουν οι αλλαγές αυτές θα χρειαστεί να εκτελεστεί η διαδικασία του συγχρονισμού έτσι ώστε τα νέα δεδομένα να είναι και στις υπόλοιπες εφαρμογές που έχουν πρόσβαση στο ημερολόγιο.

Στην ευρεία χρήση της υπηρεσίας συντελεί το ότι η πλατφόρμα του ημερολογίου είναι συμβατή με διάφορες γλώσσες προγραμματισμού και έτσι μπορούν να υλοποιηθούν εφαρμογές για όλες τις συσκευές με εξελιγμένο λειτουργικό σύστημα (Windows, Linux, MacOsX, Android, iOs, Windows Phone κτλ).

Οι βασικές έννοιες του συστήματος ενός ημερολογίου είναι:

- Συμβάν (Event) - αντιπροσωπεύει ένα συμβάν στο ημερολόγιο το οποίο έχει τίτλο, περιγραφή, ημερομηνία και συμμετέχοντες.

- Ημερολόγιο (Calendar) - αντιπροσωπεύει ένα ημερολόγιο το οποίο περιέχει πολλά συμβάντα. Ένας χρήστης μπορεί να έχει πολλά ημερολόγια. Περιέχει επιπλέον πληροφορίες όπως η περιγραφή του ημερολογίου, ο ιδιοκτήτης κτλ.
- Λίστα Ημερολογίων (Calendar List) - αντιπροσωπεύει μια λίστα με όλα τα ημερολόγια ενός χρήστη της υπηρεσίας αυτής.
- Ρύθμιση (Setting) - αντιπροσωπεύει μια επιλογή του χρήστη, η οποία επηρεάζει τον τρόπο λειτουργίας της υπηρεσίας (πχ ρύθμιση ζώνης ώρας).
- ACL (Access Control Rule) - αντιπροσωπεύει έναν κανόνα πρόσβασης του ημερολογίου (όπως παραδείγματος χάρη το αν το ημερολόγιο είναι δημόσιο ή ιδιωτικό).
- Χρώμα (Color) - αντιπροσωπεύει το χρώμα για κάποια στοιχεία στο περιβάλλον χρήστη της εφαρμογής και τα συμβάντα.
- Ελεύθερος / Απασχολημένος (Free / Busy) - αντιπροσωπεύει μια χρονική περίοδο στο ημερολόγιο όπου ο χρήστης είναι είτε απασχολημένος είτε ελεύθερος. Αν είναι απασχολημένος δεν μπορούν να υπάρχουν συμβάντα σε αυτό το διάστημα.

Το API της Google λειτουργεί με “resources” και “collections” για να χειριστεί τις προαναφερθέντες έννοιες. Ένα resource αντιπροσωπεύει μια συγκεκριμένη οντότητα η οποία περιέχει δεδομένα για την εφαρμογή. Πολλά resource μαζί απαρτίζουν ένα collection το οποίο περιέχει πολλές χρήσιμες μεθόδους μαζικής διαχείρισης δεδομένων.

Οι προγραμματιστές διαμορφώνουν τον κωδικά τους έτσι ώστε να είναι συμβατός με αυτήν την δομή και έτσι η επικοινωνία με την υπηρεσία της Google να είναι ευκολότερη.

2.2 Πως χρησιμοποιείται

Η χρήση του API μπορεί να γίνει απευθείας με κλήσεις RESTful προς τον server της Google, είτε με χρήση κάποιων από τις έτοιμες βιβλιοθήκες που παρέχει η εταιρεία.

Επίσης είναι απαραίτητη η ύπαρξη ενός λογαριασμού στην Google καθώς και η καταχώρηση του project στο Google API Console έτσι ώστε να πάρει ο προγραμματιστής ένα API Key, ένα κλειδί το οποίο είναι απαραίτητο για την χρήση της υπηρεσίας.

Αν ο προγραμματιστής επιλέξει την χρήση της RESTful μεθόδου επικοινωνίας, θα χρειαστεί αν στέλνει request σε διάφορα URL και έτσι να παίρνει απαντήσεις με τα δεδομένα που χρειάζεται. Όλες οι απαντήσεις είναι σε JSON μορφή οπότε είναι πιθανόν να χρειαστεί να τις αναλύσει (parse) πριν τις χρησιμοποιήσει στην εφαρμογή του.

Δείγματα από URL για την κλήση διαφόρων μεθόδων:

```
1 https://www.googleapis.com/calendar/v3/lists/calendarListID/calendar
    ?parameters
2 https://www.googleapis.com/calendar/v3/users/userID/lists?parameters
```

Αντιθέτως με την χρήση των έτοιμων βιβλιοθηκών οι διαδικασίες επικοινωνίας είναι έτοιμες και ο προγραμματιστής μπορεί να πετύχει την επικοινωνία με τους servers της Google γρηγορότερα και ευκολότερα. Υπάρχουν βιβλιοθήκες για την Java, την PHP, την Python, το .NET περιβάλλον και την Ruby και πολλές άλλες γλώσσες και τεχνολογίες.

Αφού κατεβάσουμε τις βιβλιοθήκες και τις βάλουμε σε κάποιο σημείο μέσα στο project, μπορούμε να τις χρησιμοποιήσουμε μέσα από τον κώδικα:

```
1 require_once "../src/apiClient.php";
2 require_once "../src/contrib/apiCalendarService.php";
```

Στην συνέχεια είναι απαραίτητο να ρυθμίσουμε κάποιες παραμέτρους έτσι ώστε να είμαστε σε θέση να χρησιμοποιήσουμε αυτήν την υπηρεσία:

```
1  global $apiConfig;
2
3  $apiConfig = array(
4      // Site name to show in Google's OAuth authentication screen
5      'site_name' => 'www.example.org',
6
7      // OAuth2 Setting, you can get these keys on the API Access tab
         on
8      // the Google APIs Console
9      'oauth2_client_id' => 'YOUR_CLIENT_ID',
10     'oauth2_client_secret' => 'YOUR_CLIENT_SECRET',
```

```
    'oauth2_redirect_uri' => 'YOUR_REDIRECT_URL',

    // The developer key; you get this from the Google APIs Console
    'developer_key' => 'YOUR_DEVELOPER_KEY',
    ...

    // Which Authentication, Storage and HTTP IO classes to use.
    'authClass' => 'apiOAuth2',
    ....

    // Definition of service specific values like scopes, OAuth
       token URLs, etc
    'services' => array(
        'calendar' => array('scope' => 'https://www.googleapis.com/
           auth/calendar'),
    )
);
```

Έπειτα θα χρειαστεί να γίνει εκκίνηση του service και να ολοκληρωθεί η διαδικασία πιστοποίησης (authenticate) με την χρήση του API Key που αντιστοιχεί στην εφαρμογή.

```
<?php
session_start();

require_once "../src/apiClient.php";
require_once "../src/contrib/apiCalendarService.php";

$apiClient = new apiClient();
$apiClient->setUseObjects(true);
$service = new apiCalendarService($apiClient);

if (isset($_SESSION['oauth_access_token'])) {
  $apiClient->setAccessToken($_SESSION['oauth_access_token']);
} else {
  $token = $apiClient->authenticate();
  $_SESSION['oauth_access_token'] = $token;
}
...
```

Με αυτόν τον τρόπο εκτελούνται οι διαδικασίες ανταλλαγής δεδομένων μεταξύ του Google Calendar και του συστήματος του προγραμματιστή.

2.3 Συγχρονισμός ραντεβού

Ο συγχρονισμός δεδομένων μεταξύ δυο συστημάτων είναι μια περίπλοκη και υποτιμημένη διαδικασία, διότι ο προγραμματιστής έχει κάνει αρκετή δουλειά έτσι ώστε να καταφέρει να γεφυρώσει και τις δυο πηγές δεδομένων με τον καλύτερο τρόπο. Το αποτέλεσμα δεν μπορεί ποτέ να είναι 100% επιτυχές διότι μερικές φορές τα δεδομένα και οι αλλαγές μπορεί να έρχονται σε σύγκρουση (conflict) και έτσι θα χρειαστεί να παρθούν αποφάσεις είτε με βάση κάποιους κανόνες, είτε από τον ίδιο τον χρήστη για το ποια αλλαγή θα υπερισχύσει. Το πράγμα μάλιστα δυσκολεύει περισσότερο όταν δεν υπάρχει πρόσβαση στον κώδικα του ενός από τα δύο συστήματα (πχ Google Calendar) και όλη η διαδικασία θα πρέπει να τρέξει από το άλλο.

Στην περίπτωση του Easy!Appointments θα πρέπει να υλοποιηθεί μια διαδικασία η οποία θα συγχρονίζει τα ραντεβού και τα συμβάντα του συστήματος με αυτά του Google Calendar. Η διαδικασία αυτή θα εκτελείτε όταν δημιουργούνται συγκεκριμένα συμβάντα (πχ. προσθήκη ραντεβού) και θα φέρνει και τα δύο πλάνα στην ίδια κατάσταση. Ο συγχρονισμός θα εκτελείται κάθε φορά για το πλάνο ενός πάροχου υπηρεσιών και εφόσον έχει ήδη δοθεί η άδεια στην εφαρμογή να έχει πρόσβαση στα δεδομένα του Google Calendar του συγκεκριμένου χρήστη.

Με αυτόν τον τρόπο τα ραντεβού και οι αλλαγές που θα γίνονται από τα δυο συστήματα θα συγχωνεύονται και ο χρήστης θα μπορεί να τα διαχειρίζεται και από τις δύο πλευρές. Το μόνο πρόβλημα είναι ότι από την πλευρά του Google Calendar δεν είναι δυνατό να εκκινηθεί η διαδικασία του συγχρονισμού και έτσι αυτό θα πρέπει να γίνεται πάντοτε από την πλευρά του Easy!Appointments.

Η διαδικασία αυτή θα μπορούσε να αυτοματοποιηθεί με την χρήση της μεθόδου cron job, αλλά κάτι τέτοιο θα μπορούσε να αποφέρει επιπλέον προβλήματα, μιας και είναι απαραίτητο ο χρήστης να έχει τα κατάλληλα δικαιώματα στον server για να το κάνει και αυτό δεν είναι πάντα εφικτό. Οπότε η μέθοδος αυτή απορρίπτεται λόγο αυτής της δυσκολίας.

Η μέθοδος συγχρονισμού του Easy!Appointments είναι αμφίδρομη. Με την έννοια

αυτή εννοείται ότι συγχρονίζονται τόσο οι αλλαγές που γίνονται στο Easy!Appointments, όσο και οι αλλαγές που γίνονται από το Google Calendar, προσφέροντας έτσι μεγαλύτερη ελευθερία και προσβασιμότητα στα δεδομένα των χρηστών της εφαρμογής. Παρακάτω αναλύονται τα βήματα που ακολουθούνται κατα την διαδικασία του συγχρονισμού.

1. Η διαδικασία χωρίζεται σε δύο μέρη. Το πρώτο μέρος έχει να κάνει με τον συγχρονισμό μιας ενέργειας που μόλις έχει γίνει στο Easy!Appointments (πχ ένας πελάτης πραγματοποίησε μια κράτηση στο πλάνο ενός πάροχου υπηρεσιών). Το καινούργιο αυτό ραντεβού που μόλις καταχωρήθηκε στο σύστημα θα χρειαστεί να ενσωματωθεί και στο Google Calendar. Έτσι τρέχει μια διαδικασία η οποία προσθέτει αυτό το ραντεβού στην υπηρεσία της Google.

2. Εκτός όμως του ραντεβού που δημιουργήθηκε στο Easy!Appointments, θα χρειαστεί να ληφθούν και οι αλλαγές που έχουν γίνει στο Google Calendar. Για αυτόν τον λόγο είναι απαραίτητο να ανιχνευθούν όλα τα καταχωρημένα ραντεβού και να ελεγχθούν για τυχόν αλλαγές. Επειδή αυτό όμως μπορεί να γίνει αρκετά χρονοβόρο υπάρχει μια παράμετρος στο σύστημα του Easy!Appointments η οποία καθορίζει το χρονικό διάστημα στο παρελθόν και το μέλλον, για το οποίο θέλει ο χρήστης να εκτελείται ο συγχρονισμός. Επίσης τα ραντεβού που έχουν συγχρονιστεί με το Google Calendar έχουν κρατημένο το id της εγγραφής στο σύστημα της Google, έτσι ώστε να είναι δυνατό να ανιχνευθούν οι αλλαγές που έχουν γίνει από τον χρήστη. Έτσι αν για παράδειγμα ένας χρήστης διαγράψει ένα ραντεβού από το Google Calendar το οποίο ήταν συγχρονισμένο και στο Easy!Appointments, η διαδικασία του συγχρονισμού θα καταλάβει ότι το ραντεβού λείπει και έτσι θα το διαγράψει και από το σύστημα του Easy!Appointments.

Κεφάλαιο 3

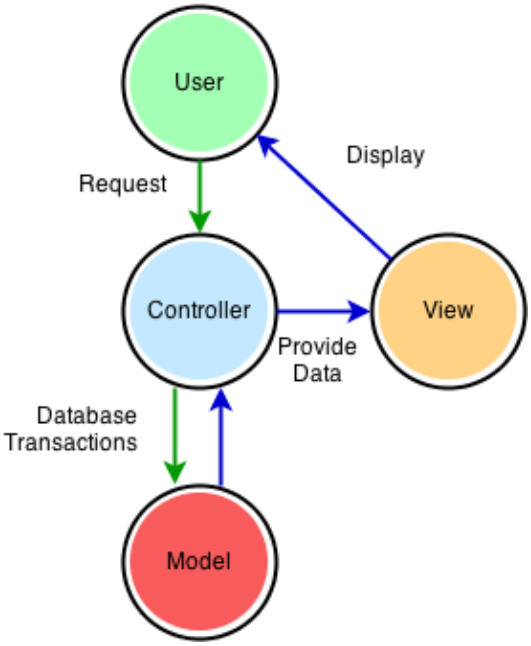
Εξωτερικά Εργαλεία

Εκτός του Calendar API και των βιβλιοθηκών που παρέχει η Google, έχουν χρησιμοποιηθεί και κάποια άλλα εργαλεία ανάπτυξης λογισμικού, τα οποία βοήθησαν στην άρτια και ποιοτικότερη παραγωγή του συστήματος κρατήσεων ραντεβού. Τα εργαλεία αυτά είναι όλα ανοιχτού κώδικα (open source) και έχουν στόχο να βοηθήσουν τον προγραμματιστή να επικεντρωθεί περισσότερο σε αυτό που έχει να κάνει και όχι τόσο στα τετριμμένα πράγματα τα οποία αποσπούν μεγάλο χρονικό διάστημα άσκοπα. Εν ολίγοις πρόκειται για ένα σύνολο από διάφορα framework τα οποία είναι πολύ χρήσιμα για οποιαδήποτε ανάπτυξη λογισμικού.

3.1 CodeIgniter

Το CodeIgniter είναι ένα PHP framework το οποίο έχει ως στόχο την αποδοτικότητα και την ταχύτητα, μιας και καταναλώνει πολύ λίγους υπολογιστικούς πόρους σε αντίθεση με άλλα βοηθητικά συστήματα του είδους του. Θετικό στοιχείο είναι ότι είναι πολύ απλό στην χρήση, δίνει την δυνατότητα στον προγραμματιστή να παραμετροποιήσει τον πυρήνα του καταπώς τον βολεύει και βασίζεται στην αρχιτεκτονική MVC (Model - View - Controller). Όντας έργο ανοιχτού λογισμικού, κατέχει μια μεγάλη κοινότητα που το υποστηρίζει και επίσης προσφέρει μια καλά ενημερωμένη γνωσιακή βάση, η οποία μπορεί να καθοδηγήσει τον προγραμματιστή στο πως θα χρησιμοποιήσει το framework.

Η αρχιτεκτονική MVC είναι η πλέον διαδεδομένη κυρίως στις διαδικτυακές εφαρμογές αφού αποσκοπεί στην καλύτερη οργάνωση και συντήρηση του κώδικα. Ουσια-



Διάγραμμα 3.1: Αρχιτεκτονική MVC

στικά πρόκειται για τον διαχωρισμό της εφαρμογής σε τρία μέρη:

1. Models: Περιέχουν συναρτήσεις και μεθόδους που αλληλεπιδρούν με την βάση δεδομένων. Χρησιμοποιούνται σε διάφορες περιπτώσεις από τους Controllers. Με αυτόν τον τρόπο επιτυγχάνεται η επαναχρησιμοποίηση ενός μέρος του κώδικα, κάτι το οποίο είναι πολύ σημαντικό στην αρχιτεκτονική ενός συστήματος.

2. Views: Τα views είναι τα κομμάτια κώδικα τα οποία παράγουν το αποτέλεσμα το οποίο βλέπει ο χρήστης κάθε φορά. Στόχος τους είναι απλώς να δείξουν και όχι να υπολογίσουν ή να φέρουν κάποια δεδομένα (αυτό είναι δουλειά των άλλων δυο τμημάτων της αρχιτεκτονικής). Κάθε φορά που χρειάζεται να αλλάξει κάτι στην εμφάνιση του συστήματος μπορεί να γίνει αλλαγή στο αντίστοιχο view χωρίς να επηρεαστούν τα άλλα συστήματα.

3. Controllers: Το μέρος αυτό του συστήματος αναλαμβάνει να οργανώσει τα άλλα δυο. Κάθε φορά που πρέπει να παραχθεί μια σελίδα, η διαδικασία θα ξεκινήσει από τον αντίστοιχο controller. Αυτός στην συνέχεια θα καλέσει τις απαραίτητες συναρτήσεις και θα παρέχει τα δεδομένα που απαιτεί το view για να εμφανιστεί σωστά η σελίδα.

Το σύστημα που υλοποιήθηκε χρησιμοποιεί το CodeIgniter για την κάλυψη των

βασικών εργασιών έτσι ώστε να υπάρχει η δομή MVC στον κώδικα. Επίσης γίνεται χρήση της ενσωματωμένης βιβλιοθήκης επικοινωνίας με την βάση δεδομένων. Εκτός αυτών των δύο, ο υπόλοιπος κώδικας έχει γραφεί κανονικά και τηρεί τις προϋποθέσεις της πτυχιακής εργασίας.

3.2 jQuery & jQuery UI

Ένα μεγάλο μέρος των σύγχρονων διαδικτυακών εφαρμογών βασίζει την λειτουργία του σε κώδικα Javascript έτσι ώστε να κάνει το λογισμικό πιο φιλικό προς τον χρήστη. Διάφορες βιβλιοθήκες έχουν δημιουργηθεί τα τελευταία χρόνια που στόχο έχουν την εξέλιξη και την αποδοτικότερη χρήση της γλώσσας javscript. Η πιο δημοφιλής από όλες αυτές τις βιβλιοθήκες είναι η βιβλιοθήκη jQuery η οποία συνοδεύεται απο το jQuery UI, ένα framework για την παραγωγή ελεκτηρίων (controls) στα οποία μπορεί ο κάθε χρήστης να εκτελέσει διάφορες ενέργειες. Το jQuery έχει καταφέρει να απλοποιήσει την συγγραφή Javascript κώδικα και επιπλέον παρέχει στην διάθεση του προγραμματιστή έτοιμες ρουτίνες animation και διαφόρων άλλων ενεργειών, οι οποίες διαφορετικά θα καταλάμβαναν αρκετό χρόνο για την υλοποίηση τους. Πρόκειται για μια βιβλιοθήκη ανοιχτού λογισμικού η οποία υποστηρίζεται από μια πολύ μεγάλη κοινότητα προγραμματιστών, οι οποίοι δημιουργούν μάλιστα πολλά πρόσθετα (plugins) με λειτουργίες οι οποίες δεν είναι διαθέσιμες στην βασική βιβλιοθήκη.

