

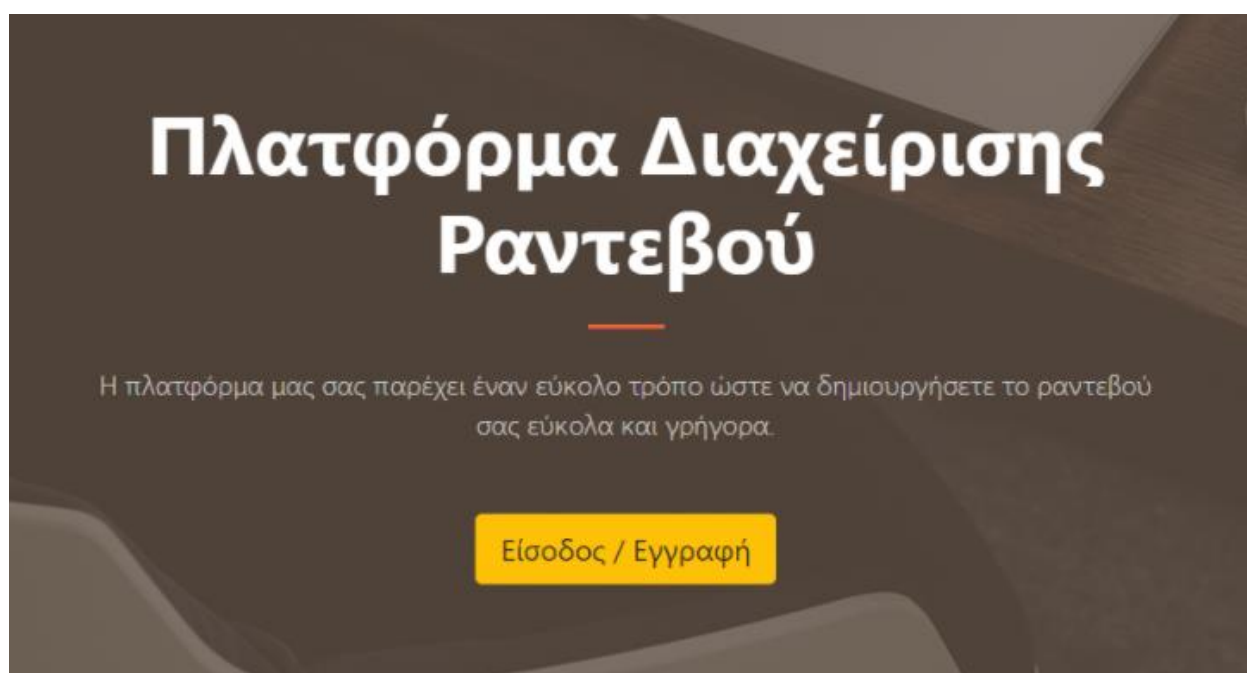


ΧΑΡΟΚΟΠΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ

Χαροκόπειο Πανεπιστήμιο
Πληροφορικής και Τηλεματικής
Πληροφορική και Τηλεματική
Τεχνολογίες και Εφαρμογές Ιστού

Πλατφόρμα διαχείρισης ραντεβού για
φορείς του δημοσίου
Διπλωματική Εργασία

Άγγελος Λίχας



Αθήνα, 2021



ΧΑΡΟΚΟΠΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ

Χαροκόπειο Πανεπιστήμιο
Πληροφορικής και Τηλεματικής
Πληροφορική και Τηλεματική
Τεχνολογίες και Εφαρμογές Ιστού

Τριμελής Εξεταστική Επιτροπή

ΤΣΑΔΗΜΑΣ ΑΝΑΡΓΥΡΟΣ

Ε.ΔΙ.Π, Πληροφορικής & Τηλεματικής, Χαροκόπειο

ΧΡΗΣΤΟΣ ΔΙΟΥ

Επίκουρος καθηγητής, Πληροφορικής & Τηλεματικής, Χαροκόπειο

ΔΑΛΑΚΑΣ ΒΑΣΙΛΕΙΟΣ

Ε.ΔΙ.Π, Τμήμα, Πληροφορικής & Τηλεματικής, Χαροκόπειο

Ο/Η Άγγελος Λίχας

δηλώνω υπεύθυνα ότι:

- 1) Είμαι ο κάτοχος των πνευματικών δικαιωμάτων της πρωτότυπης αυτής εργασίας και από όσο γνωρίζω η εργασία μου δε συκοφαντεί πρόσωπα, ούτε προσβάλλει τα πνευματικά δικαιώματα τρίτων.
- 2) Αποδέχομαι ότι η ΒΚΠ μπορεί, χωρίς να αλλάξει το περιεχόμενο της εργασίας μου, να τη διαθέσει σε ηλεκτρονική μορφή μέσα από τη ψηφιακή Βιβλιοθήκη της, να την αντιγράψει σε οποιοδήποτε μέσο ή/και σε οποιοδήποτε μορφότυπο καθώς και να κρατά περισσότερα από ένα αντίγραφα για λόγους συντήρησης και ασφάλειας.

ΑΦΙΕΡΩΣΗ

Αφιερώνω αυτήν την διπλωματική εργασία στην οικογένεια μου για την στήριξη που μου έδωσαν, στους καθηγητές μου για τις γνώσεις που μου έδωσαν και στο Πανεπιστήμιο γιατί χωρίς αυτό δεν θα είχα την ευκαιρία να εξελιχθώ τόσο.

ΕΥΧΑΡΙΣΤΙΕΣ

Θα ήθελα να ευχαριστήσω όλους του καθηγητές του Χαροκοπείου Πανεπιστημίου, που μου πρόσφεραν τις γνώσεις τους και τις εμπειρίες τους με τον καλύτερο δυνατό τρόπο. Για την παρούσα διπλωματική εργασία θα ήθελα να ευχαριστήσω τον κύριο Τσαδήμα Ανάργυρο γιατί χωρίς τις συμβουλές και τις υποδείξεις του δεν είχα φτάσει σε αυτό το σημείο. Δείχνοντας μεγάλη υπομονή ως προς τα λάθη μου καταφέραμε να φέρουμε εις πέρας αυτό το έργο με επιτυχία και τον ευχαριστώ για την συνεργασία αυτή.

ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

Περίληψη στα Ελληνικά	8
Abstract	9
Κατάλογος Εικόνων	10
Κεφάλαιο 1 ^ο Εισαγωγή.....	12
1.1. Λειτουργικές Απαιτήσεις.....	12
1.2. Επεξήγηση Απαιτήσεων	13
1.3. Προσέγγιση.....	14
1.4. Δομή Κειμένου	15
2. Κεφάλαιο 2 ^ο Ανάλυση και Σχεδίαση.....	16
2.1. Ανάλυση.....	16
2.1.1. Use Case Διάγραμμα.....	17
2.1.2. Activity Διάγραμμα	18
2.2. Σχεδίαση	20
2.2.1. Class Διάγραμμα	20
2.2.2. Component Διάγραμμα	21
2.3. Προσχέδια Περιβάλλοντος εφαρμογής	22
2.3.1. Αρχική σελίδα.....	22
2.3.2. Ραντεβού χρήστη - φορέα	22
2.3.3. Login	23
3. Κεφάλαιο 3 ^ο Τεχνολογικό Πλαίσιο	24
3.1. React	24
3.2. Bootstrap	25
3.3. CSS	26
3.4. MySQL.....	27
3.5. Keycloak.....	28
3.6. Docker.....	29
4. Κεφάλαιο 4 ^ο Συστατικά στοιχεία και Επεξήγηση	30
4.1. Εγκατάσταση Απαιτούμενων Προγραμμάτων	30
4.1.1. Εγκατάσταση Node JS	30
4.1.2. Εγκατάσταση React	31
4.1.3. Εγκατάσταση MySQL και phpMyAdmin	32
4.1.4. Εγκατάσταση Docker.....	33
4.2. Δημιουργία Βάσης Δεδομένων	35
4.2.1. Δημιουργία Βάσης	35
4.2.2. Δημιουργία Πινάκων.....	35
4.3. Εγκατάσταση και ρύθμιση Keycloak	37
4.3.1. Εγκατάσταση Keycloak Container	37
4.3.2. Βασικές ρυθμίσεις Keycloak	38
4.4. Δημιουργία API Server	41
4.5. Δημιουργία εφαρμογής.....	44
4.5.1. Αρχική σελίδα.....	44
4.5.2. Σελίδα Εισόδου / Εγγραφής	45
4.5.3. Σελίδα Χρήστη.....	47
4.5.3.1. Εμφάνιση Ραντεβού	47

4.5.3.2.	Ημερολόγιο Ραντεβού	50
4.5.3.3.	Δημιουργία Ραντεβού	51
4.5.4.	Σελίδα Φορέα	52
4.5.4.1.	Διαχωρισμός Φορέα.....	52
4.5.4.2.	Ακύρωση Ραντεβού	53
4.5.5.	Αποσύνδεση	54
5.	Κεφάλαιο 5 ^ο Σενάρια Χρήσης	55
5.1.	Αρχική Σελίδα	55
5.2.	Είσοδος χρήστη/φορέα στο σύστημα	56
5.3.	Εγγραφή Χρήστη.....	57
5.4.	Σελίδα Χρήστη	58
5.5.	Δημιουργία Νέου Ραντεβού	58
5.6.	Ακύρωση Ραντεβού	60
5.7.	Αποσύνδεση Χρήστη	61
5.8.	Σελίδα Φορέα	61
5.9.	Επιβεβαίωση Ραντεβού.....	62
5.10.	Προβολή Αρχείου Ραντεβού	62
6.	Κεφάλαιο 6 ^ο Μελλοντική Εξέλιξη και Συμπεράσματα	63
6.1.	Μελλοντική Εξέλιξη	63
6.2.	Συμπεράσματα	63
6.3.	Συνεισφορά	64
7.	Βιβλιογραφία	66

Περίληψη στα Ελληνικά

Ο σκοπός της συγκεκριμένης διπλωματικής εργασίας ήταν να δημιουργήσουμε μία ηλεκτρονική πλατφόρμα διαχείρισης ραντεβού για τους φορείς του δημοσίου. Συγκεκριμένα κατασκευάσαμε πολλαπλά συστήματα που επικοινωνούν μεταξύ τους, έτσι ώστε να επιτρέπουν στους χρήστες να κλείνουν εύκολα το ραντεβού τους στον φορέα που επιθυμούν. Αντίθετα οι φορείς μπορούν εύκολα να δουν τα ραντεβού τους ώστε να οργανώσουν την ημέρα τους. Αναλύοντας το πρόβλημα που κλήθηκε να επιλύσει το σύστημα μας, εξηγήσαμε τις απαιτήσεις των χρηστών και τις προτάσεις για την ικανοποίηση αυτών. Αναφερθήκαμε στις τεχνολογίες που χρησιμοποιήσαμε και στον λόγο για τον οποίο τις επιλέξαμε. Παρουσιάσαμε τα συστήματα που υλοποιήσαμε και χρησιμοποιήσαμε με την χρήση HTML, React JS, MySQL και Express δείχνοντας τα κομμάτια κώδικα που χρησιμοποιήσαμε. Δείξαμε τον τρόπο με τον οποίο στήθηκε το κάθε σύστημα μας, τον προγραμματισμό τους, την βάση δεδομένων μας και την σύνδεση όλων αυτών μεταξύ τους. Μέσα από σενάρια χρήσης βοηθήσαμε τον εκάστοτε χρήστη να καταλάβει πως να λειτουργήσει το περιβάλλον του συστήματος μας. Αξιολογήσαμε την υλοποίηση μας με βάση τις απαιτήσεις για το ποσοστό ικανοποίησης τους. Τέλος αναφερθήκαμε στην μελλοντική εξέλιξη του συστήματος, στα συμπεράσματα τα οποία λάβαμε αλλά και στις γνώσεις και εμπειρίες που αποκτήσαμε μέσα από αυτήν την διπλωματική εργασία.

Λέξεις κλειδιά: Προγραμματισμός, Σύστημα, Σύνδεση, Απαιτήσεις, Πληροφορίες

Abstract

The purpose of this dissertation was to create an electronic appointment management platform for public institutions. Specifically, we have built multiple systems that communicate with each other, to allow users to easily book an appointment with the institution they want. On the contrary, the organizers can easily see their appointments to organize their day. Analyzing the problem that our system was called to solve, we explained the requirements of the users and the suggestions for their satisfaction. We talked about the technologies we used and the reason why we chose them. We presented the systems we implemented and used using HTML, React JS, MySQL and Express showing the code snippets we used. We showed how each of our systems was set up, their programming, our database and the connection of all of them to each other. Through usage scenarios, we helped each user to understand how our system environment works. We evaluated our implementation based on the requirements for their satisfaction rate. Finally, we referred to the future evolution of the system, the conclusions we received but also to the knowledge and experiences gained through this dissertation.