Η βιβλιοθήκη jQuery ουσιαστικά λειτουργεί σαν ένα επίπεδο πάνω από την Javascript, βοηθώντας τον προγραμματιστή να γράψει διάφορες δομές κώδικα πιο γρήγορα και οργανωμένα. Ο κώδικας που γράφεται είναι πάλι Javascript οπότε είναι πολύ εύκολο στον καθένα να χρησιμοποιήσει την βιβλιοθήκη. Στόχος της είναι η πιο εύκολη περιήγηση στα αντικείμενα μιας σελίδας (DOM elements), η εύκολη δημιουργία εφέ κινήσεων τα οποία προσδίδουν πολύ αισθητικά σε μια ιστοσελίδα, η ευκολότερη χρήση της τεχνολογίας AJAX, μιας τεχνολογίας η οποία χρησιμοποιείται όλο και περισσότερο από τα σύγχρονα συστήματα.

Το σύστημα που παράχθηκε χρησιμοποιεί αυτήν την βιβλιοθήκη για τον προγραμματισμό του client-side μέρους της εφαρμογής. Ανάλογα με την κάθε περίσταση, μερικές φορές είναι αποδοτικότερο και χρησιμότερο να χρησιμοποιηθεί Javascript έναντι της PHP και για αυτόν τον λόγο το jQuery Framework βοηθάει στην απόδοση που έχει

ο κώδικας της εφαρμογής.

3.3 Bootstrap

Το Bootstrap είναι ένα ολοκληρωμένο CSS Framework, με την προσθήκη κάποιων βιβλιοθηκών Javascript έτσι ώστε να προσφέρει μερικές επιπλέον δυνατότητές. Έχει κατασκευαστεί από την εταιρεία πίσω από την σελίδα κοινωνικής δικτύωσης Twitter και χρησιμοποιείται από αυτό ως βάση για την υλοποίηση του. Η εταιρεία έχει διάθεση το framework ως ανοιχτό λογισμικό και οι προγραμματιστές μπορούν να το χρησιμοποιήσουν ελεύθερα στις σελίδες τους. Η ίδια η εταιρεία επωφελείται μέσω της συμμετοχής της κοινότητας για να ανάπτυξη περαιτέρω αυτό το framework.

Το framework αυτό περιέχει έτοιμο κώδικα CSS ο οποίος ακολουθεί την μεθοδολογία παραγωγής responsive ιστοσελίδων. Με την έννοια αυτή εννοείται ότι η μορφοποίηση των σελίδων στοχεύει στο να είναι συμβατή με οποιαδήποτε συσκευή, λειτουργικό σύστημα και περιηγητή ιστού. Είναι πολύ σημαντικό για την ανάπτυξη μιας εφαρμογής, το να είναι διαθέσιμη σε οποιοδήποτε μηχάνημα υποστηρίζει τα τελευταία standards του ιστού. Απαρτίζεται από κανονικό CSS και Javascript το οποίο μάλιστα χρησιμοποιεί την βιβλιοθήκη jQuery για την υλοποίηση διαφόρων τεχνικών (πχ εμφάνιση παραθύρου διαλόγου κτλ).

Στην παραγωγή του συστήματος κρατήσεων ραντεβού χρησιμοποιήθηκε ως η βάση για τη μορφοποίηση και κάποιες μέθοδοι Javascript φάνηκαν αρκετά χρήσιμα σε κάποια σημεία. Στην συνέχεια με βάση αυτά γράφτηκαν ξεχωριστά αρχεία CSS και Javascript τα οποία αποτελούν το τελικό αποτέλεσμα που έπρεπε να επιτευχθεί.

Κεφάλαιο 4

Σενάρια Χρήσης

Το κεφάλαιο αυτό έχει ως στόχο να δώσει μια τυπική περιγραφή της χρήσης της εφαρμογής, για όλους τους διαθέσιμους ρόλους των χρηστών της, έτσι ώστε να γίνει περισσότερο κατανοητός ο τρόπος με τον οποίον θα λειτουργεί το σύστημα κρατήσεων ραντεβού.

4.1 Σενάριο χρήσης διαχειριστή

Μετά από αρκετό καιρό χρήσης του Easy!Appointments η εταιρεία προσθέτει μια νέα υπηρεσία στο ενεργητικό της και για τον σκοπό αυτό ανοίγει ένα νέο τμήμα υπαλλήλων. Ο διαχειριστής του συστήματος πρέπει να ενημερώσει την εφαρμογή και να προσθέσει την νέα υπηρεσία, καθώς και τους νέους πάροχους υπηρεσιών, έτσι ώστε να μπορούν οι πελάτες να κλείνουν ραντεβού μαζί τους από εδώ και πέρα. Εφόσον γίνει αυτό, οι πελάτες θα μπορούν να επιλέξουν τις αντίστοιχες εγγραφές από την φόρμα κράτησης ραντεβού.

4.2 Σενάριο χρήσης πάροχου υπηρεσιών

Ο πάροχος υπηρεσιών της εφαρμογής λαμβάνει μια ειδοποίηση από την εφαρμογή (email) ότι έχει γίνει μια κράτηση για ραντεβού. Βλέποντας τα στοιχεία της κράτησης και την ημερομηνία αποφασίζει ότι δεν θα μπορέσει να είναι εκείνη την στιγμή διαθέσιμος, οπότε συνδέεται στην εφαρμογή και αλλάζει την ημερομηνία του ραντεβού. Αμέσως μετά πηγαίνει στο πρόγραμμά του και ενημερώνει την χρονική στιγμή στην

οποία δεν θα είναι διαθέσιμος, έτσι ώστε να μην μπορούν πλέον οι πελάτες να κάνουν κρατήσεις σε εκείνη την χρονική περίοδο. Στην συνέχεια αποστέλλεται ειδοποίηση στον πελάτη και αυτός μπορεί να κρίνει αν τον βολεύει η νέα ημερομηνία. Αν όχι θα πρέπει να ακυρώσει το ραντεβού και να το ξανά-προσθέσει σε κάποια άλλη χρονική στιγμή.

4.3 Σενάριο χρήσης πελάτη

Ο πελάτης ενδιαφέρεται να κλείσει ραντεβού στην επιχείρηση για μια συγκεκριμένη υπηρεσία. Πηγαίνει στην σελίδα της επιχείρησης και βλέπει το πλάνο, αφού έχει επιλέξει ποια υπηρεσία και ποιόν υπάλληλο προτιμάει. Στην συνέχεια επιλέγει μια χρονική στιγμή που τον βολεύει και την κατοχυρώνει. Η διαδικασία ολοκληρώνεται με την αποστολή ενός email προς τον πελάτη, το οποίο περιέχει όλες τις πληροφορίες του ραντεβού, έτσι ώστε να είναι πάντα προσβάσιμες. Σε αυτό το email περιέχεται και ένας σύνδεσμος ο οποίος επιτρέπει στον πελάτη να πραγματοποιήσει αλλαγές στο ραντεβού. Από την στιγμή αυτήν και μετά το ραντεβού έχει κατοχυρωθεί και ο πελάτης θα ενημερώνεται για οποιαδήποτε αλλαγή μπορεί να γίνει στο ραντεβού του με email.

4.4 Σενάριο χρήσης γραμματέα

Ένας από τους πάροχους υπηρεσίας έχει κλειστεί εντελώς από ραντεβού και δεν μπορεί να δεχτεί άλλα για αυτήν την εβδομάδα. Ένας άλλος πάροχος προσφέρεται να βοηθήσει και έτσι κάποια ραντεβού πρέπει να μεταφερθούν στο ημερολογιακό πλάνο του δεύτερου πάροχου. Την διαδικασία αυτήν θα πρέπει να την αναλάβει η γραμματεία γιατί όλοι οι άλλοι είναι πολύ απασχολημένοι με το να εξυπηρετήσουν τους πελάτες τους.

Κεφάλαιο 5

Περιπτώσεις Χρήσης

Σε αυτό το κεφάλαιο θα γίνει η αναλυτική περιγραφή των περιπτώσεων χρήσης του συστήματος που υλοποιήθηκε. Θα περιγραφούν τόσο η βασική ροή όσο και οι εναλλακτικές ροές για όλες τις περιπτώσεις χρήσης. Το κεφάλαιο χωρίζεται σε τμήματα ανάλογα με τον ρόλο (actor) της εφαρμογής στον οποίο ανήκουν.

5.1 Πελάτης

5.1.1 Κράτηση ραντεβού

Η βασικότερη περίπτωση χρήσης της εφαρμογής είναι η διαδικασία της κράτησης ραντεβού του πελάτη με έναν πάροχο υπηρεσίας για την υπηρεσία που τον ενδιαφέρει. Πρόκειται για την κυριότερη περίπτωση χρήσης, μιας και το σύστημα έχει ως στόχο την ευκολότερη διαχείριση των ραντεβού με τους πελάτες.

ΒΑΣΙΚΗ ΡΟΗ

Ο χρήστης μπαίνει στην σελίδα κράτησης ραντεβού και επιλέγει την υπηρεσία και τον πάροχο που τον ενδιαφέρει. Στην συνέχεια θα χρειαστεί να επιλέξει μια από τις διαθέσιμες ημερομηνίες και ώρες για να κλείσει το ραντεβού του. Αφού γίνει και αυτό θα πρέπει να συμπληρώσει τα στοιχεία του στην φόρμα που θα εμφανιστεί έτσι ώστε να μπορέσει η εταιρεία να έρθει σε επαφή μαζί του αν χρειαστεί. Τέλος ένα email θα σταλθεί πίσω στον πελάτη ότι το ραντεβού του έχει καταχωρηθεί με επιτυχία. Σε αυτό το email θα εμπεριέχεται και ένα link το οποίο θα του επιτρέπει να κάνει τροποποιήσει ή και να ακυρώσει το συγκεκριμένο ραντεβού.

ΕΝΑΛΛΑΚΤΙΚΕΣ ΡΟΕΣ

- Αν ο πελάτης αργήσει να επιλέξει ημερομηνία και στο ενδιάμεσο τον προλάβει ένας άλλος, θα πρέπει να επιστραφεί μήνυμα το οποίο θα τον προτρέψει να βρει άλλη ημερομηνία και ώρα για το ραντεβού του.

- Όταν ο πελάτης συμπληρώσει τα στοιχεία του και αφήσει κενό ένα πεδίο το οποίο είναι υποχρεωτικό για να ολοκληρωθεί η διαδικασία, θα εμφανιστεί μήνυμα το οποίο θα τον προτρέψει να συμπληρώσει όλα τα υποχρεωτικά πεδία.

5.1.2 Επεξεργασία - ακύρωση ραντεβού

Εφόσον καταχωρηθεί ένα ραντεβού είναι πολύ σημαντικό να μπορέσει και να τροποποιηθεί με κάποιον τρόπο. Το σύστημα από την στιγμή που καταχωρεί ένα ραντεβού κρατάει και τα στοιχεία του πελάτη σε μια εγγραφή. Παρ' όλα αυτά δεν θα ήταν καλό να αναγκάζει τον πελάτη να δημιουργεί νέο χρήστη (με username και password) έτσι ώστε να μπορέσει να κάνει αλλαγές. Κάτι τέτοιο θα μείωνε την αποδοτικότητα της εφαρμογής μιας και προσθέτει ένα επιπλέον βήμα στην όλη διαδικασία, το οποίο μάλιστα θεωρείται εκνευριστικό αφού ένας μέσος χρήστης του διαδυκτίου θα χρειαστεί να δημιουργήσει δεκάδες λογαριασμούς σε διάφορες ιστοσελίδες. Λαμβάνοντας αυτά υπόψιν για να μπορέσει ο πελάτης να πραγματοποιήσει αλλαγές ή και ακύρωση σε κάποιο ραντεβού του θα ακολουθεί έναν μοναδικό σύνδεσμο ο οποίος θα του έρχεται με email.

ΒΑΣΙΚΗ ΡΟΗ

Ο χρήστης ολοκληρώνει την διαδικασία κράτησης ραντεβού. Σε αυτήν την διαδικασία έχει ήδη δώσει το email του, οπότε του έρχεται ένα email το οποίο περιέχει τις πληροφορίες του ραντεβού στο οποίο έχει κάνει την κράτηση και μαζί έναν σύνδεσμο, ο οποίος επιτρέπει στον χρήστη να πραγματοποιήσει αλλαγές στο συγκεκριμένο ραντεβού ή και να το ακυρώσει. Αφού ο χρήστης ακολουθήσει τον σύνδεσμο θα βρεθεί σε μια σελίδα η οποία θα περιέχει τις πληροφορίες του ραντεβού και θα του επιτρέπει να κάνει διάφορες αλλαγές. Όταν ολοκληρώσει την διαδικασία θα πατάει ένα κουμπί το οποίο θα αποθηκεύει τις αλλαγές και ένα νέο email θα έρχεται πάλι στον χρήστη αλλά και στον συγκεκριμένο πάροχο ότι έχουν πραγματοποιηθεί αλλαγές στο πλάνο του.

ΕΝΑΛΛΑΚΤΙΚΕΣ ΡΟΕΣ

- Ο χρήστης μπορεί εν τέλη να μην θέλει να αποθηκεύσει τις αλλαγές του και έτσι να κλείσει την σελίδα.
- Ο διαχειριστής του συστήματος μπορεί να έχει ορίσει ένα χρονικό περιθώριο πριν το ραντεβού, στο οποίο δεν επιτρέπεται να γίνονται αλλαγές (λόγω σταθερότητας του πλάνου). Αν ο χρήστης βρίσκεται μέσα σε αυτό το περιθώριο τότε θα εμφανιστεί μήνυμα το οποίο θα τον ενημερώνει για τον λόγο τον οποίο δεν μπορεί να πραγματοποιήσει αλλαγές στο ραντεβού του.

5.2 Πάροχος Υπηρεσιών

5.2.1 Συγχρονισμός πλάνου με το Google Calendar

Βασικό στοιχείο για την χρησιμότητα και την απόδοση του συστήματος είναι η διαχείριση των δεδομένων να γίνεται από πολλά συστήματα. Κάτι τέτοιο μπορεί να επιτεφθχεί με τον συγρονισμό των ραντεβού με το Google Calendar API. Σε αυτό ο χρήστης θα μπορεί να πραγματοποιεί αλλαγές στο πλάνο του μέσω του Google Calendar και αυτές να εφαρμόζονται και στο σύστημα κρατήσεων ραντεβού, κάνοντας έτσι την εργασία του πολύ εύκολη.

ΒΑΣΙΚΗ ΡΟΗ

Ο χρήστης βλέπει το πλάνο του μέσω της υπηρεσίας Google Calendar και προσθέτει ένα συμβάν κατά την διάρκεια του οποίο δεν είναι διαθέσιμος. Έπειτα από λίγο τρέχει χειροκίνητα τον συγχρονισμό από το Easy!Appointments και αυτό ανακαλύπτει ότι υπάρχει ένα νέο συμβάν στο Google Calendar το οποίο δεν είναι καταχωρημένο στην βάση δεδομένων του. Αμέσως μετά παίρνει τα στοιχεία του νέου συμβάντος μέσω του API που παρέχει η Google και το αποθηκεύει στην βάση δεδομένων έτσι ώστε να μην είναι διαθέσιμος ο πάροχος την συγκεκριμένη χρονική στιγμή. Την επόμενη φορά που θα πάει ένας πελάτης να κλείσει ραντεβού με τον συγκεκριμένο πάροχο θα δει ότι το συγκεκριμένο χρονικό διάστημα δεν είναι διαθέσιμο.

ΕΝΑΛΛΑΚΤΙΚΕΣ ΡΟΕΣ

- Υπάρχει η περίπτωση στην οποία ο πάροχος έχει πραγματοποιήσει αλλαγές στο Google Calendar και στο Easy!Appointments ταυτόχρονα, χωρίς να έχει τρέξει

η διαδικασία του συγχρονισμού. Σε αυτήν την περίπτωση υπάρχει μεγάλη πιθανότητα να δημιουργηθεί κάποια σύγκρουση (conflict) και να υπάρχουν δυο συμβάντα τα οποία να ανταποκρίνονται στην ίδια χρονική περίοδο. Σε αυτήν την κατάσταση ο χρήστης είναι υπεύθυνος να λύσει την σύγκρουση μεταξύ των δύο συμβάντων και να φέρει το πλάνο του στην σωστή του μορφή.

- Πιθανό είναι επίσης να γίνει μια αλλαγή σε ένα συγχρονισμένο συμβάν στο Google Calendar το οποίο όμως να έχει αλλαχθεί και στο Easy!Appointments. Σε αυτήν την περίπτωση θεωρείται ότι υπερισχύει η αλλαγή που έχει γίνει στο Easy!Appointmets διότι δεν υπάρχει η δυνατότητα να ελεγχθεί και στα δύο συστήματα το πότε (χρονική στιγμή) έχει γίνει η τροποποίηση.

5.2.2 Διαχείριση ραντεβού

Όπως και ο πελάτης, έτσι και ο πάροχος υπηρεσιών θα πρέπει να έχει την δυνατότητα να διαχειρίζεται τα ραντεβού του (εφόσον του έχει δοθεί το δικαίωμα από τον διαχειριστή). Με αυτόν τον τρόπο θα έχει την δυνατότητα να πραγματοποιεί αλλαγές στο ημερολόγιο του, να αλλάζει την ημερομηνία των ραντεβού του και να καθορίζει το πλάνο του καταπώς τον βολεύει. Το Easy!Appointments εμφανίζει όλα τα ραντεβού σε ημερολόγιο, στο οποίο ο χρήστης μπορεί να περιηγηθεί χρονικά.

ΒΑΣΙΚΗ ΡΟΗ

Ο χρήστης ενημερώνεται από τον προϊστάμενο του ότι θα πρέπει να πραγματοποιήσει κάποια εργασία σε ένα χρονικό διάστημα στο οποίο υπάρχουν καταχωρημένα ραντεβού με πελάτες της εταιρείας. Κάποια από τα ραντεβού μπορεί να τα αναλάβει κάποιος άλλος πάροχος και για αυτόν τον λόγο ο χρήστης επεξεργάζεται αυτά τα ραντεβού και αλλάζει τον πάροχο τους έτσι ώστε να μετατεθούν στο πλάνου του αντίστοιχου χρήστη. Όσα περισσέψουν θα χρειαστεί να τα μεταφέρει χρονικά ή να τα διαγράψει προτείνοντας στους πελάτες να κλείσουν κάποια άλλη στιγμή.

ΕΝΑΛΛΑΚΤΙΚΕΣ ΡΟΕΣ

- Καθώς ο πάροχος υπηρεσιών επεξεργάζεται ένα ραντεβού μπορεί να αποφασίσει ότι δεν θέλει να αποθηκεύσει κάποιο ραντεβού και έτσι να πατήσει το κουμπί ακύρωσης και να μην επιρρεάσει το ραντεβού.

- Σε αντίθεση με τον πελάτη, ένας πάροχος υπηρεσιών μπορεί να αλλάξει την διάρκεια ενός ραντεβού, ανεξάρτητα από το πόση ώρα διαρκεί το ραντεβού του. Οπότε στην φόρμα επεξεργασίας ενός ραντεβού υπάρχει η δυνατότητα επιλογής ημερομηνίας και ώρας έναρξης και τερματισμού του ραντεβού.

5.2.3 Λήψη ειδοποιήσεων από το σύστημα

Κάθε φορά που πραγματοποιείται μια ενέργεια που αφορά κάποιο ραντεβού στο σύστημα ο πάροχος υπηρεσίας θα ενημερώνεται με email (εκτός και αν απενεργοποιήσει αυτήν την ρύθμιση). Έτσι θα είναι πάντα ενήμερος σχετικά με την κατάσταση των ραντεβού του. Σε αυτά τα μηνύματα θα περιέχεται και ένα μοναδικό link το οποίο θα δίνει την δυνατότητα στον πάροχο να πραγματοποίηση αλλαγές γρήγορα στο ραντεβού που τον ενδιαφέρει.

ΒΑΣΙΚΗ ΡΟΗ

Ο πάροχος υπηρεσιών μπαίνει στο σύστημα και πραγματοποιεί αλλαγές σε ένα ήδη καταχωρημένο ραντεβού. Με την αποθήκευση των αλλαγών αυτός και ο πελάτης θα λάβουν ένα email στο οποίο θα φαίνονται τα νέα στοιχεία του ραντεβού. Με αυτόν τον τρόπο μπορεί να επαληθευθεί ότι οι αλλαγές που έγιναν είναι σωστές και επίσης δίνεται η δυνατότητα στον χρήστη να χρησιμοποιήσει το link του ραντεβού για να πραγματοποιήσει άλλες αλλαγές, αν αυτό είναι απαραίτητο.

5.2.4 Διαχείριση Πελατών

Από την στιγμή που ένας πάροχος θα έχει το δικαίωμα να επεξεργαστεί τις πληροφορίες ενός ραντεβού θα μπορεί να επεξεργάζεται και τα στοιχεία των πελατών που υπάγονται σε αυτό. Με αυτήν την δυνατότητα θα μπορεί να είναι σε θέση να ενημερώσει κάποια πεδία στην εγγραφή του πελάτη τα οποία είναι πιθανώς λανθασμένα ή και να προσθέσει νέες πληροφορίες.

ΒΑΣΙΚΗ ΡΟΗ

Ο πάροχος μπαίνει στο σύστημα και επιλέγει να επεξεργαστεί κάποιο ραντεβού. Στην οθόνη επεξεργασίας του ραντεβού θα μπορέσει να τροποποιήσει και τα στοιχεία των πελατών. Όταν είναι έτοιμος μπορεί να αποθηκεύσει τις αλλαγές στην βάση δεδομένων.

ΕΝΑΛΛΑΚΤΙΚΕΣ ΡΟΕΣ

Ο πάροχος μπορεί να αλλάξει τα στοιχεία ενός πελάτη χωρίς να πραγματοποιήσει αλλαγές στις πληροφορίες του ραντεβού του ίδιου.

5.3 Γραμματέας

5.3.1 Διαχείριση ραντεβού

Ο χρήστης γραμματέας μπορεί να πραγματοποιήσει αλλαγές στα ραντεβού ενός ή περισσοτέρων πάροχων υπηρεσίας. Με αυτόν τον τρόπο μια εταιρεία μπορεί να αναθέσει όλη την διαχείριση των ραντεβού σε ένα άτομο και οι πάροχοι απλώς να βλέπουν τα ραντεβού τους στο πλάνο.

ΒΑΣΙΚΗ ΡΟΗ

Ο χρήστης γραμματέας δέχεται τηλεφώνημα από κάποιον πελάτη ο οποίος επιθυμεί να αλλάξει την ώρα του ραντεβού του αλλά δεν έχει σύνδεση με το διαδίκτυο στο σημείο που βρίσκεται. Έτσι ο γραμματέας πραγματοποιεί την αλλαγή του ραντεβού καταπώς μπορεί να βολεύει τόσο τον πάροχο υπηρεσίας όσο και τον πελάτη. Όταν τελειώσει μπορεί να αποθηκεύσει τις αλλαγές στο σύστημα.

5.3.2 Διαχείριση πελατών

Εκτός των ραντεβού, ο χρήστης γραμματέας είναι σε θέση να πραγματοποιήσει αλλαγές και στα στοιχεία των πελατών, οργανώνοντας και κρατώντας πλήρη πελατολόγιο για την επιχείριση. Μελλοντικά θα είναι σε θέση να βρίσκει πολύ εύκολα οποιονδήποτε πελάτη της επιχείρισης και να βλέπει τα στοιχεία του.

ΒΑΣΙΚΗ ΡΟΗ

Ο γραμματέας πηγαίνει στο backend στην σελίδα τον πελατών και φιλτράρει τον πελάτη που τον ενδιαφέρει και αμέσως βλέπει τα στοιχεία του. Πολύ εύκολα μπορεί να πραγματοποιήσει αλλαγές, αλλά και να τον διαγράψει από το σύστημα.

ΕΝΑΛΛΑΚΤΙΚΕΣ ΡΟΕΣ

- Ο γραμματέας μπορεί να επεξεργαστεί τα στοιχεία ενός πελάτη και την στιγμή που επεξεργάζεται ένα ραντεβού. Παρόλα αυτά σε αυτήν την περίπτωση κυρίαρχη οντότητα είναι το ραντεβού και έτσι ο χρήστης δεν έχει την δυνατότητα να

φιλτράρει τους πελάτες καταπώς θέλει. Επίσης ένας πελάτης μπορεί να διαγραφεί μόνο όταν διαγραφεί και το ραντεβού του.

5.4 Διαχειριστής

5.4.1 Εγκατάσταση της εφαρμογής

Αυτή η περίπτωση χρήσης περιλαμβάνει την ρύθμιση του server όπου θα τρέξει το Easy!Appointments και την δημιουργία του λογαριασμού του διαχειριστή. Επίσης μπορεί να καθοριστούν και τα κλειδιά που απαιτούνται για την χρήση του Google Calendar API.

ΒΑΣΙΚΗ ΡΟΗ

Ο διαχειριστής τοποθετεί τα αρχεία της εφαρμογής στον server του και πηγαίνει με τον περιηγητή στην διεύθυνση που δείχνει τον κύριο κατάλογο. Η εφαρμογή καταλαβαίνει ότι δεν έχει ακόμα πραγματοποιηθεί εγκατάσταση και μεταφέρει τον χρήστη στην διαδικασία εγκατάστασης. Σε αυτήν την διαδικασία ο διαχειριστής θα πρέπει να ορίσει σημαντικά στοιχεία που αφορούν την λειτουργία του συστήματος και τα στοιχεία του λογαριασμού του. Όλα αυτά τα στοιχεία βέβαια θα είναι διαθέσιμα προς επεξεργασία από την αντίστοιχη σελίδα στο backend.

5.4.2 Παραμετροποίηση της εφαρμογής

Για να μπορέσει να λειτουργήσει η εφαρμογή σύμφωνα με την μορφή της επιχείρησης θα χρειαστεί να παραμετροποίηση από τον διαχειριστή. Η παραμετροποίηση περιλαμβάνει τα ωράρια λειτουργίας της επιχείρησης, την διαχείριση των υπηρεσιών που θα είναι διαθέσιμες προς το κοινό, καθώς και την διαχείριση των πάροχων υπηρεσιών.

ΒΑΣΙΚΗ ΡΟΗ

Ο διαχειριστής εισέρχεται στο backend μέρος της εφαρμογής και επιλέγει την σελίδα των ρυθμίσεων. Εκεί έχει την δυνατότητα να θέσει τις τιμές σε διάφορες παραμέτρους, οι οποίες καθορίζουν τον τρόπο με τον οποίο λειτουργεί το σύστημα. Στόχος είναι αυτός να συμβαδίζει με τον τρόπο λειτουργίας της επιχείρισης έτσι ώστε να μπορεί η εταιρεία να επωφεληθεί από το Easy!Appointments.

5.4.3 Διαχείριση ραντεβού

Ο διαχειριστής ως χρήστης έχει πρόσβαση σε όλες τις πληροφορίες του συστήματος. Έτσι μπορεί να μεταβάλει και να προσθέσει ραντεβού στο σύστημα σαν να ήταν ένας πάροχος υπηρεσιών ή ένας χρήστης γραμματέας.

ΒΑΣΙΚΗ ΡΟΗ

Ο διαχειριστής συνδέεται στο backend μέρος της εφαρμογής και πηγαίνει στην σελίδα του ημερολογίου. Εκεί μπορεί να δει τα ραντεβού όλων των πάροχων υπηρεσιών και να πραγματοποιήσει αλλαγές σε αυτά. Με το που αποθηκευτούν οι αλλαγές σε ένα ραντεβού ο πελάτης και ο πάροχος θα ενημερωθούν με email, όπως θα γινόταν δηλαδή αν την επεξεργασία την έκανε ένας πάροχος.

5.4.4 Διαχείριση χρηστών

Τις υπηρεσίες που προσφέρει η εταιρεία τις αναλαμβάνουν κάποιοι υπάλληλοι (ή ομάδες υπαλλήλων), οι οποίοι αναφέρονται στο σύστημα ως πάροχοι υπηρεσιών. Τα στοιχεία τους, τις αρμοδιότητές τους και τα δικαιώματα μέσα στο σύστημα τα ορίζει μόνο ο διαχειριστής του συστήματος, από το backend μέρος της εφαρμογής. Επίσης είναι δυνατή η διαχείριση των διαχειριστών, γραμματειών και πελατών του συστήματος.

ΒΑΣΙΚΗ ΡΟΗ

Ο διαχειριστής της εφαρμογής συνδέεται στο backend μέρος του συστήματος και πηγαίνει στην σελίδα των πάροχων υπηρεσίας. Εκεί βλέπει μια λίστα με τους ήδη καταχωρημένους πάροχους υπηρεσιών. Αν θέλει μπορεί να προσθέσει έναν καινούργιο πάροχο, ή να επεξεργαστεί και να διαγράψει κάποιον πάροχο που υπάρχει ήδη στην βάση δεδομένων.

5.4.5 Διαχείριση υπηρεσιών

Οι πελάτες που θα επισκέπτονται τον ιστότοπο του Easy!Appointments της επιχείρησης θα κλείνουν ραντεβού για συγκεκριμένες υπηρεσίες. Το ποιες υπηρεσίες θα είναι διαθέσιμες και ποιοι πάροχοι υπηρεσιών μπορούν να εξυπηρετήσουν τι, το διαχειρίζεται ο διαχειριστής του συστήματος. Αποτελεί υποπερίπτωση χρήσης της παραμετροποίησης της εφαρμογής.

Κεφάλαιο 6

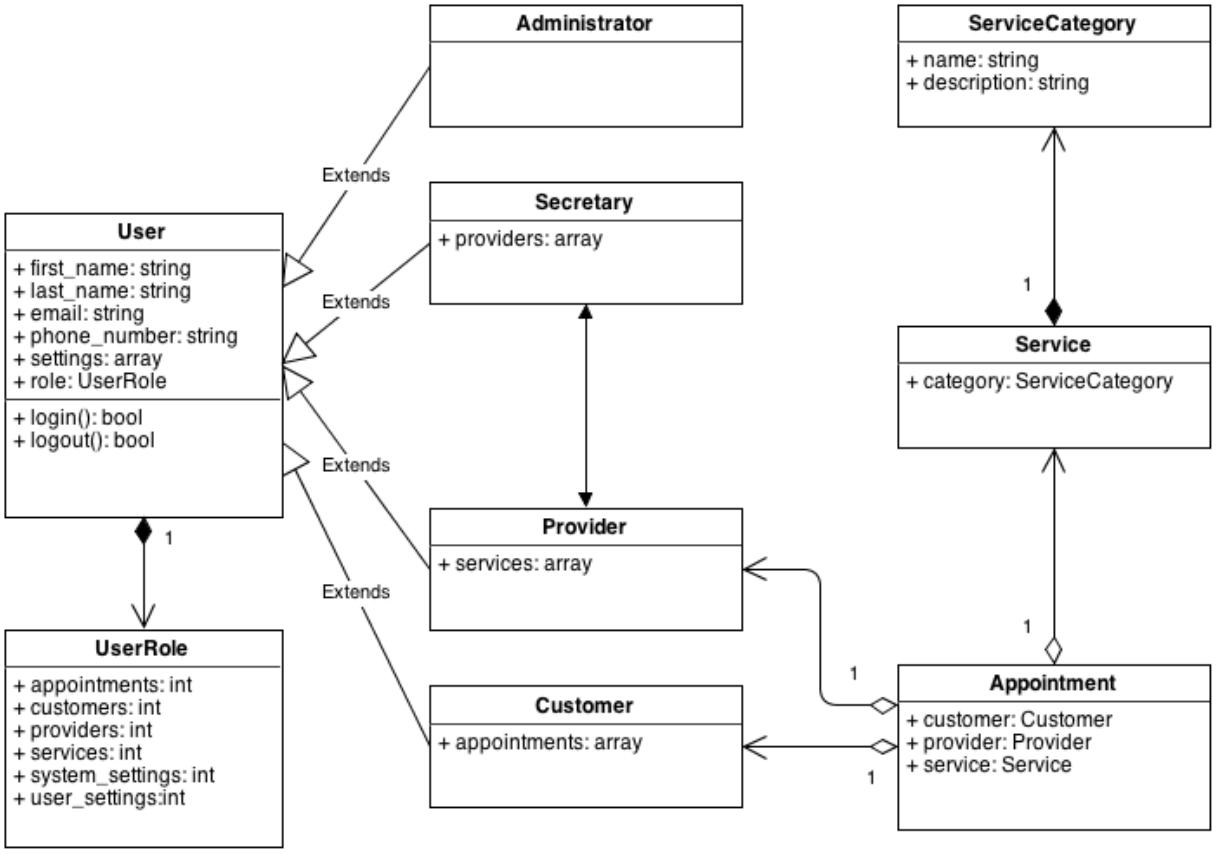
Σχεδίαση & Υλοποίηση

Σε αυτό το κεφάλαιο γίνεται ανάλυση του συστήματος στα επιμέρους μέρη που το απαρτίζουν και περιγράφεται η διαδικασία της υλοποίησης τους. Επεξηγούνται επίσης τα σημαντικότερα σημεία στον κώδικα και οι αλγόριθμοι που χρησιμοποιούνται για την επίλυση των κυριότερων λειτουργιών. Έχουν συμπεριληφθεί τμήματα κώδικα αλλά και διαγράμματα τα οποία βοηθούν στην κατανόηση των λύσεων που έχουν χρησιμοποιηθεί.

6.1 Ανάλυση δεδομένων

Το κυριότερο πρόβλημα που προσπαθεί να λύσει το σύστημα είναι η κράτηση και η διαχείριση ραντεβού από μια επιχείριση. Σε αυτήν την περίπτωση χρήσης έχει επικεντρωθεί η σχεδίαση και η υλοποίηση του συστήματος το οποίο περιέχει και άλλες δυνατότητες οι οποίες μπορούν όμως να θεωρηθούν λιγότερο σημαντικές. Έχοντας υπόψιν την έννοια ”ραντεβού” ως την κύρια οντότητα της εφαρμογής, σχεδιάστηκε το παρακάτω μοντέλο το οποίο διευκρινίζει τις σχέσεις των οντοτήτων του συστήματος μεταξύ τους.

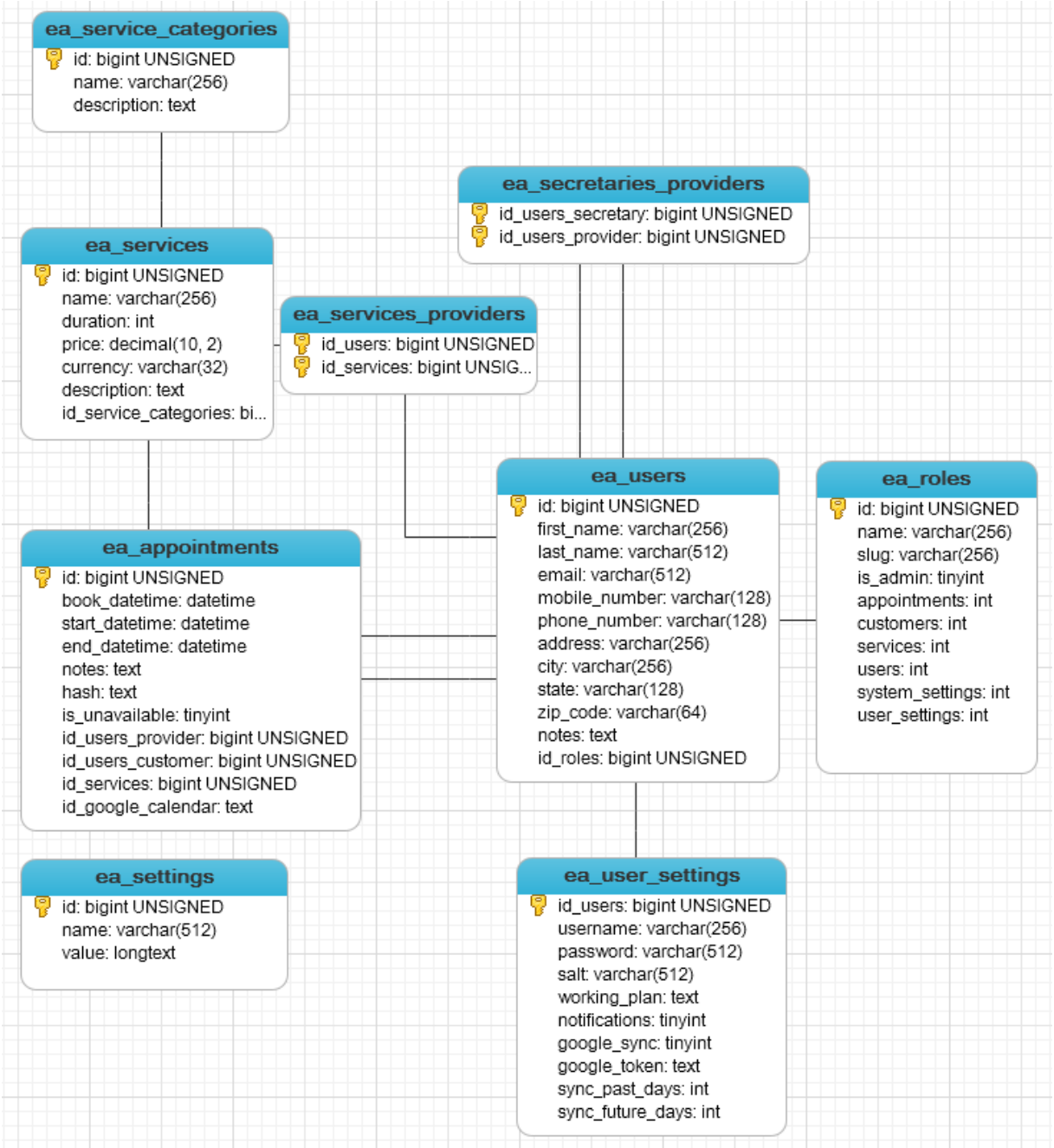
Με βάση αυτό το σχεδιάγραμμα μπορεί πολύ εύκολα να προκύψει και το σχεδιακό μοντέλο της βάσης δεδομένων, δεδομένου ότι έχουμε και τις οντότητες, αλλά και τις σχέσεις μεταξύ τους. Όλοι οι χρήστες κληρονομούν την συμπεριφορά τους από μια οντότητα (User) και επιπρόσθετα κατέχουν διάφορες ιδιότητες που είναι αναγκαίες για τον ρόλο τους μέσα στην εφαρμογή. Για παράδειγμα ο χρήστης γραμματέας (Secretary) περιέχει έναν πίνακα από πάροχους (Providers) τους οποίους μπορεί να διαχειριστεί,



Διάγραμμα 6.1: Domain model του συστήματος

ή ένα ραντεβού είναι ξεκάθαρο ότι περιέχει στην πληροφορία του έναν πελάτη, έναν πάροχο και μια υπηρεσία.