Keywords: Programming, System, Connection, Requirements, Information

Κατάλογος Εικόνων

Εικ. 1 Use Case Διάγραμμα	17
Εικ. 2 Activity Διάγραμμα Δημιουργίας Ραντεβού	18
Εικ. 3 Activity Διάγραμμα Προβολής Ραντεβού.....	19
Εικ. 4 Activity Διάγραμμα Λειτουργιών	19
Εικ. 5 Class Διάγραμμα χρήστη User.....	20
Εικ. 6 Class Διάγραμμα χρήστη Carrier	20
Εικ. 7 Component Διάγραμμα.....	21
Εικ. 8 Προσχέδιο Αρχικής Σελίδας	22
Εικ. 9 Προσχέδιο σελίδα ραντεβού.....	22
Εικ. 10 Προσχέδιο Εισόδου στο σύστημα.....	23
Εικ. 11 Λογότυπο React.....	24
Εικ. 12 Λογότυπο Bootstrap.....	25
Εικ. 13 Λογότυπο CSS	26
Εικ. 14 Λογότυπο MySQL	27
Εικ. 15 Λογότυπο Keycloak.....	28
Εικ. 16 Λογότυπο Docker	29
Εικ. 17 Ιστοσελίδα Node JS	30
Εικ. 18 Αρχική σελίδα νέας εφαρμογής React.....	31
Εικ. 19 Ιστοσελίδα Xampp.....	32
Εικ. 20 Πίνακας ελέγχου Xampp	32
Εικ. 21 Ιστοσελίδα Docker.....	33
Εικ. 22 Αρχική οθόνη εφαρμογής Docker Desktop.....	33
Εικ. 23 Containers Docker Desktop.....	34
Εικ. 24 Στιγμιότυπο πλαϊνού μενού phpMyAdmin	35
Εικ. 25 Container Keycloak.....	37
Εικ. 26 Προσθήκη περιβάλλοντος στο Keycloak.....	38
Εικ. 27 Επιλογή Clients στο Keycloak	38
Εικ. 28 Επιλογή Roles στο Keycloak	39
Εικ. 29 Επιλογή Users στο Keycloak	40
Εικ. 30 Δημιουργία κωδικού πρόσβασης για χρήστη	40
Εικ. 31 Αρχική σελίδα εφαρμογής	44
Εικ. 32 Σελίδα εισόδου στο σύστημα.....	45
Εικ. 33 Σελίδα Χρήστη	47
Εικ. 34 Ημερολόγιο για νέο ραντεβού	50
Εικ. 35 Σελίδα φορέα	52
Εικ. 36 Όνομα χρήστη και κουμπί αποσύνδεσης	54
Εικ. 37 Βασική οθόνη εφαρμογής.....	55
Εικ. 38 Πληροφορίες εφαρμογής.....	55
Εικ. 39 Κουμπί εισόδου/εγγραφής	56
Εικ. 40 Συμπλήρωση φόρμας εισόδου	56
Εικ. 41 Μήνυμα λάθους κατά την είσοδο.....	56
Εικ. 42 Κουμπί εγγραφής	57
Εικ. 43 Φόρμα εγγραφής	57
Εικ. 44 Μενού χρήστη	58

Εικ. 45 Επιλογή δημιουργία νέου ραντεβού.....	58
Εικ. 46 Επιλογή φορέα	58
Εικ. 47 Επιλογή ημερομηνίας ραντεβού	59
Εικ. 48 Διαθέσιμες και μη ώρες ραντεβού	59
Εικ. 49 Φόρμα νέου ραντεβού	59
Εικ. 50 Κουμπί αποστολής	60
Εικ. 51 Λίστα επιβεβαιωμένων ραντεβού	60
Εικ. 52 Κουμπί ακύρωσης	60
Εικ. 53 Φόρμα ακύρωσης ραντεβού	60
Εικ. 54 Κουμπί αποσύνδεσης.....	61
Εικ. 55 Σελίδα φορέα	61
Εικ. 56 Επιβεβαίωση ραντεβού.....	62
Εικ. 57 Κουμπί προβολής αρχείου	62

Κεφάλαιο 1^ο Εισαγωγή

Λόγω της κατάστασης που επικρατεί στις μέρες μας είναι δύσκολο οι πολίτες να βρουν ελεύθερη ώρα στους δημόσιους φορείς ώστε να ολοκληρώσουν τις υποχρεώσεις τους. Το προσωπικό μειώθηκε στο μισό για την ασφάλεια τους και πολλοί δουλεύουν απομακρυσμένα όσο μπορούν. Αυτό δημιούργησε ένα πρόβλημα συμφόρησης των ραντεβού για τους δημόσιους φορείς λόγω του μειωμένου προσωπικού. Αρκετές υπηρεσίες δημιούργησαν δικτυακές πύλες ώστε να εξυπηρετούν τις απλές απαιτήσεις των πολιτών, όμως δεν μπορούσαν να εξυπηρετήσουν όλα τα αιτήματα έτσι. Δηλαδή κάποιος πολίτης ο οποίος θα είχε ένα πολύ εξειδικευμένο αίτημα δεν μπορούσε να γνωρίζει πότε θα επιλυθεί άμεσα. Ή αν η υπηρεσία δεν επέτρεπε στον πελάτη της να κλείσει απομακρυσμένα ραντεβού και δεν υπάρχει κάποιο σύστημα ώστε να τα οργανώσει. Αυτές οι ανάγκες μας οδήγησαν στην εξερεύνηση του τρόπου λύσης των προβλημάτων των υπηρεσιών.

1.1. Λειτουργικές Απαιτήσεις

Οι βασικοί χρήστες του συστήματος είναι δύο: οι πολίτες - χρήστες του συστήματος και οι υπάλληλοι - φορείς του φορέα/υπηρεσίας. Οι χρήστες πρέπει να μπορούν να κλείσουν το ραντεβού στον φορέα που επιθυμούν εύκολα και γρήγορα. Να μπορούν να δουν τα ραντεβού τους σε πολλούς φορείς αν υπάρχουν και να ενημερώνονται για ακυρώσεις στα ραντεβού τους. Οι φορείς πρέπει να μπορούν να βλέπουν τα ραντεβού που έχουν, μαζί με τα σχόλια για το ραντεβού ή και τα επιμέρους αρχεία που χρειάζονται για το αίτημα τους. Όπως επίσης να ακυρώνουν ραντεβού στα οποία βλέπουν ότι δεν υπάρχουν σωστά στοιχεία.

Απαιτήσεις Συστήματος

Αναλύοντας το ζήτημα που είχαμε να φέρουμε εις πέρας και αφού συζητήσαμε με τους χρήστες για τις ανάγκες τους καταλήξαμε σε ένα σύνολο απαιτήσεων.

Απαιτήσεις του συστήματος:

- Να μπορεί ο χρήστης να επιλέγει τον φορέα που επιθυμεί
- Να προβάλλει τις διαθέσιμες ώρες των ραντεβού
- Να προσθέτει σχόλιο στο ραντεβού του
- Να υπάρχει δυνατότητα προσθήκης αρχείου
- Να υπάρχει δυνατότητα ακύρωσης ραντεβού
- Να ενημερώνονται οι χρήστες για ακυρώσεις
- Να υπάρχει ιστορικό με τα ραντεβού
- Να εμφανίζονται απλά και κατανοητά
- Να είναι απλό και εύχρηστο
- Να δουλεύει σε πληθώρα συσκευών
- Να υπάρχει ασφάλεια για τους χρήστες του συστήματος
- Να υπάρχει δυνατότητα επέκτασης των δυνατοτήτων

1.2. Επεξήγηση Απαιτήσεων

Ο χρήστης θα πρέπει να μπορεί εύκολα να κλείσει το ραντεβού στον φορέα που επιθυμεί. Έτσι θα πρέπει να υπάρχει μία λίστα με τους διαθέσιμους φορείς ώστε να επιλέγει αυτόν που θέλει να κλείσει ραντεβού.