Για την διαχείριση των δεδομένων της βάσης δημιουργήθηκαν ειδικές κλάσεις (models) οι οποίες περιέχουν μεθόδους που χρησιμοποιούνται από τους controllers του συστήματος. Το CodeIgniter δίνει στον προγραμματιστή ένα δικό του μέσο επικοινωνίας με την βάση δεδομένων, το οποίο είναι ένα πολύ ισχυρό και ευέλικτο εργαλείο. Η επονομαζόμενη Database Class του CodeIgniter επιτρέπει στον προγραμματιστεί να εκτελεί ερωτήματα προς την βάση, να παράγει αποτελέσματα και να τα αναλύει σε ξεχωριστές εγγραφές, να κρατάει στην μνήμη ερωτήματα για γρηγορότερη ανταπόκριση (query caching) και κυριότερο την κλάση Active Record. Η κλάση αυτή έχει έναν δικό της τρόπο για την εκτέλεση των ερωτημάτων προς την βάση. Όλα τα τμήματα ενός τυπικού ερωτήματος είναι μέθοδοι, οι οποίες χρησιμοποιούνται από τον προγραμματιστή ως το μέσο επικοινωνίας με την βάση δεδομένων. Το θετικό είναι ότι ανεξαρτήτως τον τύπο της βάσης η κλάση αυτή λειτουργεί με τον ίδιο τρόπο (MySQL, PostGre, MSSQL κτλ). Η τεχνική αυτή λέγεται Active Record Database Pattern και έχει να κάνει με την



Διάγραμμα 6.2: Σχεσιακό μοντέλο της βάσης δεδομένων (ER).

αλλαγή adapter στην κλάση ανάλογα με τον τύπο της βάσης. Σε κάθε περίπτωση όμως ο τρόπος λειτουργίας της είναι ο ίδιος. Στο παρακάτω τμήμα κώδικα αναφέρεται ένα παράδειγμα για το πως μπορεί να βρεθεί το αναγνωριστικό μιας εγγραφής χρησιμοποιώντας ως κλειδί την διεύθυνση email.

```
1 <?php
2 /**
3  * Find the database record id of an admin user.
```

```
4   *
5   * @param array $admin Contains the admin data. The 'email'
6   * value is required in order to find the record id.
7   * @return int Returns the record id
8   * @throws Exception When the 'email' value is not present
9   * on the $admin array.
10  */
11 public function find_record_id($admin) {
12   if (!isset($admin['email'])) {
13     throw new Exception('Admin email was not provided: '
14       . print_r($admin, TRUE));
15   }
16
17   $result = $this->db
18       ->select('ea_users.id')
19       ->from('ea_users')
20       ->join('ea_roles', 'ea_roles.id = ea_users.id_roles', 'inner')
21       ->where('ea_users.email', $admin['email'])
22       ->where('ea_roles.slug', DB_SLUG_ADMIN)
23       ->get();
24
25   if ($result->num_rows() == 0) {
26     throw new Exception('Could not find admin record id.');
27   }
28
29   return intval($result->row()->id);
30 }
```

6.2 Αρχιτεκτονική κώδικα

Η εφαρμογή είναι γραμμένη χρησιμοποιώντας τις εξής τεχνολογίες: PHP, Javascript, HTML, CSS, MySQL. Εκτός αυτών έχουν χρησιμοποιηθεί και κάποια βοηθητικά εργαλεία τα οποία διευκολύνουν τον προγραμματιστή στο να πετύχει καλύτερο αποτέλεσμα σε μικρότερο χρόνο. Αυτά τα εργαλεία (frameworks) όπως έχουν αναφερθεί και σε προηγούμενο κεφάλαιο είναι τα CodeIgniter (PHP), jQuery (Javascript), Bootstrap (CSS + Javascript).

Όσον αφορά την αρχιτεκτονική του κώδικα έχει επιλεχθεί το μοντέλο MVC (Model - View - Controller) το οποίο υλοποιείται με άριστη απόδοση και οργάνωση χάρη στο framework CodeIgniter. Ο κώδικας PHP έχει χωριστεί σε τρία μέρη (models, views, controllers) και με αυτόν τον τρόπο παραμένει σε όλο τον κώδικα της εφαρμογής. Ο διαχωρισμός αυτός βελτιώνει τις συνθήκες συντήρησης γιατί είναι ξεκάθαρο σε ποιο από τα τρία ξεχωριστά σημεία ανήκει μια λειτουργία, όταν αυτή αναζητείται από τον προγραμματιστή. Έχουν συγγραφεί και δοκιμαστεί κλάσεις models για κάθε οντότητα οι οποίες αναλαμβάνουν την διαχείριση των δεδομένων με την βάση και παρέχουν μεθόδους που επαναχρησιμοποιούνται σε διάφορες περιπτώσεις. Επίσης έχουν δημιουργηθεί views για κάθε σελίδα που μπορεί να δει ο χρήστης τα οποία συνδέονται με ένα κομμάτι CSS κώδικα, υπεύθυνο για την μορφοποίησή τους. Τέλος τον συντονισμό αυτών των τμημάτων αναλαμβάνουν οι κλάσεις controllers οι οποίες είτε είναι υπεύθυνες για την σωστή φόρτωση μιας σελίδας της εφαρμογής, είτε απαντούν σε κλήσεις της JavaScript που γίνονται μέσω της τεχνολογίας AJAX.

Πολύ μεγάλο μέρος της εφαρμογής έχει γραφεί σε JavaScript για να μπορέσει το περιβάλλον εργασίας του χρήστη να γίνει αρκετά φιλικό και λειτουργικό. Ο JavaScript κώδικας χωρίζεται σε διάφορες κλάσεις και namespace τα οποία χρησιμοποιούνται από μια ή και παραπάνω σελίδες και στόχο έχουν να ”ζωντανέψουν” το περιεχόμενο προσθέτοντας διαδραστικότητα. Πολλές φορές είναι απαραίτητο να εκτελεστούν κλήσεις AJAX προς τον server για την λήψη πρόσθετων πληροφοριών, είτε για να αποσταλούν δεδομένα τα οποία σηματοδοτούν για παράδειγμα κάποια επεξεργασία ή και διαγραφή εγγραφής από την βάση δεδομένων. Η χρήση του AJAX κρίνεται σημαντική διότι με αυτήν αποφεύγονται οι συνεχείς επαναφορτώσεις των σελίδων, οι οποίες θα γινόταν για να μπορέσει ο client να επικοινωνήσει με τον server. Αυτό το κομμάτι αναλαμβάνεται εξ ολοκλήρου από την JavaScript και προσδίδει ευελιξία και ταχύτητα στην χρήση της εφαρμογής. Το framework jQuery αποτελεί σημαντικό εργαλείο για την διεκπεραίωση διαφόρων λειτουργιών μέσω της JavaScript διότι δίνει την δυνατότητα στον προγραμματιστή να γράψει κώδικα όμορφα δομημένο και πολύ πιο αποδοτικό από ότι θα ήταν χωρίς την χρήση του. Αυτή η ιδιότητα της βιβλιοθήκης συντελεί και στην δημοτικότητά της και την χρήση της από κολοσσούς ανάπτυξης λογισμικού.

Για την μορφοποίηση των σελίδων της εφαρμογής χρησιμοποιήθηκε το πιο διαδεδομένο CSS framework την συγκεκριμένη την περίοδο, το Bootstap. Χρησιμοποιώντας

αυτό το framework γράφτηκε νέο CSS το οποίο μορφοποιεί τις σελίδες έτσι ώστε να ανταποκρίνονται όπως πρέπει σε διάφορα μεγέθη οθονών, ενώ παράλληλα δεν χαλάει την συμβατότητα μεταξύ των διάφορων περιηγητών διαδικτύου. Το Bootstrap περιέχει και κάποια πρόσθετα JavaScript τα οποία βοηθούν σημαντικά στο οπτικό αποτέλεσμα της διεπαφής χρήστη.

Στο παρακάτω σχεδιάγραμμα γίνεται σαφής ο διαχωρισμός του κώδικα του συστήματος στα διάφορα τμήματα που το απαρτίζουν και η χρήση των εξωτερικών εργαλείων που συντέλεσαν στην ορθή ανάπτυξη της εφαρμογής.

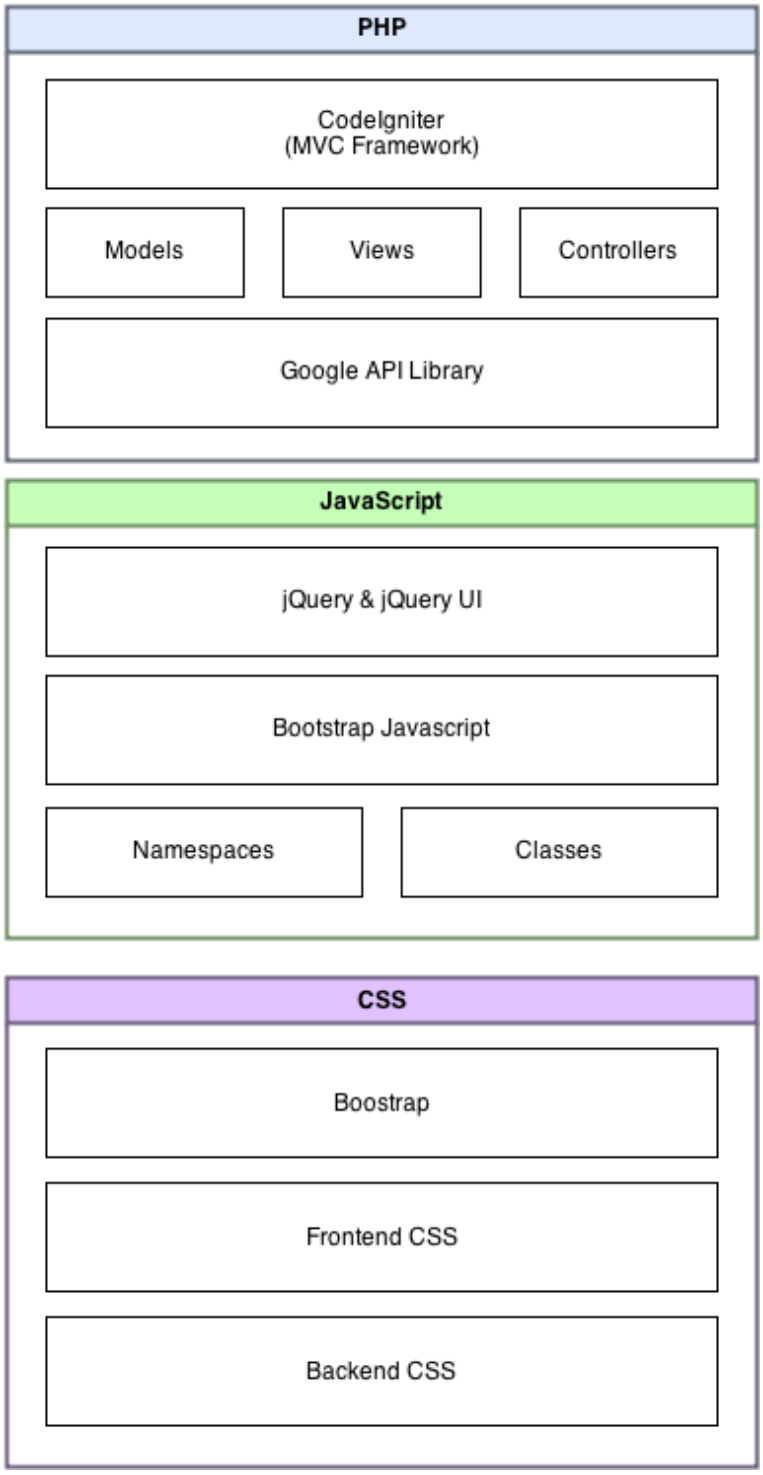
6.3 Υλοποίηση συστήματος

Εφόσον ο αρχικός σχεδιασμός είχε ολοκληρωθεί ξεκίνησε η υλοποίηση της εφαρμογής με πρώτη εργασία τον σχεδιασμό της βάσης δεδομένων. Έχοντας ήδη σχεδιασμένο το domain model η δημιουργία του σχήματος της βάσης έγινε γρήγορα και διατηρήθηκε ως την ολοκλήρωση του έργου με μικρές προσθήκες όπου ήταν απαραίτητο.

Στην συνέχεια, πριν γραφεί κώδικας θα έπρεπε να γίνει η επιλογή και το στήσιμο των εξωτερικών βιβλιοθηκών που θα κρίνονταν απαραίτητα για την λειτουργία του συστήματος. Σε αυτήν την φάση επιλέχθηκαν οι βασικές βιβλιοθήκες (CodeIgniter, Google API Library, jQuery, Bootstrap) καθώς και η σημαντικότερη περίπτωση χρήσης για να υλοποιηθεί πρώτη. Αυτή δεν ήταν άλλη από την κράτηση ενός ραντεβού από τον πελάτη. Αυτή η απόφαση πάρθηκε γιατί με αυτόν τον τρόπο θα καθορίζονταν εν μέρη και η αρχιτεκτονική του συστήματος καθώς αυτό θα εξελισσόταν σταδιακά με την ολοκλήρωση και των υπόλοιπων περιπτώσεων χρήσης.

Η κύρια ροή εργασιών ως προς την υλοποίηση μιας περίπτωσης χρήσης αποτελείται από τα παρακάτω βήματα:

1. Συγγραφή της κλάσης model για την συγκεκριμένη οντότητα. Μερικές φορές αυτή η διαδικασία μπορεί να συμπεριλάμβανε και την δημιουργία model και για άλλες οντότητες που εμπλέκονταν στην περίπτωση χρήσης, έτσι ώστε να μπορέσει να λειτουργήσει σωστά ο κώδικας συνολικά. Οι περισσότερες κλάσεις ακολουθούν το ίδιο πρότυπο σχεδίασης και μεθόδων με μικρές διαφοροποιήσεις ανάλογα με την οντότητα που διαχειρίζονται.



Διάγραμμα 6.3: Τα τμήματα που απαρτίζουν το Easy!Appointments.

2. Έλεγχος των model με δημιουργία unit tests. Μετά την ολοκλήρωση των model αυτά θα έπρεπε να δοκιμαστούν έτσι ώστε να διασφαλιστεί η σωστή λειτουργία τους. Εκτός αυτού όμως η συγγραφή unit test είναι και μια καλή ευκαιρία ως παράδειγμα της χρήσης των model από το υπόλοιπο σύστημα. Αν εντοπιζό-

ταν κάποιο πρόβλημα κατά την εκτέλεση των test αυτό διορθωνόταν και τα test εκτελόντουσαν πάλι έως ότου να ολοκληρωθούν όλα με επιτυχία.

3. Εφόσον τα model ήταν ολοκληρωμένα στην συνέχεια δημιουργήθηκαν οι controllers και οι αντίστοιχες συναρτήσεις που θα ήταν υπεύθυνες για την λειτουργία του view που αντιστοιχούσε στην εκάστοτε περίπτωση χρήσης. Έτσι εκτός από τις συναρτήσεις που αναλάμβαναν να φορτώσουν μια σελίδα της εφαρμογής συγκεντρώνοντας τα δεδομένα που ήταν απαραίτητα, υλοποιήθηκαν και οι κλήσεις AJAX που ήταν απαραίτητες από την JavaScript. Αυτές οι κλήσεις συνήθως αναλάμβαναν την διεκπεραίωση κάποιας ενέργειας προς την βάση δεδομένων και επέστρεφαν πάντα κάποιο αποτέλεσμα για να μπορέσει να συνεχίσει την λειτουργία της το τμήμα της JavaScript.

4. Στην συνέχεια υλοποιούνταν το αντίστοιχο view που θα έβλεπε ο χρήστης. Σε αυτό τοποθετούνταν ο κώδικας PHP, HTML και η μορφοποίηση της σελίδας (CSS) γραφόταν στο αντίστοιχο αρχείο css έτσι ώστε να παραχθεί ένα καλαίσθητο και φιλικό αποτέλεσμα.

5. Όταν το view ήταν έτοιμο θα έπρεπε να του προστεθεί και κάποια λειτουργικότητα έτσι ώστε να μπορεί να ανταποκριθεί στις ενέργειες του χρήστη. Για κάθε σελίδα χρησιμοποιούνται μια πληθώρα από βιβλιοθήκες, namespaces, κλάσεις και πρόσθετα JavaScript. Στα αντίστοιχα αρχεία τοποθετήθηκε ο κώδικας που θα ρύθμιζε την λειτουργία της σελίδας και τις ασύγχρονες κλήσεις προς τον server (AJAX).

6. Τέλος εφόσον όλα ήταν έτοιμα και η περίπτωση χρήσης είχε υλοποιηθεί χωρίς προβλήματα, όλος ο κώδικας που είχε γραφεί έπρεπε να εξεταστεί (review) για τυχόν προβλήματα λογικής και για την βελτίωση της απόδοσης του, μικραίνοντας όσο είναι δυνατόν την σύζευξη και αυξάνοντας την συνοχή.

Εδώ θα χρειαστεί να αναφερθεί ότι όλες οι κλήσεις AJAX που αφορούν το backend έχουν μεταφερθεί σε μια κλάση controller, ξεχωριστά από τον κύριο controller του backend για να είναι καλύτερα οργανωμένες. Αν μελλοντικά ο αριθμός τους και η πολυπλοκότητα τους αυξηθεί τότε θα χρειαστεί να διαιρεθούν ξανά για να μπορέσουν να συντηρούνται πιο εύκολα.

6.4 Περιγραφή βασικών αλγορίθμων

Σε αυτήν την ενότητα θα γίνει ανάλυση κάποιων βασικών αλγορίθμων που αποτελούν κρίσιμα τμήματα για την λειτουργία του συστήματος. Η περιγραφή θα γίνει σχολιάζοντας τα τμήματα κώδικα που απαρτίζουν αυτούς τους αλγορίθμους αναφέροντας και τις συγκεκριμένες γραμμές στον οποίο αναφέρεται. Στην επόμενη ενότητα παρέχονται κάποια σχεδιαγράμματα τα οποία μπορούν να βοηθήσουν στην κατανόηση αυτών των αλγορίθμων.

6.4.1 Πλήρης συγχρονισμός με το Google Calendar

Η διαδικασία του πλήρη συγχρονισμού των ραντεβού με το Google Calendar αποτελεί ένας από τους κυριότερους αλγορίθμους του Easy!Appointments. Η πολυπλοκότητα της διαδικασίας συγχρονισμού δεδομένων κατέστησαν την υλοποίηση αυτού του τμήματος κώδικα αρκετά ενδιαφέρον και το αποτέλεσμα κατάφερε να καλύψει τις αρχικές απαιτήσεις. Μπορεί μελλοντικά να υπάρξουν βελτιώσεις στον κώδικα, αλλά την συγκεκριμένη στιγμή ο αλγόριθμος λειτουργεί επιτυχώς και συγχρονίζει τα ραντεβού του συστήματος με τα συμβάντα που έχει περάσει ο χρήστης στο Google Calendar.

```
1  <?php
2  /**
3   * Complete synchronization of appointments between Google
4   * Calendar and Easy!Appointments.
5   *
6   * This method will completely sync the appointments of a provider
7   * with his Google Calendar account. The sync period needs to be
8   * relatively small, because a lot of API calls might be necessary
9   * and this will lead to consuming the Google limit for the Calendar
10  * API usage.
11  *
12  * @param numeric $provider_id Provider record to be synced.
13  *
14  */
15 public function sync($provider_id = NULL) {
16   try {
17     // The user must be logged in.
18     $this->load->library('session');
```

```
19      if ($this->session->userdata('user_id') == FALSE) return;
20
21      if ($provider_id === NULL) {
22        throw new Exception('Provider id not specified.');
23      }
24
25      $this->load->model('appointments_model');
26      $this->load->model('providers_model');
27      $this->load->model('services_model');
28      $this->load->model('customers_model');
29      $this->load->model('settings_model');
30
31      $provider = $this->providers_model->get_row($provider_id);
32
33      // Check whether the selected provider has google sync enabled.
34      $google_sync = $this->providers_model
35          ->get_setting('google_sync', $provider['id']);
36      if (!$google_sync) {
37        throw new Exception('The selected provider has not the  '
38            . 'Google synchronization setting enabled.');
39      }
40
41      $google_token = json_decode($this->providers_model
42          ->get_setting('google_token', $provider['id']));
43      $this->load->library('google_sync');
44      $this->google_sync->refresh_token($google_token->refresh_token);
45
46      // Fetch provider's appointments that belong to the sync
47      // time period.
48      $sync_past_days = $this->providers_model
49          ->get_setting('sync_past_days', $provider['id']);
50      $sync_future_days = $this->providers_model
51          ->get_setting('sync_future_days', $provider['id']);
52      $start = strtotime('-' . $sync_past_days
53          . ' days', strtotime(date('Y-m-d')));
54      $end = strtotime('+' . $sync_future_days
55          . ' days', strtotime(date('Y-m-d')));
56
```

```
57      $where_clause = array(
58        'start_datetime >=' => date('Y-m-d H:i:s', $start),
59        'end_datetime <=' => date('Y-m-d H:i:s', $end),
60        'id_users_provider' => $provider['id']
61      );
62
63      $appointments =
64          $this->appointments_model->get_batch($where_clause);
65
66      $company_settings = array(
67        'company_name' => $this->settings_model
68            ->get_setting('company_name'),
69        'company_link' => $this->settings_model
70            ->get_setting('company_link'),
71        'company_email' => $this->settings_model
72            ->get_setting('company_email')
73      );
74
75      // Sync each appointment with Google Calendar by following
76      // the project's sync protocol (see documentation).
77      foreach($appointments as $appointment) {
78        if ($appointment['is_unavailable'] == FALSE) {
79          $service = $this->services_model
80              ->get_row($appointment['id_services']);
81          $customer = $this->customers_model
82              ->get_row($appointment['id_users_customer']);
83        } else {
84          $service = NULL;
85          $customer = NULL;
86        }
87
88        // If current appointment not synced yet, add to gcal.
89        if ($appointment['id_google_calendar'] == NULL) {
90          $google_event = $this->google_sync
91              ->add_appointment($appointment, $provider,
92              $service, $customer, $company_settings);
93          $appointment['id_google_calendar'] = $google_event->id;
94          $this->appointments_model->add($appointment);// Save gcal id
```

```
95      } else {
96        // Appointment is synced with google calendar.
97        try {
98          $google_event = $this->google_sync
99              ->get_event($appointment['id_google_calendar']);
100
101         if ($google_event->status == 'cancelled') {
102           throw new Exception('Event is cancelled, remove the '
103               . 'record from Easy!Appointments.');
104         }
105
106         // If gcal event is different from e!a appointment
107         // then update e!a record.
108         $is_different = FALSE;
109         $appt_start = strtotime($appointment['start_datetime']);
110         $appt_end = strtotime($appointment['end_datetime']);
111         $event_start =
112           strtotime($google_event->getStart()->getDateTime());
113         $event_end =
114           strtotime($google_event->getEnd()->getDateTime());
115
116         if ($appt_start != $event_start
117             || $appt_end != $event_end) {
118           $is_different = TRUE;
119         }
120
121         if ($is_different) {
122           $appointment['start_datetime'] =
123               date('Y-m-d H:i:s', $event_start);
124           $appointment['end_datetime'] =
125               date('Y-m-d H:i:s', $event_end);
126           $this->appointments_model->add($appointment);
127         }
128
129       } catch(Exception $exc) {
130         // Appointment not found on gcal, delete from e!a.
131         $this->appointments_model->delete($appointment['id']);
132         $appointment['id_google_calendar'] = NULL;
```

```
133         }
134       }
135     }
136
137     // :: ADD GCAL EVENTS THAT ARE NOT PRESENT ON E!A
138     $events = $this->google_sync->get_sync_events($start, $end);
139
140     foreach($events->getItems() as $event) {
141       $results = $this->appointments_model->get_batch(
142           array('id_google_calendar' => $event->getId()));
143       if (count($results) == 0) {
144         // Record doesn't exist in E!A, so add the event now.
145         $appointment = array(
146           'start_datetime' => date('Y-m-d H:i:s',
147               strtotime($event->start->getDateTime())),
148           'end_datetime' => date('Y-m-d H:i:s',
149               strtotime($event->end->getDateTime())),
150           'is_unavailable' => TRUE,
151           'notes' => $event->getSummary()
152               . ' ' . $event->getDescription(),
153           'id_users_provider' => $provider_id,
154           'id_google_calendar' => $event->getId(),
155           'id_users_customer' => NULL,
156           'id_services' => NULL,
157         );
158         $this->appointments_model->add($appointment);
159       }
160     }
161
162     echo json_encode(AJAX_SUCCESS);
163
164   } catch(Exception $exc) {
165     echo json_encode(array(
166       'exceptions' => array($exc)
167     ));
168   }
169 }
```

Η μέθοδος αυτή καλείται κάθε φορά που πρέπει να τρέξει ο αλγόριθμος συγχρονισμού για έναν πάροχο υπηρεσιών. Στο πρώτο μέρος του κώδικα ελέγχεται αν ο χρήστης έχει τα δικαιώματα να τρέξει αυτήν την μέθοδο και αν έχει δοθεί το αναγνωριστικό της εγγραφής του πάροχου. Έπειτα φορτώνονται τα απαραίτητα models και γίνεται η λήψη των πληροφοριών του πάροχου από την βάση (γραμμές 18 - 31).

Για να συνεχιστεί η διαδικασία θα πρέπει να ελεγχθεί αν ο πάροχος έχει ενεργό τον συγχρονισμό με το Google Calendar. Αν η επιλογή αυτή είναι ενεργή τότε ο αλγόριθμος χρησιμοποιεί το token του πάροχου για να πιστοποιήσει την χρήση των δεδομένων του στο Google Calendar, διαφορετικά η διαδικασία τερματίζεται (γραμμές 34 - 44).

Για να γίνει εξοικονόμηση κλήσεων προς την υπηρεσία της Google αλλά και να μειωθεί ο χρόνος διεκπεραίωσης του αλγορίθμου συγχρονισμού, το χρονικό διάστημα μέσα στο οποίο θα συγχρονισθούν τα δεδομένα περιορίζεται στο εύρος ημερών που έχει τεθεί ως ρύθμιση για τον κάθε πάροχο (προεπιλεγμένη τιμή 5 ημέρες στο παρελθόν και 5 στο μέλλον). Αυτό είναι το χρονικό διάστημα στο οποίο θα ελεγχθούν όλα τα δεδομένα και από τα δύο συστήματα και θα συντονιστούν έτσι ώστε να είναι τα ίδια (γραμμές 48 - 55).

Το επόμενο κομμάτι κώδικα αφού πρώτα λάβει τα ραντεβού από την βάση δεδομένων του Easy!Appointments, εξετάζει τις εγγραφές μια προς μια για το αν έχουν συγχρονιστεί με το Google Calendar. Εδώ υπάρχουν οι εξής περιπτώσεις:

1. Το ραντεβού δεν έχει ακόμα συγχρονιστεί οπότε θα πρέπει να προστεθεί στο Google Calendar (γραμμές 89 - 94).

2. Το ραντεβού είναι συγχρονισμένο και πρέπει να ελεγχθεί αν υπάρχουν διαφορές με το συμβάν που είναι καταχωρημένο στο Google Calendar. Αν ναι τότε αυτό σημαίνει ότι ο χρήστης έχει αλλάξει τα στοιχεία του συμβάντος στο Google Calendar και η εγγραφή του ραντεβού στο Easy!Appointments θα πρέπει να ενημερωθεί (γραμμές 98 - 127).

3. Το ραντεβού είναι συγχρονισμένο αλλά δεν έχει βρεθεί στο Google Calendar. Εφόσον δεν έχει βρεθεί η εγγραφή σημαίνει ότι ο χρήστης την έχει διαγράψει από το Google Calendar και έτσι θα πρέπει να διαγραφεί και από το Easy!Appointments (γραμμές 131 - 132).

Με το πέρας αυτού του τμήματος κώδικα όλα τα ραντεβού του Easy!Appointments θα πρέπει να έχουν συγχρονιστεί με το Google Calendar.

Τα μη διαθέσιμα διαστήματα χρησιμοποιούνται ως ραντεβού στον συγκεκριμένο αλγόριθμο με την διαφορά ότι δεν υπάρχουν σε αυτά πληροφίες για κάποιο πελάτη ή υπηρεσία (γραμμές 84 - 85).

Υπάρχουν όμως συμβάντα στην υπηρεσία της Google τα οποία μπορεί να έχουν προστεθεί απευθείας στο Google Calendar και να μην υπάρχουν στο Easy!Appointments. Σε αυτήν την περίπτωση θα πρέπει να ανιχνευθούν και να εξεταστούν όλα τα συμβάντα που αντιστοιχούν στην χρονική περίοδο συγχρονισμού (5 ημέρες πριν και 5 ημέρες μετά την τρέχουσα ημερομηνία) και να ελεγχθεί αν υπάρχει κάποιο συμβάν που δεν είναι συγχρονισμένο.

Αυτήν την εργασία αναλαμβάνει το επόμενο κομμάτι κώδικα το οποίο χρησιμοποιώντας την βιβλιοθήκη Google API μπορεί να διαβάσει τα συμβάντα τα οποία βρίσκονται στο Google Calendar. Η διαδικασία ξεκινάει με την λήψη αυτών των συμβάντων τα οποία στην συνέχεια εξετάζονται ένα προς ένα για το αν υπάρχουν στο Easy!Appointments. Αν όχι τότε προστίθενται και συγχρονίζονται και στα δύο συστήματα και έτσι διασφαλίζεται η ακεραιότητα των δεδομένων και στα δύο συστήματα (γραμμές 138 - 160).

Τέλος η συνάρτηση επιστρέφει την σταθερά AJAX_SUCCESS την οποία θα διαβάσει η JavaScript και έτσι να γνωρίζει ότι η διαδικασία έχει ολοκληρωθεί με επιτυχία. Διαφορετικά αν προκύψουν σφάλματα αυτά επιστρέφονται σε JSON μορφή και εμφανίζονται με ένα φιλικό μήνυμα προς τον χρήστη.

6.4.2 Υπολογισμός διαθέσιμων ωρών πάροχου

Ένα κομβικό σημείο στον κώδικα της εφαρμογής είναι ο υπολογισμός των διαθέσιμων ωρών ενός πάροχου, στις οποίες μπορεί ένας πελάτης να κλείσει ένα ραντεβού για μια υπηρεσία, χωρίς να υπάρχει σύγκρουση με άλλα συμβάντα. Για να επιτευχθεί ο υπολογισμός αυτός χρειάζεται να γίνουν αρκετοί έλεγχοι έτσι ώστε τα αποτελέσματα να είναι σωστά και να μην δημιουργούνται προβλήματα με τα πλάνα των πάροχων υπηρεσιών. Η διαδικασία χωρίζεται σε δύο μεθόδους με την πρώτη να υπολογίζει τα ελεύθερα χρονικά διαστήματα του πάροχου και την δεύτερη να υπολογίζει τις ακριβείς ώρες στις οποίες θα μπορεί ο πελάτης να κλείσει ραντεβού.

```php
<?php
/**
 * Get an array containing the free time periods (start - end) of a
 * selected date.
 *
 * This method is very important because there are many cases where
 * the system needs to know when a provider is avaible for an
 * appointment. This method will return an array that belongs to the
 * selected date and contains values that have the start and the end
 * time of an available time period.
 *
 * @param numeric $provider_id The provider's record id.
 * @param string $selected_date The date to be checked (MySQL
 * formatted string).
 * @param array $exclude_appointments This array contains the ids of
 * the appointments that will not be taken into consideration when the
 * available time periods are calculated.
 * @return array Returns an array with the available time periods of
 * the provider.
 */
private function get_provider_available_time_periods($provider_id,
    $selected_date, $exclude_appointments = array()) {
  $this->load->model('appointments_model');
  $this->load->model('providers_model');

  // Get the provider's working plan and reserved appointments.
  $working_plan = json_decode($this->providers_model
      ->get_setting('working_plan', $provider_id), true);

  $where_clause = array(
    'DATE(start_datetime)'=>date('Y-m-d', strtotime($selected_date)),
    'id_users_provider'=>$provider_id
  );

  $reserved_appointments = $this->appointments_model
      ->get_batch($where_clause);

  // Sometimes it might be necessary to not take into account some
```

```
39  // appointment records in order to display what the providers'
40  // available time periods would be without them.
41  foreach ($exclude_appointments as $excluded_id) {
42    foreach ($reserved_appointments as $index => $reserved) {
43      if ($reserved['id'] == $excluded_id) {
44        unset($reserved_appointments[$index]);
45      }
46    }
47  }
48
49  // Find the empty spaces on the plan. The first split between the
50  // plan is due to a break (if exist). After that every reserved
51  // appointment is considered to be a taken space in the plan.
52  $selected_date_working_plan =
53      $working_plan[strtolower(date('l', strtotime($selected_date)))];
54  $available_periods_with_breaks = array();
55
56  if (isset($selected_date_working_plan['breaks'])) {
57    foreach($selected_date_working_plan['breaks'] as $index=>$break){
58      // Split the working plan to available time periods that do not
59      // contain the breaks in them.
60      $last_break_index = $index - 1;
61
62      if (count($available_periods_with_breaks) === 0) {
63        $start_hour = $selected_date_working_plan['start'];
64        $end_hour = $break['start'];
65      } else {
66        $start_hour = $selected_date_working_plan['breaks']
67            [$last_break_index]['end'];
68        $end_hour = $break['start'];
69      }
70
71      $available_periods_with_breaks[] = array(
72        'start' => $start_hour,
73        'end' => $end_hour
74      );
75    }
76    // Add the period from the last break to the end of the day.
```

```
77      $available_periods_with_breaks[] = array(
78        'start'=>$selected_date_working_plan['breaks'][$index]['end'],
79        'end'=>$selected_date_working_plan['end']
80      );
81    }
82
83    // Break the empty periods with the reserved appointments.
84    $available_periods_with_appointments =
85            $available_periods_with_breaks;
86
87    foreach($reserved_appointments as $appointment) {
88      foreach($available_periods_with_appointments as $index=>&$period){
89        $a_start =
90          date('H:i',strtotime($appointment['start_datetime']));
91        $a_end =
92          date('H:i', strtotime($appointment['end_datetime']));
93        $p_start =
94          date('H:i', strtotime($period['start']));
95        $p_end =
96          date('H:i', strtotime($period['end']));
97
98        if ($a_start <= $p_start && $a_end <= $p_end
99            && $a_end <= $p_start) {
100         // The appointment does not belong in this time period, so
101         // we will not change anything.
102       } else if ($a_start <= $p_start && $a_end <= $p_end
103           && $a_end >= $p_start) {
104         // The appointment starts before the period and finishes
105         // somewhere inside.We will need to break this period and
106         // leave the available part.
107         $period['start'] = $a_end;
108       } else if ($a_start >= $p_start && $a_end <= $p_start) {
109         // The appointment is inside the time period, so we will
110         // split the period into two new others.
111         unset($available_periods_with_appointments[$index]);
112         $available_periods_with_appointments[] = array(
113           'start' => $p_start,
114           'end' => $a_start
```

```
115         );
116         $available_periods_with_appointments[] = array(
117           'start' => $a_end,
118           'end' => $p_end
119         );
120       } else if ($a_start >= $p_start && $a_end >= $p_start
121           && $a_start <= $p_end) {
122         // The appointment starts in the period and finishes out
123         // of it. We will need to remove the time that is taken
124         // from the appointment.
125         $period['end'] = $a_start;
126       } else if ($a_start >= $p_start && $a_end >= $p_end
127           && $a_start >= $p_end) {
128         // The appointment does not belong in the period so do not
129         // change anything.
130       } else if ($a_start <= $p_start && $a_end >= $p_end
131           && $a_start <= $p_end) {
132         // The appointment is bigger than the period, so this period
133         // needs to be removed.
134         unset($available_periods_with_appointments[$index]);
135       }
136     }
137   }
138   return array_values($available_periods_with_appointments);
139 }
```

Το πρώτο πράγμα που πρέπει να γίνει είναι η λήψη του πλάνου εργασίας του πάροχου καθώς και των ήδη καταχωρημένων ραντεβού για την επιλεγμένη ημερομηνία. Επίσης υπάρχει και η περίπτωση να πρέπει να αποκλειστούν κάποια ραντεβού κατά τον υπολογισμό των διαθέσιμων ωρών οπότε αν έχουν οριστεί τέτοιες εγγραφές δεν λαμβάνονται υπόψιν στον υπολογισμό. Αυτή η επιλογή είναι απαραίτητη όταν χρειάζεται ο πελάτης να επεξεργαστεί ένα ήδη καταχωρημένο ραντεβού το οποίο δεν θα πρέπει να εμφανίζει ως δεσμευμένη την ώρα που αντιστοιχεί στο ίδιο στο ημερολόγιο του πάροχου. Τα στοιχεία αυτά θα χρησιμοποιηθούν έτσι ώστε τα διαθέσιμα διαστήματα που θα υπολογιστούν να αντιπροσωπεύουν τον χρόνο στον οποίο ο πάροχος θα είναι διαθέσιμος (γραμμές 23 - 47).