Θα πρέπει να επιλέξει την ώρα που τον βολεύει από το σύστημα για το ραντεβού του. Αν εμφανίζονται όλες οι ώρες θα υπάρχει μεγάλο πρόβλημα. Θα πρέπει λοιπόν να του εμφανίζονται οι διαθέσιμες ώρες που παρέχονται για ηλεκτρονικά ραντεβού και να επιλέγει. Είναι αναγκαίο να μπορεί να προσθέσει κάποιο σχόλιο στο ραντεβού του. Όπως αν επιθυμεί συγκεκριμένο εκπρόσωπο να το αναγράψει ή να αναγράψει το λόγο του ραντεβού του. Αν χρειάζεται να μπορεί να επισυνάψει τα δικαιολογητικά του ή αρχεία που έχουν σχέση με το ραντεβού του.

Ο χρήστης και ο φορέας αντίστοιχα πρέπει να μπορούν να ακυρώνουν τα ραντεβού. Επιπλέον θα πρέπει να αναφέρετε ο λόγος ακύρωσης για καλύτερη σαφήνεια. Για τις ακυρώσεις θα πρέπει να υπάρχει παραπάνω ενημέρωση των χρηστών. Με το που γίνεται η ακύρωση να ενημερώνεται για τον λόγο με email ώστε να ρυθμιστεί το πρόγραμμα και να μην υπάρχουν θέματα.

Είναι πολύ χρήσιμο να υπάρχει ιστορικό των ραντεβού, έτσι ώστε να μπορούν εύκολα να βρουν πότε ήταν το τελευταίο ραντεβού με τον συγκεκριμένο φορέα και τι αφορούσε. Οι πληροφορίες πρέπει να εμφανίζονται στους χρήστες απλά. Να μην υπάρχει πάρα πολύ πληροφορία και μπερδεύεται ο εκάστοτε χρήστης. Να μπορεί να καταλάβει εύκολα πότε είναι το ραντεβού, με ποιον και τι αφορά.

Το σύστημα θα πρέπει να είναι εύχρηστο. Να μην χρειάζεται να ακολουθήσει πολλά βήματα ώστε να κάνει αυτό που επιθυμεί ο χρήστης.

Χρειαζόμαστε ένα περιβάλλον χρήσης που να μπορεί να δουλεύει σε μεγάλη γκάμα συσκευών. Για αυτό το λόγο πρέπει να είναι ελαφρύ και εύκολο στην χρήση, ώστε να μπορούν να το λειτουργήσουν όλο και περισσότεροι χρήστες.

Είναι πολύ σημαντικό το σύστημα να είναι ασφαλές. Δηλαδή να μην μπορεί να μπει κάποιος ξένος χρήστης στο σύστημα και να κάνει αλλαγές. Να μπορούν μόνο εξουσιοδοτημένοι χρήστες να κάνουν τις απαραίτητες μεταβολές στο σύστημα.

Τέλος, χρειαζόμαστε ένα σύστημα, στο οποίο να μην παρέλθει ο χρόνος ανάπτυξής του σε μικρό χρονικό διάστημα. Να είναι δυνατό να γίνουν αλλαγές, βελτιώσεις και προσθήκες ανά πάσα στιγμή. Χωρίς περιορισμούς ως προς τις δυνατότητες του.

1.3. Προσέγγιση

Θεωρώντας αυτές ως βασικό κριτήριο για την επίλυση του ζητήματος επιλέξαμε ως τρόπο επίλυσης πολλαπλά συστήματα από την αρχή. Τα συστήματα αυτά θα επικοινωνούν μεταξύ τους με συγκεκριμένα πρωτόκολλα για μεγαλύτερη ασφάλεια και θα εμφανίζουν στους χρήστες τις απαραίτητες πληροφορίες. Τα συστήματα αυτά θα λειτουργούν από το cloud έτσι ώστε να επικοινωνούν με μεγάλη ταχύτητα μεταξύ τους και να μπορεί εύκολα να αποκατασταθεί η σύνδεση τους σε περίπτωση που δεν λειτουργεί κάποιος. Με αυτό τον τρόπο εξασφαλίζουμε το ότι το σύστημα θα λειτουργεί συνέχεια χωρίς να χρειάζεται ρύθμιση από την αρχή.

Το πρώτο σύστημα μας είναι η βάση της εφαρμογής. Η βάση είναι υπεύθυνη για την αποθήκευση των δεδομένων μας, όπως τα στοιχεία για τα ραντεβού, τα στοιχεία των φορέων και άλλες πληροφορίες που χρειάζεται η εφαρμογή μας ώστε να λειτουργεί. Επιλέξαμε να στήσουμε την βάση μας με MySQL ώστε να είναι ελαφρύ και να μπορεί να χρησιμοποιηθεί εύκολα από τα άλλα συστήματα.

Το δεύτερο σύστημα θα είναι ο εξυπηρετητής της εφαρμογής μας. Ο εξυπηρετητής είναι υπεύθυνος για τις κλήσεις της κύριας εφαρμογής. Όταν θα καλείτε θα φέρνει όλες τις απαραίτητες πληροφορίες από την βάση που του ζητείτε για να τα εμφανίσει στον χρήστη. Το συγκεκριμένο σύστημα μπορεί να παρακαμφθεί αν επιλέγαμε να φέρνουμε κατευθείαν τα στοιχεία από την βάση στον χρήστη. Αυτό όμως δεν προτείνεται ούτε από οικονομία πόρων, αλλά ούτε από θέμα επιδόσεων. Καλώντας συνέχεια την βάση μέσω της κύριας εφαρμογής θα προκαλούσε μεγάλο πρόβλημα στην επίδοση της καθιστώντας την μη λειτουργική. Όπως επίσης θεωρείτε σαν παράδειγμα κακού προγραμματισμού. Η γλώσσα προγραμματισμού που επιλέξαμε για τον προγραμματισμό του εξυπηρετητή είναι Javascript. Πιο συγκεκριμένα επιλέξαμε το framework Express που λειτουργεί πάνω στο Node.js. Το παρόν σύστημα παρέχει ένα αρκετά γρήγορο και ελαφρύ περιβάλλον για τον εξυπηρετητή μας, το οποίο είναι πλήρες συμβατό με την κύρια εφαρμογή μας.

Το τρίτο σύστημα είναι ο διαχειριστής των χρηστών. Για την διαχείριση των χρηστών είναι αναγκαία η διασφάλιση των στοιχείων τους. Υπάρχουν πάρα πολλοί τρόποι που μπορούμε να επιλέξουμε για αυτή την λειτουργία, άλλοι περισσότερο ασφαλείς και άλλοι λιγότερο. Κάνοντας αρκετή έρευνα καταλήξαμε στην επιλογή του Keycloak. Το Keycloak είναι μία cloud πλατφόρμα η οποία επιτρέπει εύκολα την διαχείριση των χρηστών για το σύστημα που ενσωματώνεται. Παρέχει αρκετές λειτουργίες στον υπεύθυνο προγραμματιστή του συστήματος και πολλές παραμετροποιήσεις. Ο προγραμματιστής εύκολα ορίζει ρόλους για τους χρήστες, ρυθμίζει τις προσβάσεις αυτών και μπορεί να προσθέσει χαρακτηριστικά και παραπάνω πληροφορίες πάνω τους. Είναι επεκτάσιμο με συνεχείς ενημερώσεις και νέες λειτουργίες και πλήρως συμβατό με την κύρια εφαρμογή μας. Η αποθήκευση των χρηστών γίνεται αυτόματα στο ίδιο αυτό σύστημα με κωδικοποίηση και συγκεκριμένα πρότυπα για την ασφάλεια των στοιχείων τους.