Έπειτα θα διαχωριστούν τα ελεύθερα χρονικά διαστήματα του πάροχου από τα διαλείμματα και τα ήδη καταχωρημένα ραντεβού. Αρχικά για την επιλεγμένη ημέρα του ραντεβού ελέγχονται αν υπάρχουν καθόλου διαλείμματα. Αν ναι, τότε τα διαθέσιμα διαστήματα χωρίζονται μεταξύ των διαλειμμάτων του πάροχου (γραμμές 54 - 81) και παράγονται νέα χρονικά διαστήματα. Στην συνέχεια λαμβάνονται υπόψιν τα ραντεβού που έχουν ήδη κρατηθεί. Τα διαθέσιμα χρονικά διαστήματα του πάροχου θα διασπαστούν ξανά μεταξύ των ραντεβού αυτών και έτσι θα ολοκληρωθεί η διαδικασία του υπολογισμού (γραμμές 84 - 137). Αν την επιλεγμένη ημερομηνία ο πάροχος δεν έχει κανένα ραντεβού τότε δεν πραγματοποιείται καμία επιπλέον διάσπαση και το αποτέλεσμα επιστρέφεται όπως είναι. Στην επόμενη μέθοδο ο πίνακας που περιέχει τα διαστήματα θα χρησιμοποιηθεί για να υπολογιστούν οι ακριβείς διαθέσιμες ώρες στις οποίες θα μπορεί ο πελάτης να κλείσει κάποιο ραντεβού για την επιλεγμένη υπηρεσία.

```
1  <?php
2  /**
3   * [AJAX] Get the available appointment hours for the given date.
4   *
5   * This method answers to an AJAX request. It calculates the
6   * available hours for thegiven service, provider and date.
7   *
8   * @param numeric $_POST['provider_id'] The selected provider's
9   * record id.
10  * @param string $_POST['selected_date'] The selected date of
11  * which the available hours we want to see.
12  * @param numeric $_POST['service_duration'] The selected service
13  * duration in minutes.
14  * @param string $_POST['manage_mode'] Contains either 'true' or
15  * 'false' and determines the if current user is managing an already
16  * booked appointment or not.
17  * @return Returns a json object with the available hours.
18  */
19 public function ajax_get_available_hours() {
20    $this->load->model('providers_model');
21    $this->load->model('appointments_model');
22    $this->load->model('settings_model');
23
24    try {
```

```
25      // If manage mode is TRUE then the following we should not
26      // consider the selected appointment when calculating the
27      // available time periods of the provider.
28      $exclude_appointments = ($_POST['manage_mode'] === 'true')
29          ? array($_POST['appointment_id'])
30          : array();
31
32      $empty_periods = $this->get_provider_available_time_periods(
33          $_POST['provider_id'], $_POST['selected_date'],
34          $exclude_appointments);
35
36      // Calculate the available appointment hours for the given
37      // date. The empty spaces are broken down to 15 min and if
38      // the service fit in each quarter then a new available hour
39      // is added to the "$available_hours" array.
40
41      $available_hours = array();
42
43      foreach ($empty_periods as $period) {
44        $start_hour = new DateTime($_POST['selected_date']
45            . ' ' . $period['start']);
46        $end_hour = new DateTime($_POST['selected_date']
47            . ' ' . $period['end']);
48
49        $minutes = $start_hour->format('i');
50
51        if ($minutes % 15 != 0) {
52          // Change the start hour of the current space in
53          // order to be on of the following: 00, 15, 30, 45.
54          if ($minutes < 15) {
55            $start_hour->setTime($start_hour->format('H'), 15);
56          } else if ($minutes < 30) {
57            $start_hour->setTime($start_hour->format('H'), 30);
58          } else if ($minutes < 45) {
59            $start_hour->setTime($start_hour->format('H'), 45);
60          } else {
61            $start_hour->setTime($start_hour->format('H') + 1, 00);
62          }
```

```
63        }
64
65        $current_hour = $start_hour;
66        $diff = $current_hour->diff($end_hour);
67
68        while (($diff->h * 60 + $diff->i)
69            > intval($_POST['service_duration'])) {
70          $available_hours[] = $current_hour->format('H:i');
71          $current_hour->add(new DateInterval("PT15M"));
72          $diff = $current_hour->diff($end_hour);
73        }
74      }
75
76      // If the selected date is today, remove past hours. It is
77      // important include the timeout before booking that is set
78      // in the backoffice the system. Normally we might want the
79      // customer to book an appointment that is at least half or
80      // one hour from now. The setting is stored in minutes.
81      if (date('m/d/Y', strtotime($_POST['selected_date']))
82          == date('m/d/Y')) {
83        if ($_POST['manage_mode'] === 'true') {
84          $book_advance_timeout = 0;
85        } else {
86          $book_advance_timeout =
87            $this->settings_model->get_setting('book_advance_timeout');
88        }
89
90        foreach($available_hours as $index => $value) {
91          $available_hour = strtotime($value);
92          $current_hour = strtotime('+' . $book_advance_timeout
93              . ' minutes', strtotime('now'));
94          if ($available_hour <= $current_hour) {
95            unset($available_hours[$index]);
96          }
97        }
98      }
99
100     $available_hours = array_values($available_hours);
```

```
101     echo json_encode($available_hours);
102
103   } catch(Exception $exc) {
104     echo json_encode(array(
105       'exceptions' => array(exceptionToJavaScript($exc))
106     ));
107   }
108 }
```

Η δεύτερη μέθοδος αποτελεί απάντηση σε κλήση της JavaScript με χρήση της τεχνικής AJAX. Όταν ο client καλεί αυτήν την μέθοδο, παρέχει τα στοιχεία του πάροχου, την διάρκεια της επιλεγμένης υπηρεσίας (σε λεπτά) και το αν ο χρήστης επεξεργάζεται το συγκεκριμένο ραντεβού ή όχι (παράμετρος manage_mode). Αυτό που εκτελείται αρχικά είναι η λήψη των ελεύθερων χρονικών διαστημάτων του πάροχου χρησιμοποιώντας την προαναφερθέντα μέθοδο get_provider_available_time_periods (γραμμές 30 - 36).

Έπειτα θα υπολογιστούν οι διαθέσιμες ώρες στις οποίες θα μπορέσει ο χρήστης να κλείσει ραντεβού. Αυθαίρετα και για λόγους ευχρηστίας έχει τεθεί το χρονικό διάστημα μεταξύ των ελεύθερων ωρών να είναι τα 15 λεπτά. Αυτό που κάνει το συγκεκριμένο κομμάτι κώδικα είναι ουσιαστικά ο διαχωρισμός των ελεύθερων χρονικών διαστημάτων του πάροχου σε ώρες τις οποίες χωρίζουν 15 λεπτά τουλάχιστον και οι οποίες μπορούν να χωρέσουν την διάρκεια της υπηρεσίας για την οποία ενδιαφέρεται ο πελάτης, πριν την λήξη του διαθέσιμου χρονικού διαστήματος (γραμμές 43 - 76).

Στο τελευταίο μέρος αυτής της μεθόδου ελέγχεται αν η επιλεγμένη ημερομηνία αντιστοιχεί στην σημερινή και αν αυτό ισχύει αφαιρούνται οι παρελθοντικές διαθέσιμες ώρες έτσι ώστε να μην μπορεί ο πελάτης να κλείσει ραντεβού σε μια παρελθοντική χρονική στιγμή. Το σύστημα στο frontend δεν επιτρέπει ούτως ή άλλος την επιλογή παρελθοντικής ημερομηνίας, αλλά απαιτείται στην συγκεκριμένη περίπτωση να ελεγχθούν οι παρελθοντικές ώρες του ραντεβού. Επίσης είναι σημαντικό να αναφερθεί ότι η εφαρμογή παρέχει μια παράμετρο η οποία ορίζει το χρονικό διάστημα που θα πρέπει να χωρίζει ένα ραντεβού από την ώρα που αυτό γίνεται κράτηση ή επεξεργάζεται. Ο λόγος γίνεται για την ρύθμιση του συστήματος με το όνομα ”book_advance_timeout” η οποία μετράται σε λεπτά και λαμβάνεται υπόψιν στον υπολογισμό των διαθέσιμων ωρών.

6.5 Διαγράμματα Κώδικα

Σε αυτήν την ενότητα θα παρατεθούν κάποια διαγράμματα κώδικα τα οποία θα βοηθήσουν τον αναγνώστη στην κατανόηση της λειτουργίας του συστήματος και στον τρόπο με τον οποίο διεκπεραιώνονται οι εργασίες που απαιτούνται στην εκάστοτε περίπτωση χρήσης. Τα διαγράμματα αυτά ακολουθούν το σχεδιαστικό πρότυπο UML το οποίο αποτελεί την πιο δημοφιλής γλώσσα μοντελοποίησης εδώ και αρκετά χρόνια.

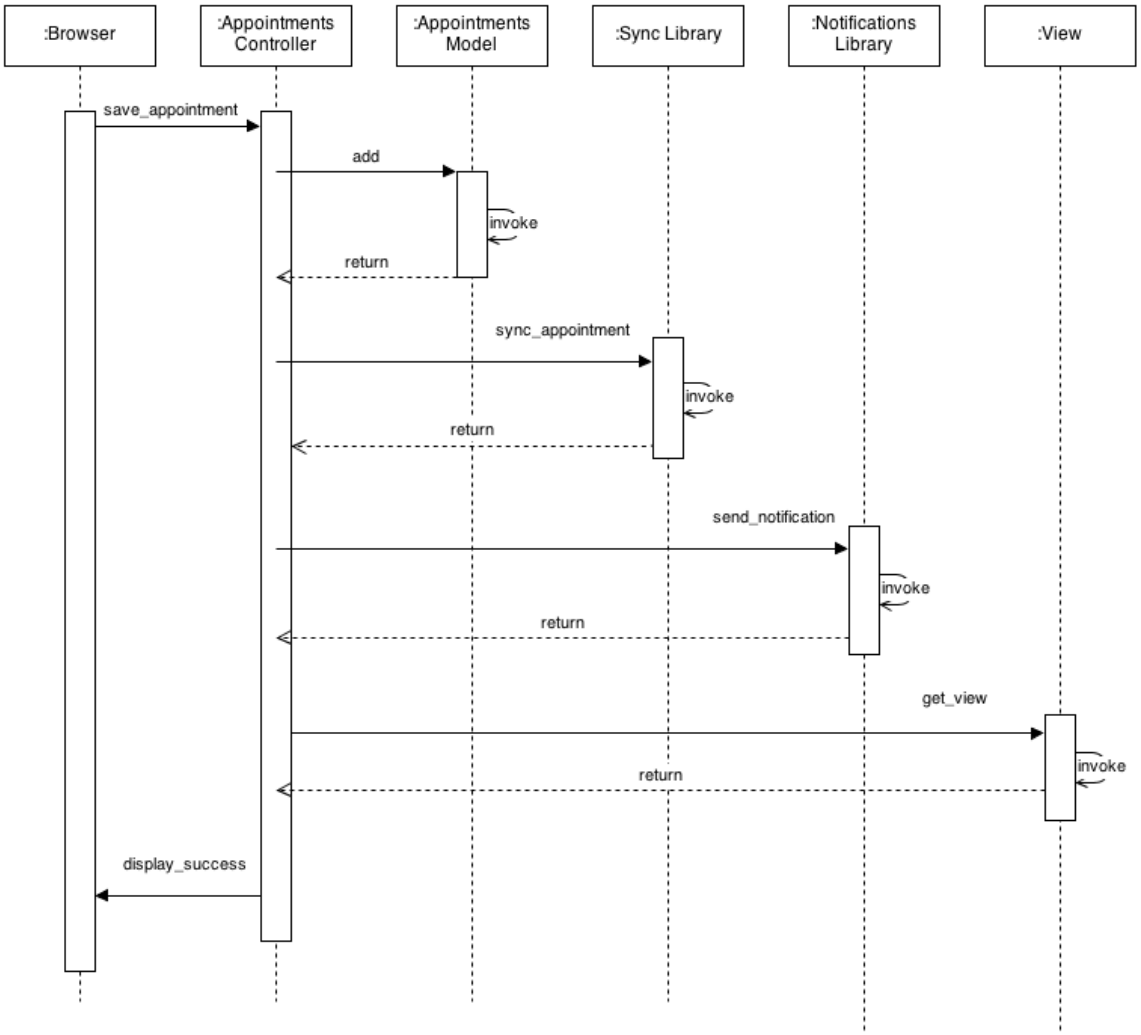
Η εφαρμογή που χρησιμοποιήθηκε για τον σχεδιασμό των διαγραμμάτων αυτών είναι η draw.io και πρόκειται για μια διαδικτυακή πλατφόρμα με την οποία μπορούν να γίνουν σχεδιαγράμματα πολλών διαφορετικών τύπων. Το draw.io είναι δωρεάν προς χρήση και μπορεί να βρεθεί στην διεύθυνση http://www.draw.io.

6.5.1 Διαγράμματα ροής

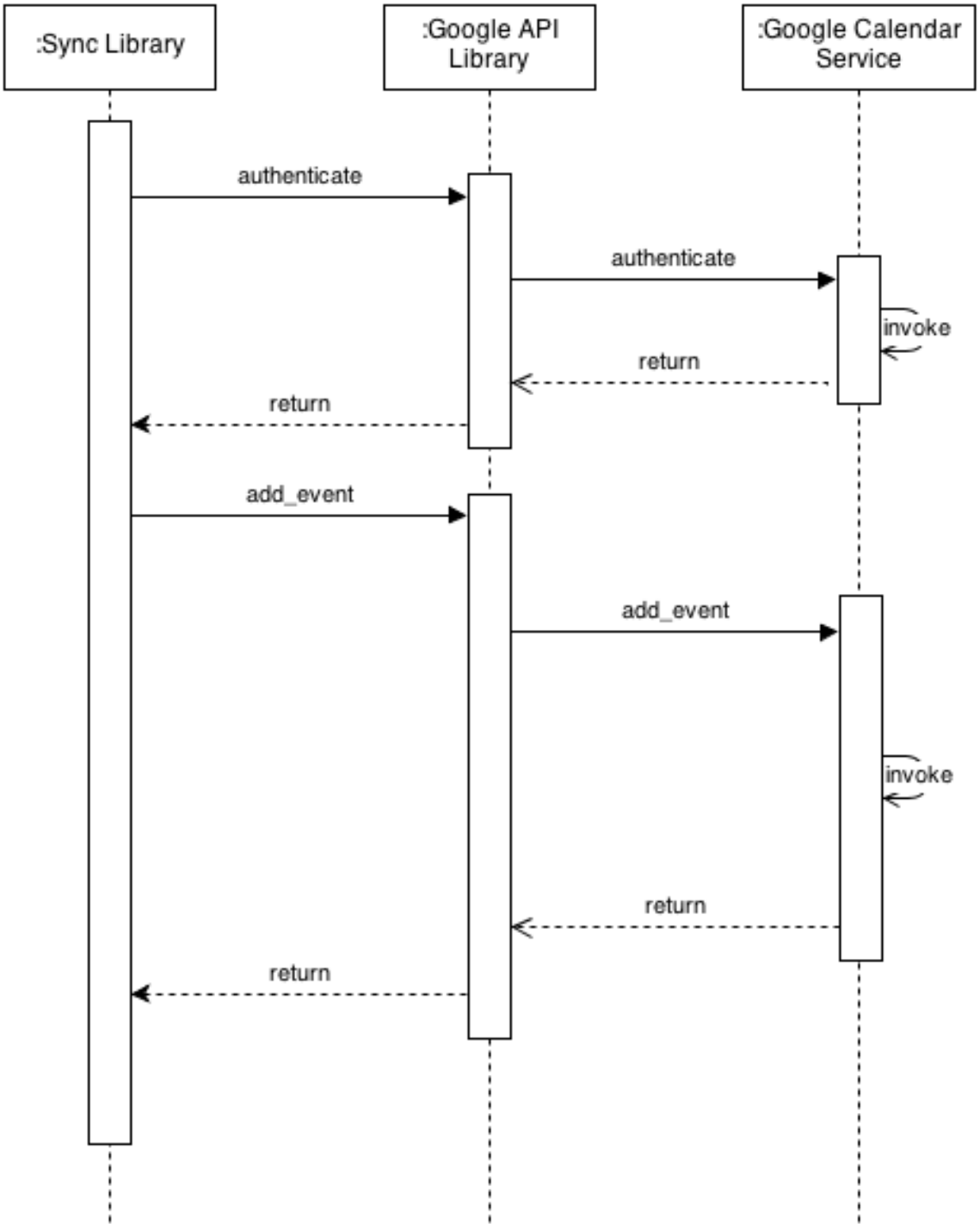
Τα διαγράμματα ροής δείχνουν τον τρόπο και την σειρά με την οποία λειτουργούν οι διεργασίες και τα αντικείμενα μεταξύ τους για την διεκπεραίωση ενός σκοπού. Σε αυτά είναι εύκολο να διακριθούν ποιοι ηθοποιοί, αντικείμενα, διεπαφές και μέθοδοι αλληλεπιδρούν έτσι ώστε τα δεδομένα που χρειάζονται για την εργασία να παραχθούν επιτυχώς και να φτάσουν ακέραια στον προορισμό τους. Σε αυτά τα διαγράμματα ο χρονικός προσδιορισμός της κάθε αλληλεπίδρασης είναι εμφανής και πολύ σημαντικός για να μπορέσει ο προγραμματιστής να καταλάβει με ποια σειρά θα πρέπει να πορευτεί η εκτέλεση έτσι ώστε αυτός να καταλήξει σε αναμενόμενο αποτέλεσμα. Συνήθως τα διαγράμματα ροής συγχέονται με το σενάριο κάποιας περίπτωσης χρήσης αλλά μπορούν να διασπαστούν και σε μικρότερα τμήματα τα οποία να επικεντρώνουν στα σημεία που είναι πιο σημαντικά.

6.5.2 Διαγράμματα δραστηριότητας

Τα διαγράμματα δραστηριότητας αποτελούν γραφικές παρουσιάσεις της δραστηριότητας του που ακολουθεί το σύστημα ανάλογα με τις αποφάσεις που λαμβάνονται μέσα από τον κώδικα. Σε αυτά τα διαγράμματα μπορούν να φανούν τα σημεία στα οποία υπάρχουν βρόγχοι επανάληψης, τα σημεία όπου μπορούν να συμβούν λογικά σφάλματα (οι απαιτήσεις για συνέχιση της εκτέλεσης δεν πληρούνται) όπως και επίσης τις διαδικασίες που τρέχουν ταυτόχρονα και σε ποιο σημείο γίνεται αυτό. Κατά



Διάγραμμα 6.4: Στο διάγραμμα ροής εμφανίζεται η διαδικασία που εκτελείται για την αποθήκευση ενός ραντεβού μετά την επιτυχή καταχώρηση του από τον πελάτη.

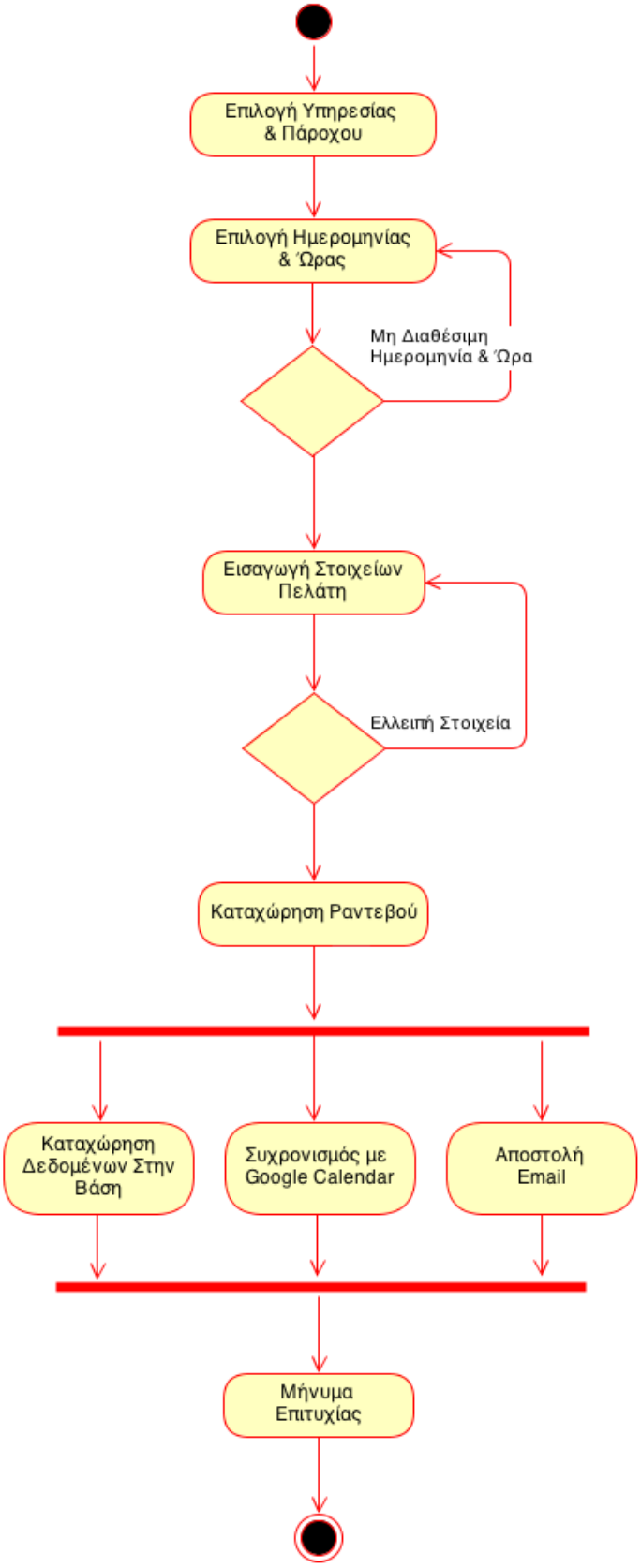


Διάγραμμα 6.5: Στο διάγραμμα ροής εμφανίζεται η διαδικασία με την οποία πραγματοποιείται η προσθήκη ενός ραντεβού στο Google Calendar.

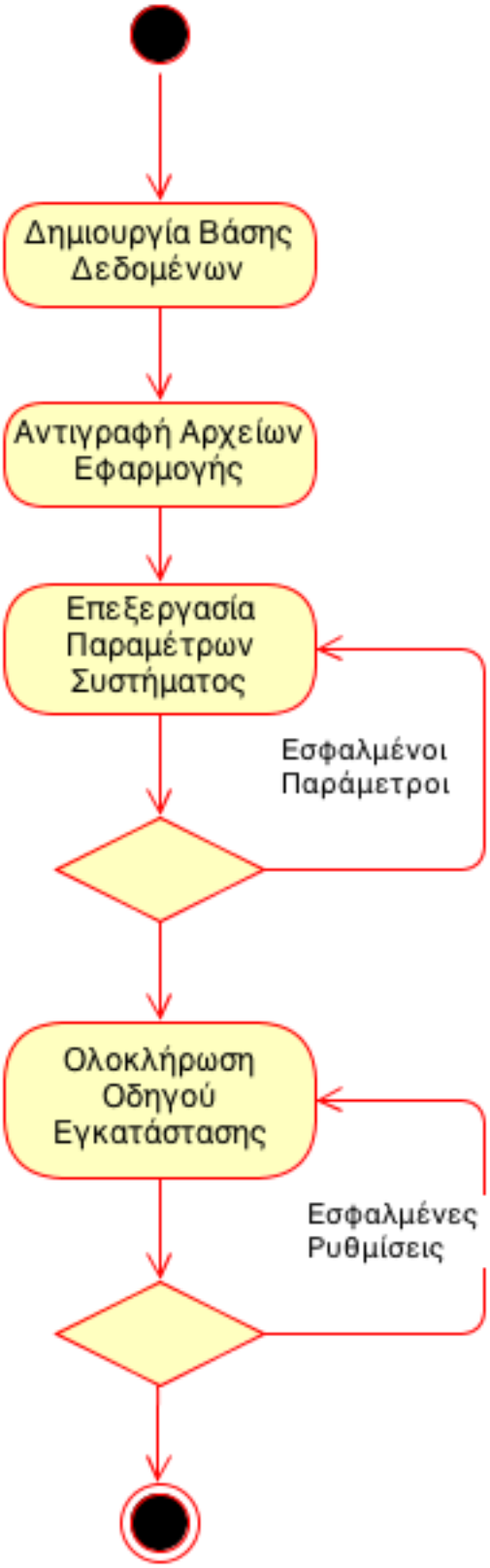
κύριο λόγο τα διαγράμματα δραστηριότητας δείχνει την συνολική ροή του ελέγχου μέσα από την εκτέλεση μιας συγκεκριμένης διαδικασίας.

6.5.3 Διαγράμματα κλάσεων

Στην τεχνολογία λογισμικού, τα διαγράμματα κλάσεων περιγράφουν την στατική δομή ενός συστήματος δείχνοντας τις κλάσεις, τις ιδιότητες, τις λειτουργίες και τις σχέσεις μεταξύ των αντικειμένων. Τα σχεδιαγράμματα αυτά είναι τα βασικότερα για έναν προγραμματιστή διότι μπορεί άμεσα να πληροφορηθεί σχετικά με την δομή του κώδικα και με ποιόν τρόπο θα πρέπει να συνεχιστεί η διαδικασία της υλοποίησης. Επίσης είναι εμφανές και οι αρχιτεκτονικές επιλογές που έχουν γίνει καθώς κάθε σχεδιαστικό πρότυπο που πρέπει να έχει ο κώδικας μπορεί να διακριθεί και να περιγραφή σε αυτά τα διαγράμματα. Καλή πρακτική είναι πάντα ένα διάγραμμα να περιέχει μόνο την ουσία οπότε στο υποκεφάλαιο αυτό εμφανίζονται διαγράμματα κλάσης τα οποία δείχνουν κάποιες σχεδιαστικές επιλογές που έχουν γίνει στην εφαρμογή.

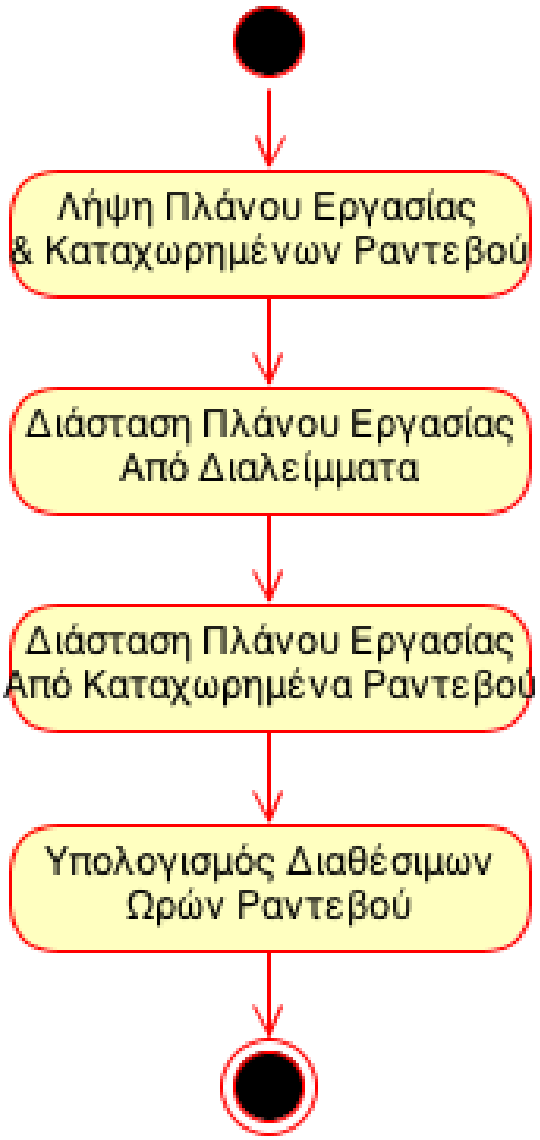


Διάγραμμα 6.6: Αναπαράσταση της διαδικασίας κράτησης ραντεβού.

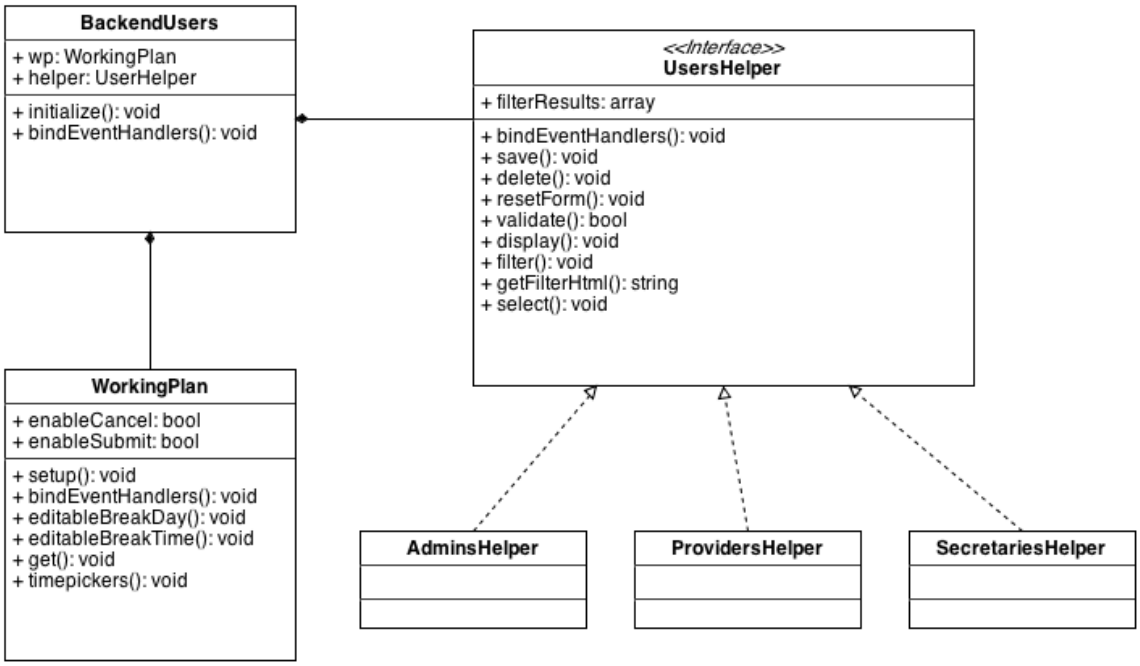


Διάγραμμα 6.7: Αναπαράσταση της διαδικασίας εγκατάστασης της εφαρμογής.

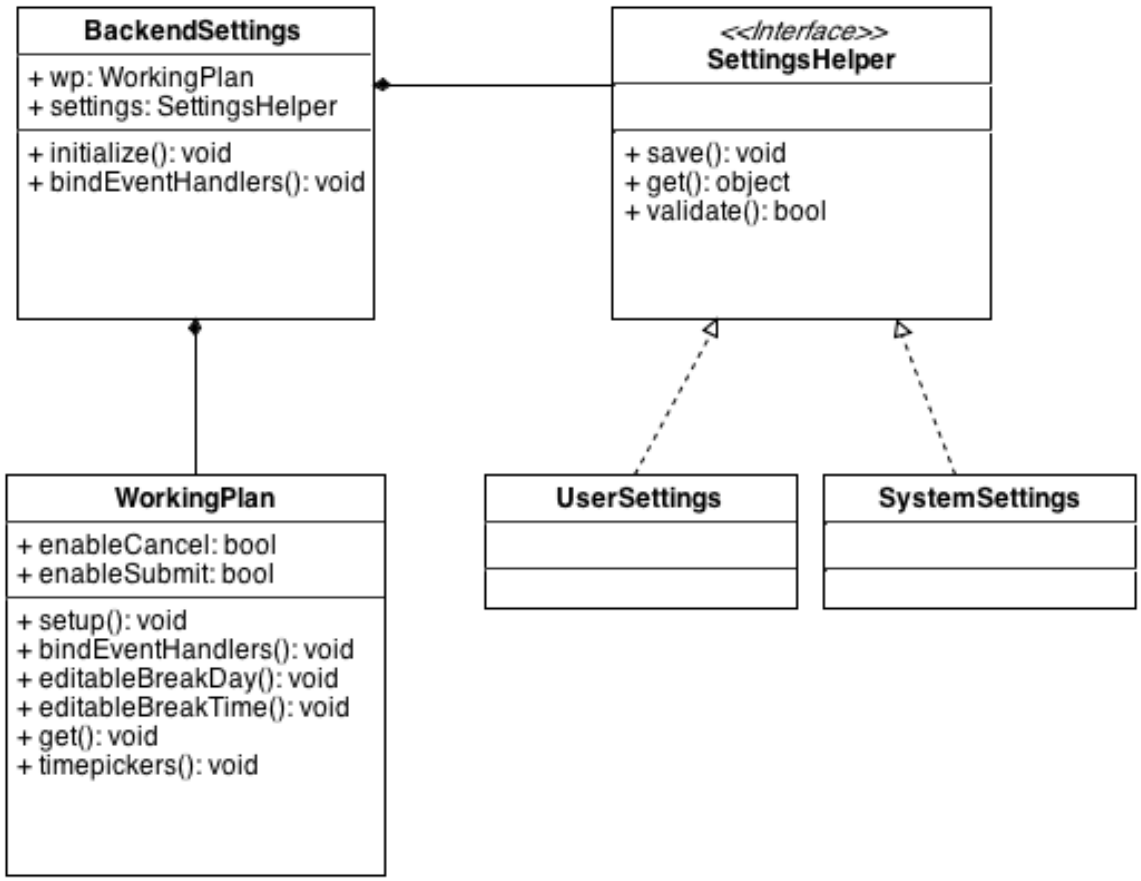
Διάγραμμα 6.8: Αναπαράσταση της διαδικασίας συγχρονισμού ραντεβού μεταξύ του Easy!Appointments και του Google Calendar. Ο αλγόριθμος συγχρονισμού είναι αμφίδρομος.



Διάγραμμα 6.9: Αναπαράσταση της διαδικασίας υπολογισμού των διαθέσιμων ωρών για κράτηση ραντεβού από τον πελάτη.



Διάγραμμα 6.10: Στο διάγραμμα αυτό φαίνεται ο τρόπος με τον οποίον υλοποιείται η λειτουργικότητα της σελίδας διαχείρισης των χρηστών του συστήματος μέσω της JavaScript. Η κάθε ενότητα διαχείρισης (διαχειριστές, πάροχοι και γραμματείς) έχουν την δικιά τους κλάση, η οποία καθορίζει με ποιόν τρόπο θα χρειαστεί να ανταποκριθεί η εφαρμογή.



Διάγραμμα 6.11: Σε αυτό το διάγραμμα κλάσεων φαίνονται οι κλάσεις που συμμετέχουν στον JavaScript κώδικα των κεντρικών ρυθμίσεων της εφαρμογής. Οι κάθε μια κλάση αντιπροσωπεύει την λειτουργία μιας κατηγορίας ρυθμίσεων και μαζί χρησιμοποιούνται για την σωστή λειτουργία της σελίδας.

Κεφάλαιο 7

Έλεγχος Συστήματος

Σε κάθε τομέα παραγωγής προϊόντων είναι πολύ σημαντικό να παράγονται προϊόντα τα οποία να τηρούν πάντα τις προδιαγραφές τους και να μπορούν να αντεπεξέλθουν στις απαιτήσεις του καταναλωτικού κοινού. Η φήμη και η εμπιστοσύνη που προσδίδει μια εταιρεία είναι κομβικά χαρακτηριστικά για την βιωσιμότητας της. Κάθε επαγγελματίας είναι απαραίτητο να είναι σε θέση να εγγυηθεί για την ποιότητα του προϊόντος ή της υπηρεσίας που παρέχει ως αντάλλαγμα της αμοιβής του.

Ο τρόπος διασφάλισης της ποιότητας διαφέρει ανάλογα με την φύση του προϊόντος ή της υπηρεσίας και μπορεί να εκτελεστεί με διάφορους μεθόδους. Για παράδειγμα αν το προϊόν ήταν κάποιο τρόφιμο, η εταιρία θα έπρεπε να είναι σίγουρη ότι είναι σε άριστη κατάσταση πριν φτάσει στο τραπέζι του καταναλωτή, διότι αν δεν το έκανε αυτό θα υπήρχαν επιπλοκές στην υγεία των καταναλωτών. Αντίστοιχα μια εταιρία που παρέχει μια υπηρεσία πρέπει να διασφαλίσει και να ελέγξει την ποιότητα παροχής της υπηρεσίας με διάφορους τρόπους. Ένας από αυτούς θα ήταν να λαμβάνει τις παρατηρήσεις των καταναλωτών αφότου λάβουν την υπηρεσία ή να περνάει από εσωτερικές εξετάσεις και εκπαίδευση τους υπαλλήλους της έτσι ώστε να είναι σίγουρη ότι αυτοί θα μπορούν να παρέχουν σωστά και αξιόπιστα την υπηρεσία στους πελάτες.

Στην διαδικασία ανάπτυξης λογισμικού υπάρχουν αντίστοιχα διάφοροι τρόποι ελέγχου ότι το λογισμικό που αναπτύσσεται τηρεί τις προδιαγραφές του. Κάποιοι από αυτούς τους τρόπους είναι τα unit testing, fuzz testing, δημοσίευση δοκιμαστική έκδοσης (beta version) κ.α. Το πιο κοντινό εργαλείο ελέγχου στον προγραμματιστή είναι η διαδικασία unit testing, η οποία εφαρμόζεται αποκλειστικά σε αντικειμενοστραφή κώδικα. Παρακάτω θα γίνει μια ανάλυση αυτής της τεχνικής ελέγχου και θα αναφερθούν οι

μέθοδοι και η διαδικασία ελέγχου πάνω στο Easy!Appointments.

7.1 Unit testing

Για την υλοποίηση unit tests πάνω στον κώδικα είναι απαραίτητο να τηρούνται δύο πράγματα: (1) η αντικειμενοστραφείς δομή και (2) η χρήση κάποιας βιβλιοθήκης ή εργαλείου το οποίο μπορεί να βοηθήσει στην οργάνωση και καλύτερη υλοποίηση των tests.

Με τον όρο unit testing εννοείται η δοκιμή μίας "λειτουργικής μονάδας" του λογισμικού που αναπτύσσεται. Η κάθε λειτουργική μονάδα απομονώνεται από τις υπόλοιπες και δοκιμάζεται ξεχωριστά σε διάφορες κατάστασης. Για αυτόν τον λόγο είναι απαραίτητο ο κώδικας να έχει αντικειμενοστραφής δομή. Η διαδικασία χωρίζεται στην συγγραφή πολλαπλών unit test, συναρτήσεων δηλαδή που δοκιμάζουν μια διαδικασία για συγκεκριμένες τιμές εισόδου. Σε κάθε περίπτωση στόχος είναι να υπάρχει ελεγχόμενη έξοδος έτσι ώστε να μπορέσει ο προγραμματιστής να είναι σίγουρος ότι το σύστημα θα λειτουργήσει σωστά σε οποιαδήποτε κατάσταση και αν βρίσκεται. Κατά την διαδικασία αυτήν μπορούν να βρεθούν πολύ εύκολα πολλά προβλήματα και ασυνέπειες στον κώδικα ενός συστήματος, τα οποία χρειάζεται να αντιμετωπιστούν.

Για να μπορέσουν να υλοποιηθούν αυτά τα test είναι απαραίτητο να χρησιμοποιηθεί κάποια βιβλιοθήκη ή εργαλείο, το οποίο θα κατέχει τις βασικές συναρτήσεις ελέγχου αποτελεσμάτων και επιπρόσθετα λειτουργίες για την παραγωγή αναφορών, οι οποίες περιέχουν τα αποτελέσματα των δοκιμών. Υπάρχουν πάρα πολλά εργαλεία που κάνουν αυτήν την δουλειά, το καθένα για μια συγκεκριμένη γλώσσα προγραμματισμού. Τα πιο διαδεδομένα εργαλεία είναι αυτά που ανήκουν στην οικογένεια xUnit (Junit, CppUnit, NUnit κ.α).

Τα εργαλεία αυτά μπορούν συνήθως κάλλιστα να συνεργαστούν μαζί με άλλα εργαλεία ανάπτυξης έτσι ώστε να είναι πολύ εύκολο για τον προγραμματιστή να συμπεριλάβει την διαδικασία unit testing στην υλοποίηση του κάθε συστήματος.

7.2 Easy!Appointments testing

Η συγγραφή των unit tests για το Easy!Appointments έγινε με την χρήση της ενσωματωμένης βιβλιοθήκης που παρέχει το CodeIginter. Η βιβλιοθήκη παρέχει τις βασικές λειτουργίες ελέγχου και παραγωγής αναφορών για τα tests του κώδικα. Προτιμήθηκε έναντι του phpunit λόγω της καλύτερης απόδοσης σε σχέση με το CodeIgniter Framework.