Η κύρια εφαρμογή μας είναι υπεύθυνη για την σωστή παρουσίαση των πληροφοριών στον χρήστη. Οι χρήστες της εφαρμογής είναι δύο: οι πολίτες - χρήστες του συστήματος και οι υπάλληλοι – φορείς. Ο κάθε ένας έχει διαφορετικές οθόνες προβολής που θα του δείχνουν τα ραντεβού του, το ιστορικό του αλλά και στους χρήστες την επιλογή να κλείσουν στον φορέα που επιθυμούν νέο ραντεβού.

Έτσι δημιουργήσαμε ένα πλήθος σελίδων βασιζόμενοι σε αυτά για να μπορεί να χρησιμοποιηθεί από τους χρήστες εύκολα χωρίς την ανάγκη κάποιας επιπλέον συσκευής. Κατασκευάζοντας ένα ελαφρύ περιβάλλον με ένα εύκολο μενού χρήσης για την λειτουργία του θα μπορεί να χρησιμοποιηθεί από περισσότερο κόσμο ακόμα και από πιο αρχάριους.

Το περιβάλλον επιλέξαμε να δημιουργηθεί σε React. Είναι μία βιβλιοθήκη της Javascript για να δημιουργούνται περιβάλλοντα εφαρμογών. Επιλέξαμε την React γιατί είναι από τις πιο δημοφιλείς βιβλιοθήκες αυτή την στιγμή με πάρα πολλές δυνατότητες και λειτουργίες. Επιλέχτηκε με γνώμονα να είναι συμβατό με τα υπόλοιπα συστήματα και ελαφρύ ώστε να μην υπάρχουν απαιτήσεις πόρων αλλά ούτε εξειδικευμένα εργαλεία για την λειτουργία από τους χρήστες.

Με αυτά τα συστήματα και την επικοινωνία μεταξύ τους καταφέραμε να δημιουργήσουμε την πλατφόρμα για την διαχείριση των ραντεβού έτσι ώστε να υπάρχει καλύτερη οργάνωση. Βοηθώντας έτσι αρκετούς χρήστες μειώνοντας το άγχος και βελτιώνοντας τις συνθήκες για αυτούς.

1.4. Δομή Κειμένου

Στο 2^ο Κεφάλαιο θα δούμε την ανάλυση του συστήματος μέσα από διαγράμματα δείχνοντας καλύτερα τις συνδέσεις των συστημάτων. Στο 3^ο Κεφάλαιο θα αναλύσουμε τις τεχνολογίες που χρησιμοποιήσαμε, με πληροφορίες σχετικά με αυτές και τους λόγους για τους οποίους τις επιλέξαμε. Στο 4^ο Κεφάλαιο θα εξετάσουμε τον τρόπο με τον οποίο στήσαμε το σύστημα, πως ρυθμίσαμε τις δομές και πως το προγραμματίσαμε αναλύοντας τον κώδικα που χρησιμοποιήσαμε. Στο 5^ο Κεφάλαιο θα παρουσιάσουμε κάποια σενάρια χρήσης με το πως μπορεί ο χρήστης να χρησιμοποιήσει το περιβάλλον εξηγώντας τις λειτουργίες του. Στο 6^ο Κεφάλαιο θα αξιολογήσουμε τις απαιτήσεις του συστήματος και κατά πόσο μπορέσαμε να τις επιλύσουμε. Τέλος στο 7^ο Κεφάλαιο θα δούμε τα συμπεράσματα που καταλήξαμε επιλέγοντας αυτήν την υλοποίηση, την μελλοντική του εξέλιξη αλλά και την συνεισφορά του.

2. Κεφάλαιο 2^ο Ανάλυση και Σχεδίαση

Βρίσκοντας λοιπόν την επίλυση για το πρόβλημα μας ακολούθησε η σχεδίαση των συστημάτων. Με την σχεδίαση βλέπουμε από πριν πως συνδέονται τα συστήματα μεταξύ τους, τις λειτουργίες αυτών και τα προσχέδια για το περιβάλλον της εφαρμογής μας. Τα διαγράμματα που χρησιμοποιήσαμε δημιουργήθηκαν στην εφαρμογή Visual Paradigm [13]