Η διαδικασία της δοκιμής του συστήματος ξεκίνησε από τα models, τις λειτουργικές μονάδες που διαχειρίζονται την κίνηση προς και από την βάση δεδομένων. Είναι απαραίτητο για το σύστημα να κατέχει ακέραια δεδομένα μιας και όλη η εφαρμογή βασίζεται σε αυτά. Η κάθε μέθοδος του κάθε model δοκιμάστηκε ξεχωριστά από τις υπόλοιπες για 3-5 διαφορετικές περιπτώσεις. Όσο αναπτύσσεται το σύστημα τόσο αυξάνονται και unit tests.

Για να γίνει αυτόματη εκτέλεση όλων των unit test που αντιστοιχούν σε ένα συγκεκριμένο model γράφτηκε η παρακάτω μέθοδος η οποία αφού ελέγξει τα ονόματα των test μεθόδων, εκτελεί μόνο εκεί τα οποία ξεκινούν από την λέξη ”test”. Έτσι αν κάποια μέθοδος δεν είναι έτοιμη ή δεν πρέπει να συμπεριληφθεί στην εκτέλεση των unit test αρκεί να αλλάξει την αρχή του ονοματός της και η μέθοδος δεν θα την λάβει υπόψιν.

```
1  <?php
2  /**
3   * Run all the available tests
4   */
5  public function run_all() {
6    // All the methods whose names start with ”test” are going to be
7    // executed. If you want a method to not be executed remove the
8    // ”test” keyword from the beginning.
9    $class_methods = get_class_methods('Unit_tests_admins_model');
10   foreach ($class_methods as $method_name) {
11     if (substr($method_name, 0, 5) === 'test_') {
12       call_user_func(array($this, $method_name));
13     }
14   }
15 }
```

7.3 Παραδείγματα

Στον παρακάτω κώδικα δοκιμάζεται η βασική ροή της περίπτωσης χρήσης "προσθήκη ραντεβού". Σε αυτό το test case η είσοδος της μεθόδου add() είναι σωστή και έτσι περιμένουμε ότι και το αποτέλεσμα της διαδικασίας θα είναι επιτυχία. Στο τέλος αφαιρούμε την εγγραφή που προστέθηκε για να μην μείνουν κατάλοιπα στην βάση. Σε κάθε unit test χρησιμοποιείται μόνο μια μέθοδος του model. Έτσι το κάθε test δεν επηρεάζεται από τυχόν προβλήματα σε άλλες μεθόδους του model.

```
1  <?php
2  /**
3    * Test the appointment add method - insert new record.
4    */
5  private function test_add_appointment_insert() {
6      // Add - insert new appointment record to the database.
7      $appointment_data = array(
8       'start_datetime' => '2013-05-01 12:30:00',
9       'end_datetime' => '2013-05-01 13:00:00',
10      'notes' => 'Some notes right here...',
11      'id_users_provider' => $this->provider_id,
12      'id_users_customer' => $this->customer_id,
13      'id_services' => $this->service_id
14    );
15     $appointment_data['id'] = $this->CI->Appointments_Model
16             ->add($appointment_data);
17     $this->CI->unit->run($appointment_data['id'], 'is_int',
18              'Test if add() appointment (insert operation) '
19        . 'returned the db row id.');
20
21     // Check if the record is the one that was inserted.
22     $db_data = $this->CI->db->get_where('ea_appointments',
23             array('id' => $appointment_data['id']))->row_array();
24     $this->CI->unit->run($appointment_data, $db_data, 'Test if add() '
25       . 'appointment (insert operation) has successfully '
26       . 'inserted a record.');
27
28     // Delete inserted record.
29     $this->CI->db->delete('ea_appointments',
```

```
30              array('id' => $appointment_data['id']));
31  }
```

Στο παρακάτω unit test δοκιμάζεται η μέθοδος get_value() η οποία επιστρέφει την τιμή ενός πεδίου από την βάση. Στο συγκεκριμένο test case δίνεται ως παράμετρος ένα id εγγραφής, το οποίο δεν υπάρχει στην βάση. Η αναμενόμενη συμπεριφορά από το model είναι να εμφανιστεί ένα exception το οποίο να ειδοποιεί ότι η εγγραφή με το συγκεκριμένο id δεν βρέθηκε στην βάση.

```
1   <?php
2   /**
3    * Test the get field value method with a record id that
4    * doesn't exist in the db.
5    *
6    * A database exception is expected.
7    */
8   private function test_get_value_record_does_not_exist() {
9       $random_record_id = 843521368768;
10
11      $has_thrown_exception = FALSE;
12
13      try {
14          $this->CI->Appointments_Model
15          ->get_value('start_datetime', $random_record_id);
16      } catch (InvalidArgumentException $db_exc) {
17          $has_thrown_exception = TRUE;
18      }
19
20      $this->CI->unit->run($has_thrown_exception, TRUE,
21        'Test get_value() with record id that does not exist.');
22  }
```

Κάποια unit test δοκιμάζουν τις μεθόδους για σωστές τιμές και αναμένουν την επιτυχή ολοκλήρωση των διαδικασιών τους. Τα περισσότερα όμως tests σκοπό έχουν να δουν την συμπεριφορά του συστήματος για τιμές οι οποίες δεν είναι φυσιολογικές. Με αυτόν τον τρόπο μπορούν να προβλεφθούν πολλά bug και άλλα προβλήματα στον κώδικα και η εφαρμογή να είναι περισσότερο αξιόπιστη και δυνατή απέναντι σε σφάλματα.

Κεφάλαιο 8

Συμπεράσματα

Το αποτέλεσμα της εκπόνησης της εργασίας αυτής είναι ένα πλήρης σύστημα διαχείρισης ραντεβού, το οποίο μπορεί να παραμετροποιηθεί επαρκώς έτσι ώστε να καλύψει τις ανάγκες οποιασδήποτε επιχείρισης ανεξαρτήτου ειδικότητας και μεγέθους. Ο αρχικός σχεδιασμός αποδείχθηκε σωστός και έτσι το τελικό προϊόν πληροί τις απαιτήσεις για τις οποίες αναπτύχθηκε.

8.1 Προβλήματα

8.1.1 Διαχείριση χρόνου

Σημαντικότερο πρόβλημα σχετικά με την υλοποίηση της εφαρμογής ήταν η χρονική καθυστέρηση μιας και ανάμεσα στην ανάληψη της εργασίας και την περαίωση της, πραγματοποιήθηκε η πρακτική άσκηση σε εταιρεία πληροφορικής, καθώς και εργασία εκτός σχολής με άλλες εταιρείες πληροφορικής. Οι εξωτερικές υποχρεώσεις αυτές αποσπούσαν την συνεχή και ομαλή ανάπτυξη, κάτι που συνεχώς διασπούσε τον ειρμό και τον δημιουργικό οίστρο. Συμπέρασμα αυτού του σημαντικού προβλήματος είναι ότι θα πρέπει να γίνεται σαφής και ορθός προγραμματισμός του χρόνου υλοποίησης ενός έργου γιατί διαφορετικά οι πιθανότητες για χαμηλότερη ποιότητα υπηρεσίας ή ακόμα και αποτυχίας του έργου αυξάνονται εκθετικά.

8.1.2 Συγχρονισμός δεδομένων με το Google Calendar

Όσον αφορά την συνεργασία του συστήματος με την υπηρεσία Google Calendar, αλλά και γενικότερα με άλλες πιθανές υπηρεσίες το ζήτημα παραμένει στο πως θα παραμείνουν τα δεδομένα ακέραια και ενημερωμένα και στα δύο συστήματα, όταν δεν υπάρχει ένα κοινό μέσο αποθήκευσης. Το θέμα γιγαντώνεται μάλιστα όταν δεν υπάρχει πρόσβαση στον κώδικα του ενός από τα δύο συστήματα έτσι ώστε να δημιουργηθεί μια ”γέφυρα δεδομένων”. Για την επίλυση αυτού του θέματος ήταν αναγκαίο να δημιουργηθεί ένας αλγόριθμος συγχρονισμού ο οποίος θα ενεργοποιούνταν από την πλευρά του Easy!Appointments και θα αναλάμβανε την ενημέρωση και τον δύο συστημάτων με τα τελευταία δεδομένα. Για αυτόν τον σκοπό θα έπρεπε να καταγραφούν και να υλοποιηθούν κάποιοι κανόνες συγχρονισμού οι οποίοι θα μετέφεραν επιτυχώς τα ραντεβού αμφίδρομα και στα δύο συστήματα. Στις περιπτώσεις όπου η μεταφορά αυτή θα ήταν αδύνατη (σύγκρουση δεδομένων) ο χρήστης θα έπρεπε να αποφασίσει ποια εκδοχή των δεδομένων θα υπερισχύσει στο τέλος.

8.1.3 Διαχωρισμός δικαιωμάτων χρηστών

Ένα ακόμα πρόβλημα που αντιμετωπίστηκε κατά την διάρκεια την ανάπτυξης του έργου ήταν ο διαχωρισμός των δικαιωμάτων των χρηστών μέσα στο σύστημα. Ο κάθε χρήστης αναλόγως το είδος του (διαχειριστής, πάροχως, γραμματέας) έχει διαφορετικές δυνατότητες και δικαιώματα στα δεδομένα που αποθηκεύονται από το σύστημα. Αυτό συμβαίνει γιατί στις περισσότερες περιπτώσεις θα πρέπει να τηρηθεί η ιεραρχία της επιχείρησης, αλλά και επίσης γιατί θα πρέπει να διασφαλιστεί η ακεραιότητα των δεδομένων από τυχόν εσφαλμένες ενέργειες χρηστών σε βασικές ρυθμίσεις του συστήματος. Για τις κυριότερες ρυθμίσεις απαιτούνται τα δικαιώματα διαχειριστή και έτσι το σύστημα χρειάζεται απαραιτήτως πάντα έναν χρήστη διαχειριστή (ο χρήστης που δημιουργείται κατά την εγκατάσταση είναι ουσιαστικά ο πρώτος διαχειριστής της εφαρμογής). Για να λυθεί αυτό το πρόβλημα η εγγραφή του κάθε χρήστης στην βάση δεδομένων συνδέεται με έναν ρόλο, ο οποίος περιέχει τα δικαιώματα που του αντιστοιχούν. Έτσι για παράδειγμα ένας χρήστης που προορίζεται για πάροχος υπηρεσίας, θα έχει τα δικαιώματα που αντιστοιχούν στον ρόλο ”Πάροχος Υπηρεσίας”, όπως αυτά είναι αποθηκευμένα στην βάση δεδομένων. Έτσι κάθε φορά που συνδέεται ένας χρή-

στης στο διαχειριστικό κομμάτι της εφαρμογής τα δεδομένα σχετικά με τα δικαιώματα του και τον ρόλο του διαβάζονται από σελίδα σε σελίδα και η εφαρμογή μπορεί και γνωρίζει με ποιόν τρόπο θα πρέπει να εμφανιστούν τα δεδομένα και ποιες ενέργειες είναι διαθέσιμες στην κάθε περίπτωση.

8.2 Εξέλιξη της εφαρμογής

Όπως και σε κάθε έργο λογισμικού, υπάρχουν πολλά πράγματα τα οποία μπορούν να εξελιχθούν και να βελτιωθούν, καθώς και δυνατότητες οι οποίες μπορούν να προστεθούν για να κάνουν την εφαρμογή πιο εύχρηστη και αποδοτικότερη. Οι βελτιώσεις αυτές γίνονται στην φάση της συντήρησης και επέκτασης, σταδιακά, με σκοπό την ύπαρξη ενός ενημερωμένου προϊόντος στην αγορά. Με αυτόν τον τρόπο οι εταιρείες θα μπορούν να εμπιστεύονται την εν λόγω εφαρμογή και να την χρησιμοποιούν ως το δικό τους εργαλείο διαχείρισης των ραντεβού. Παρακάτω περιγράφονται κάποια σημεία στα οποία θα μπορούσε να εξελιχθεί μελλοντικά το σύστημα που παράχθηκε.

8.2.1 Mobile design

Με την πάροδο του χρόνου όλο και περισσότερες ”έξυπνες” συσκευές βρίσκονται στα χέρια των καταναλωτών και έτσι δημιουργείται η ανάγκη για χρήση των διαδικτυακών εφαρμογών από οθόνες που έχουν διαφορετικά μεγέθη οθονών. Εφόσον οι διαστάσεις για τις οθόνες του υπολογιστή έχουν καλυφθεί, το επόμενο βήμα είναι να σχεδιαστεί όλο το σύστημα για κινητές συσκευές και tablet. Με αυτόν τον τρόπο θα μπορούν οι χρήστες του Easy!Appointments να χρησιμοποιούν το σύστημα από το κινητό του πολύ πιο άνετα και έτσι να είναι πάντα ενημερωμένοι σχετικά με τα ραντεβού τους όπου και αν βρίσκονται. Προϋπόθεση για αυτό πάντα είναι μια ενεργή σύνδεση με το διαδίκτυο. Για να υλοποιηθεί αυτή η δυνατότητα θα χρειαστεί να γραφεί CSS κώδικας ο οποίος να εμφανίζει την εφαρμογή διαφορετικά σε κινητές συσκευές.

8.2.2 Μετάφραση της διεπαφής χρήστη

Η πρώτη υλοποίηση του συστήματος έχει γίνει εξολοκλήρου στην αγγλική γλώσσα, όπως και με τα περισσότερα συστήματα που απευθύνονται σε ένα ευρύ καταναλωτικό

κοινό. Το Easy!Appointments δέχεται κείμενο και σε άλλες γλώσσες (χρησιμοποιείται το encoding UTF-8) αλλά η διεπαφή, τα μηνύματα και τα αντικείμενα ελέγχου είναι όλα στα Αγγλικά. Για να γίνει πιο εύκολη η χρήση της εφαρμογής και από ανθρώπους οι οποίοι δεν είναι τόσο εξοικειωμένοι με αυτήν την γλώσσα θα πρέπει να μεταφραστεί όλο το σύστημα και σε άλλες κοινές γλώσσες. Για να επιτευχθεί αυτό στην συγκεκριμένη περίπτωση θα πρέπει να χρησιμοποιηθεί η ενσωματωμένη τεχνική του CodeIgniter.

8.2.3 Αναφορές δεδομένων

Το σύστημα μέσω της λειτουργίας του κρατάει διάφορα δεδομένα (ραντεβού, πελάτες, πάροχοι κτλ). Καλό θα ήταν αυτά τα δεδομένα να μπορούν να εξαχθούν με κάποιον τρόπο έτσι ώστε να μπορέσουν να χρησιμοποιηθούν και με άλλους τρόπους. Μια περίπτωση χρήσης θα ήταν η εξαγωγή των σημερινών ραντεβού ενός πάροχου σε μια εκτυπώσιμη αναφορά, έτσι ώστε να μπορεί ο χρήστης να την τυπώσει και να την έχει ως λίστα στο γραφείο. Μια άλλη περίπτωση χρήσης θα ήταν να εκτυπωθούν τα στοιχεία ενός πελάτη, καθώς και το ιστορικό των ραντεβού του. Οι αναφορές αυτές είναι πολύ χρήσιμες γιατί μπορούν να αναδείξουν γρήγορα δεδομένα και μάλιστα σε εκτυπώσιμη μορφή, κάτι που ακόμα χρειάζονται πολλές εταιρείες.

8.2.4 Στατιστικές πληροφορίες

Μια χρήσιμη μελλοντική δυνατότητα θα ήταν η συλλογή στατιστικών πληροφοριών σχετικά με τα ραντεβού που κλίνονται στο σύστημα. Από αυτήν την διαδικασία θα μπορούσαν να βγουν σημαντικές πληροφορίες όπως το ποια υπηρεσία ή πάροχος προτιμάται πιο συχνά, ποιες μέρες έχουν τα περισσότερα ραντεβού, πόσο συχνά ακυρώνονται τα ραντεβού και σε πόσο χρονικό διάστημα πριν. Αυτές οι πληροφορίες είναι πολύ σημαντικές για μια εταιρεία η οποία λειτουργεί με κρατήσεις ραντεβού γιατί έτσι μπορεί να γνωρίζει με ποιόν τρόπο λειτουργεί το πελατειακό κοινό της και έτσι να λειτουργεί αναλόγως για να παρέχει καλύτερη εξυπηρέτηση. Ένα παράδειγμα θα ήταν η περίπτωση ενός μεγάλου κομμωτηρίου στην οποία οι πελάτες θα κρατούσαν πάρα πολλά ραντεβού το Σάββατο κάθε εβδομάδας, γιατί πιθανόν έχουν περισσότερο χρόνο. Τα στατιστικά (εκτός της πείρας) θα έδειχναν την αυξημένη κίνηση του Σαββάτου και

έτσι ο διαχειριστής θα είχε διαθέσιμους όλους του πάροχους προς ραντεβού, για να μπορέσει να καλυφθεί το κοινό όσο καλύτερα γίνεται.

8.2.5 Δημιουργία RESTful υπηρεσίας

Κάθε μεγάλο διαδικτυακό σύστημα παρέχει και μια RESTful υπηρεσία η οποία μπορεί να απαντάει με δεδομένα σε διάφορες κλήσεις που της γίνονται, εφόσον βέβαια έχει πιστοποιηθεί ο client που επικοινωνεί μαζί τους. Με αυτόν τον τρόπο θα μπορούν άλλοι προγραμματιστές να φτιάχνουν εφαρμογές οι οποίες θα επικοινωνούν με το Easy!Appointments και θα διαχειρίζονται τα δεδομένα του συστήματος. Η υλοποίηση αυτής της δυνατότητας θα βοηθούσε πολύ την εξέλιξη και την χρήση του Easy!Appointments γιατί από εδώ και πέρα διάφορες εφαρμογές θα υλοποιούνταν κάνοντας δυνατή την χρήση των δεδομένων σε πολλές διαφορετικές περιστάσεις. Κάτι που είναι πάρα πολύ χρήσιμο για κάθε επαγγελματία (παραδείγματος χάρη η χρήση πελατών από CRM εφαρμογή).

8.2.6 Βελτίωση κώδικα

Τελευταίο αλλά και όχι λιγότερο σημαντικό είναι η συνεχής βελτίωση και ενημέρωση του κώδικα έτσι ώστε να είναι πάντα στην καλύτερη δυνατή κατάσταση. Καθώς εξελίσσεται ένα σύστημα λογισμικού είναι απαραίτητο να βελτιώνεται ο κώδικας και η δομή του. Επίσης είναι απαραίτητο να ενημερώνονται και τα εξωτερικά εργαλεία τα οποία χρησιμοποιούνται έτσι ώστε να διασφαλίζεται η ασφάλεια και η ποιότητα του συστήματος. Κατά καιρούς εμφανίζονται διάφορες ενημερώσεις ασφαλείας αλλά και διορθώσεων σφαλμάτων σε αυτά τα framework (CodeIgniter, jQuery κτλ) τα οποία θα χρειαστεί να συμπεριληφθούν και στο Easy!Appointments. Κάθε φορά που ο χρήστης λαμβάνει μια νέα έκδοση της εφαρμογής θα πρέπει ο κώδικας που την απαρτίζει να βρίσκεται σε πολύ καλή κατάσταση, να έχει ελεγχθεί και να λειτουργεί σωστά έτσι ώστε να εμπνέει εμπιστοσύνη προς τους χρήστες.

Βιβλιογραφία

[1] John W. Dower *Readings compiled for History 21.479.* 1991.

[2] The Japan Reader *Imperial Japan 1800-1945* 1973: Random House, N.Y.

[3] E. H. Norman *Japan's emergence as a modern state* 1940: International Secretariat, Institute of Pacific Relations.

[4] Bob Tadashi Wakabayashi *Anti-Foreignism and Western Learning in Early-Modern Japan* 1986: Harvard University Press.