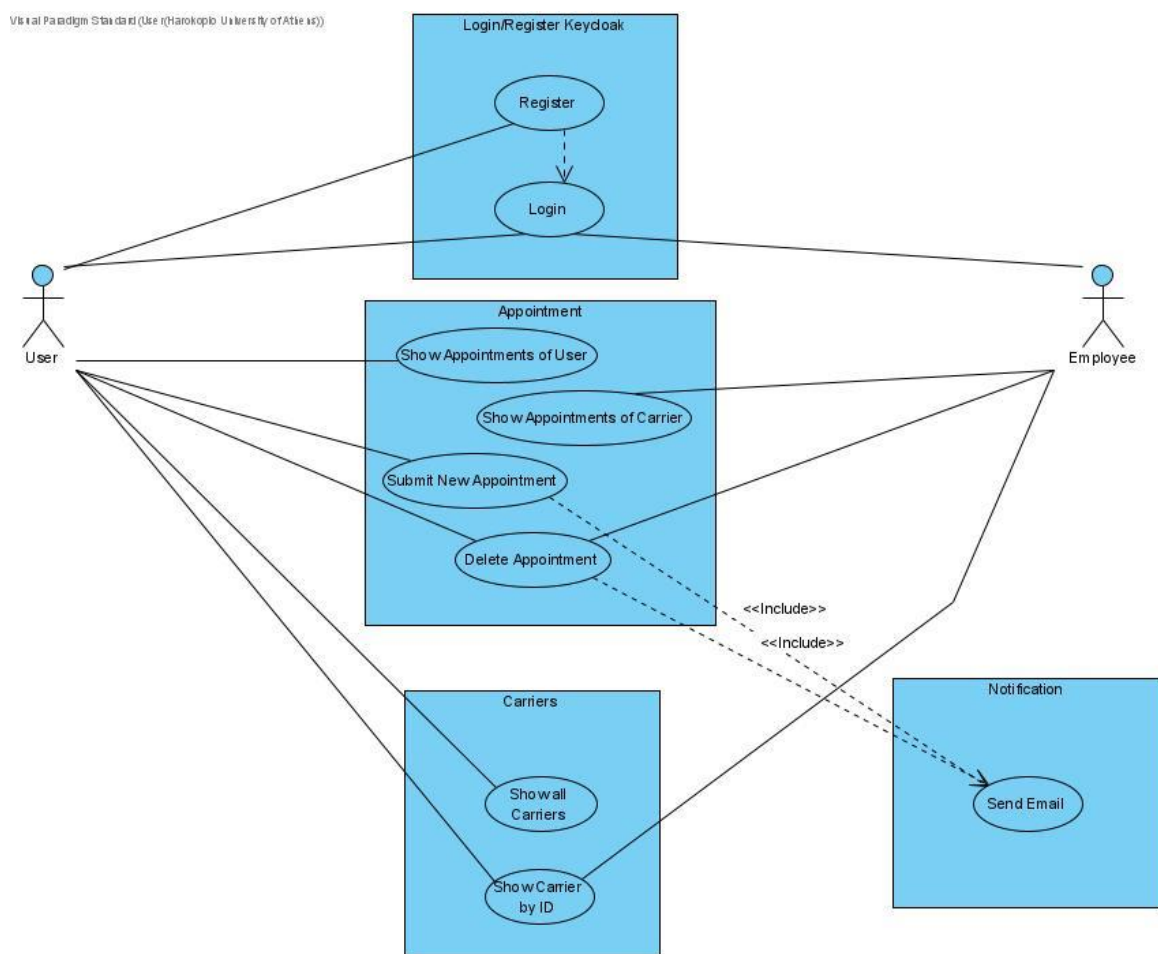
2.1. Ανάλυση

Η πλειοψηφία των μεθοδολογιών και των εργαλείων αντίστροφης μηχανίκευσης εφαρμογών παγκόσμιου ιστού βασίζονται στη UML, που είναι το πιο ευρέως αποδεκτό πρότυπο μοντελοποίησης το οποίο υιοθετείται κατά τη διαδικασία σχεδιασμού ενός λογισμικού (και για τις παραδοσιακές εφαρμογές αλλά και για τις εφαρμογές παγκόσμιου ιστού). Οι τεχνικές που βασίζονται στη UML παρέχουν ένα σταθερό και οικείο περιβάλλον εργασίας για τους μηχανικούς, προκειμένου να μοντελοποιήσουν όχι μόνο τα στοιχεία της εφαρμογής, αλλά και τη συμπεριφορά της. Αυτός είναι ο λόγος που επιλέξαμε να γίνει η ανάλυση σε διαγράμματα UML.

2.1.1. Use Case Διάγραμμα

Το use case διάγραμμα είναι υπεύθυνο για την παρουσίαση των λειτουργιών που παρέχονται από τα συστήματα στους χρήστες. Δείχνει τις σχέσεις των λειτουργιών μεταξύ τους και ποιες εξαρτώνται από ποιες άλλες, όπως και ποιες έχουν αλληλουχία. Με αυτό τον τρόπο μπορούμε να προγραμματίσουμε με ποιες λειτουργίες θα ξεκινήσουμε την υλοποίηση μας ώστε να συνεχίσουμε με αυτές που εξαρτώνται από άλλες.

Στο παρακάτω use case διάγραμμα μπορούμε να δούμε πως ο χρήστης μπορεί να κάνει εγγραφή, login, να δει τα ραντεβού του, να κλείσει νέο ραντεβού, να κάνει ακύρωση ραντεβού, να δει τους διαθέσιμους φορείς, και να δει τις διαθέσιμες ώρες συγκεκριμένου φορέα. Από την άλλη ο υπάλληλος του φορέα μπορεί να κάνει login στο σύστημα, να δει τα ραντεβού του φορέα που δουλεύει, να διαγράψει ραντεβού.

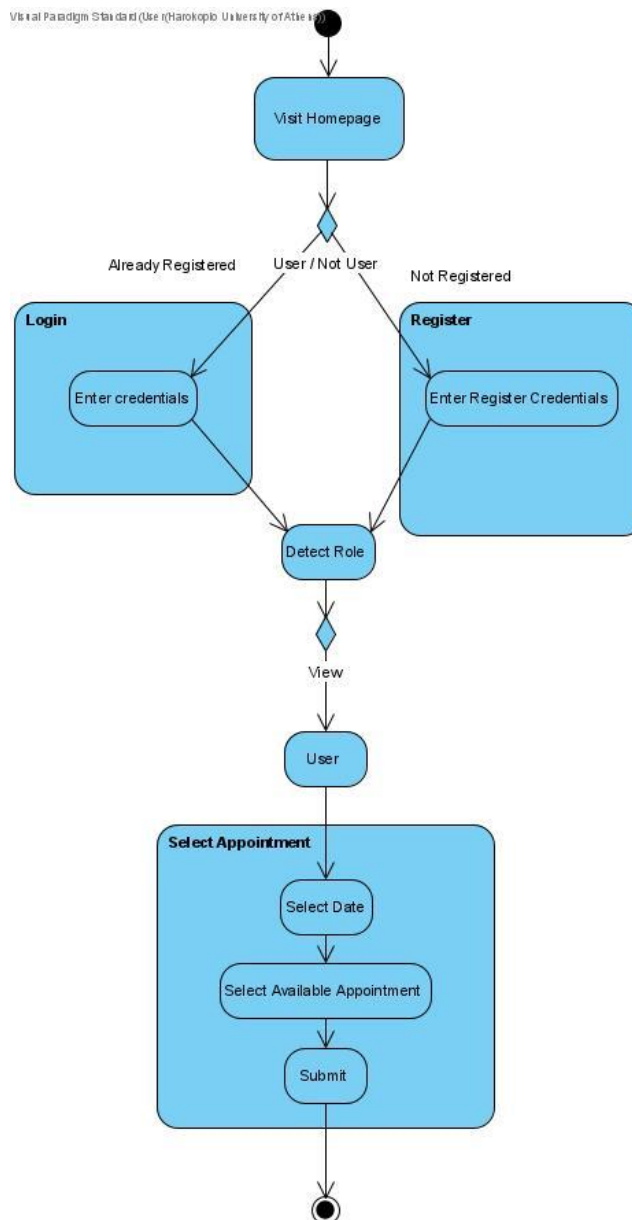


Εικ. 1 Use Case Διάγραμμα

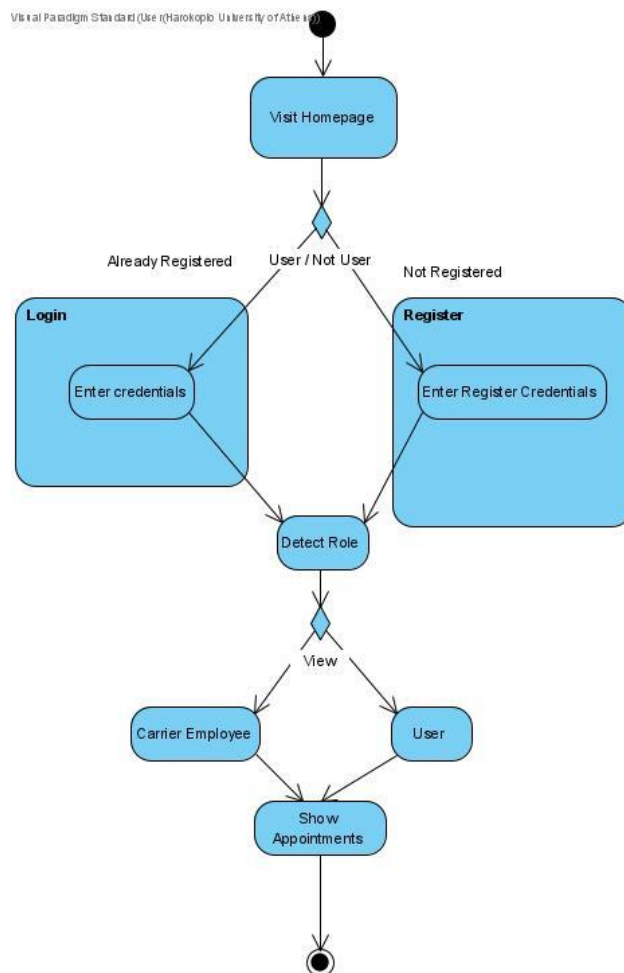
2.1.2. Activity Διάγραμμα

Το activity διάγραμμα παρουσιάζει την πορεία του χρήστη από την σύνδεση του στο σύστημα μέχρι το τέλος μία διαδικασίας. Εμφανίζει αναλυτικότερα τα βήματα που ακολουθεί στην εφαρμογή.

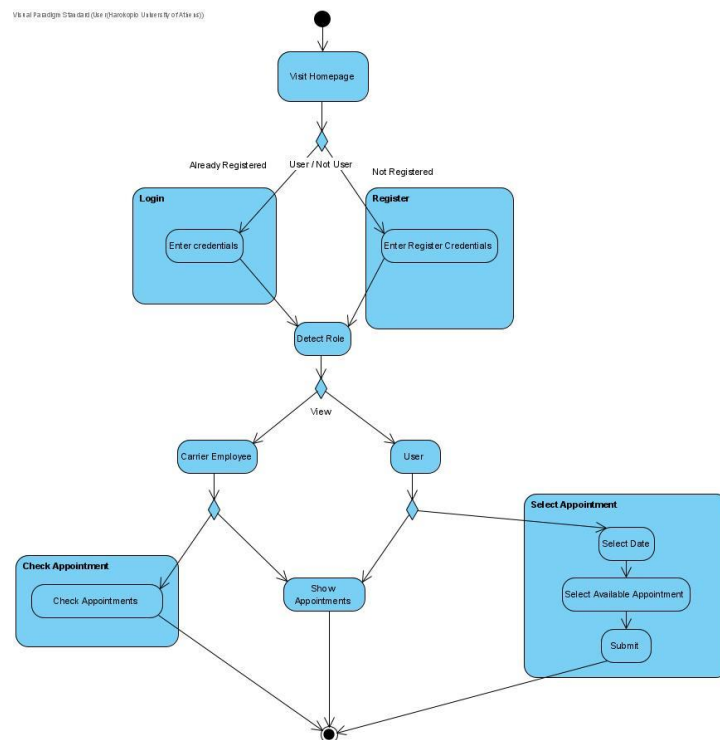
Ο χρήστης επισκέπτεται την αρχική σελίδα της εφαρμογής επιλέγει την σύνδεση ή εγγραφή, αν ο χρήστης έχει λογαριασμό συνδέεται με τα στοιχεία του, αν όχι επιλέγει την εγγραφή συμπληρώνει τα στοιχεία και γίνεται η καταχώριση. Ανάλογα τον ρόλο του αν είναι χρήστης, μπορεί να επιλέξει να δει τα ραντεβού του ή να κλείσει νέο ραντεβού. Αν είναι φορέας βλέπει τα ραντεβού και κάνει έλεγχο των ραντεβού.



Εικ. 2 Activity Διάγραμμα Δημιουργίας Ραντεβού



Εικ. 3 Activity Διάγραμμα Προβολής Ραντεβού

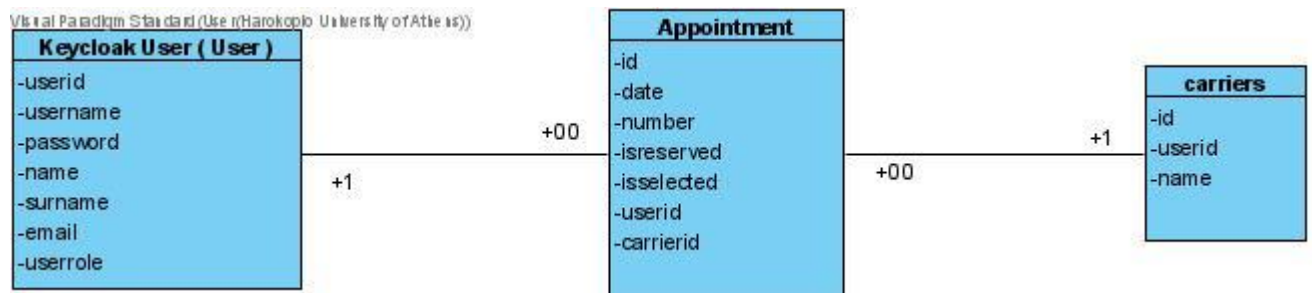


Εικ. 4 Activity Διάγραμμα Λειτουργιών

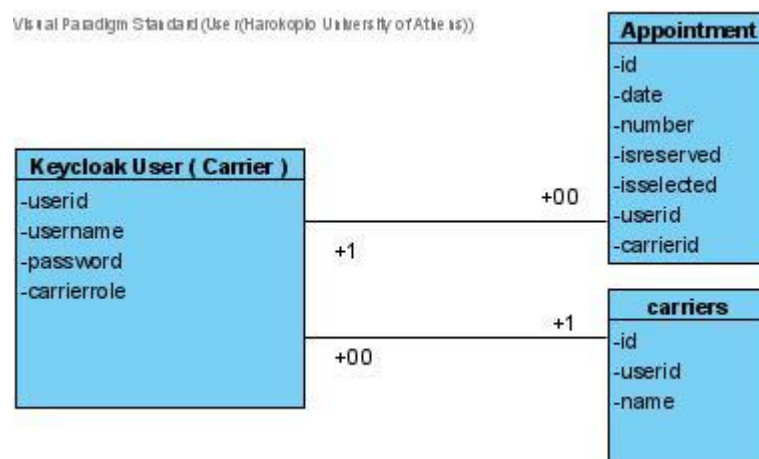
2.2. Σχεδίαση

2.2.1. Class Διάγραμμα

Στο class διάγραμμα μπορούμε να δούμε τα πεδία τα οποία αποθηκεύονται στην βάση μας και ζητάμε όταν θέλουμε να δούμε τα ραντεβού. Για το ραντεβού μπορούμε να δούμε όλα τα πεδία τα οποία καλεί ο χρήστης και αποθηκεύει και αντίστοιχα βλέπει ο φορέας για τον οποίο αντιστοιχεί το ραντεβού. Τα πεδία του χρήστη έρχονται δυναμικά από το Keycloak σύστημα.



Εικ. 5 Class Διάγραμμα χρήστη User

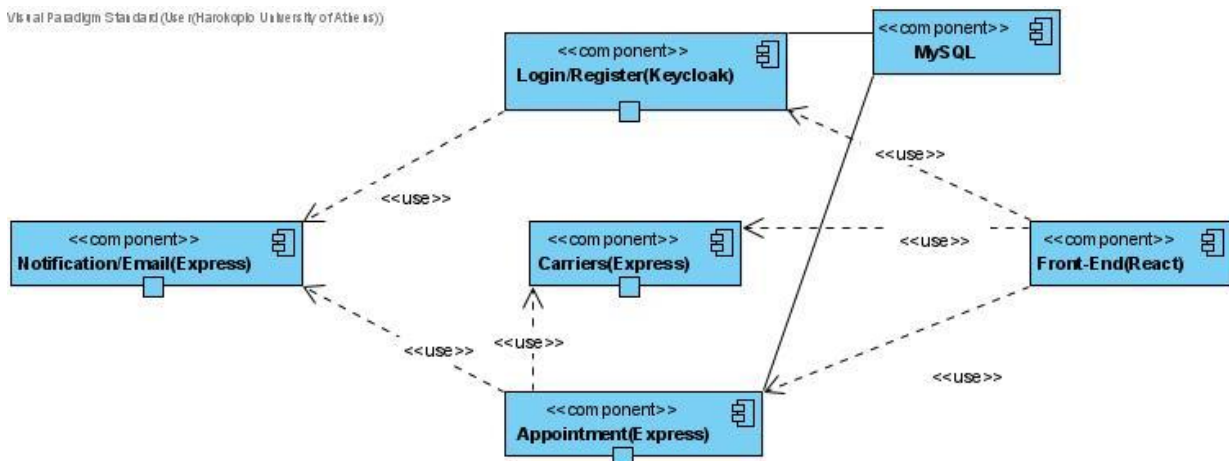


Εικ. 6 Class Διάγραμμα χρήστη Carrier

2.2.2. Component Διάγραμμα

Για να δούμε τα συστατικά(components) των συστημάτων χρησιμοποιούμε το component διάγραμμα. Παρουσιάζει το πως κάθε component χρησιμοποιεί το άλλο και πως συνδέονται και εξαρτώνται το ένα από το άλλο.

UML 2.5.1 (UML 2.5.1)



Εικ. 7 Component Διάγραμμα

2.3. Προσχέδια Περιβάλλοντος εφαρμογής

Τα προσχέδια της εφαρμογής θα μας βοηθήσουν στο πως θα παρουσιάσουμε την εφαρμογή στους χρήστες. Δείχνουν περίπου πως θα εμφανίζονται οι πληροφορίες και δίνουν μία βάση στον προγραμματισμό του περιβάλλοντος ώστε να μην προγραμματίζουμε στα τυφλά.

2.3.1. Αρχική σελίδα

Τίτλος

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla quam velit, vulputate eu pharetra nec, mattis ac neque. Duis vulputate commodo lectus, ac blandit elit tincidunt id.

Σύνδεση / Εγγραφή

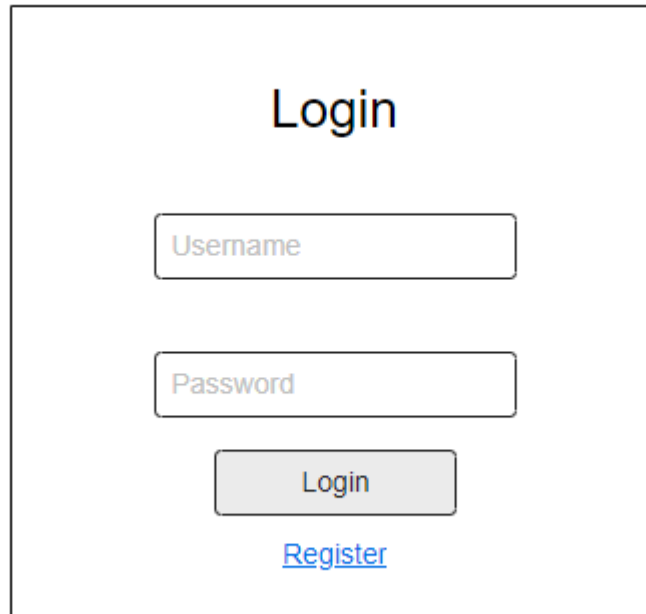
Εικ. 8 Προσχέδιο Αρχικής Σελίδας

2.3.2. Ραντεβού χρήστη - φορέα

Home	Dashboard			Αποσύνδεση
Ραντεβού	Ιστορικό	Νέο ραντεβού		
Name	Date	Actions		
Lorem ipsum dolor sit amet	July 4, 2021	Delete		
Lorem ipsum dolor sit amet	July 7, 2021	Delete		
Lorem ipsum dolor sit amet	August 10, 2021	Delete		
Lorem ipsum dolor sit amet	September 20, 2021	Delete		

Εικ. 9 Προσχέδιο σελίδα ραντεβού

2.3.3. Login



The diagram shows a login form within a rectangular border. At the top center is the title "Login". Below it are two input fields: the first is labeled "Username" and the second is labeled "Password". Below the password field is a button labeled "Login". At the bottom of the form is a blue, underlined link labeled "Register".

Εικ. 10 Προσχέδιο Εισόδου στο σύστημα

3. Κεφάλαιο 3^ο Τεχνολογικό Πλαίσιο

3.1. React [1]



Εικ. 11 Λογότυπο React

Η React είναι μια βιβλιοθήκη JavaScript ανοιχτού κώδικα front-end για τη δημιουργία διεπαφών χρήστη ή στοιχείων διεπαφής χρήστη. Συντηρείται από το Facebook και μια κοινότητα μεμονωμένων προγραμματιστών και εταιρειών. Η React μπορεί να χρησιμοποιηθεί ως βάση για την ανάπτυξη μιας σελίδας ή εφαρμογών για κινητά. Ωστόσο, το React ασχολείται μόνο με τη διαχείριση της κατάστασης και την απόδοση αυτής της κατάστασης στο DOM, επομένως η δημιουργία εφαρμογών React απαιτεί συνήθως τη χρήση πρόσθετων βιβλιοθηκών για δρομολόγηση, καθώς και ορισμένες λειτουργίες από την πλευρά του πελάτη.

Η React επιτρέπει στους προγραμματιστές να δημιουργούν μεγάλες εφαρμογές ιστού που μπορούν να αλλάζουν δεδομένα, χωρίς να επαναφορτώνουν τη σελίδα. Ο κύριος σκοπός της React είναι να είναι γρήγορος, επεκτάσιμος και απλός. Λειτουργεί μόνο σε διεπαφές χρήστη στην εφαρμογή. Αυτό αντιστοιχεί στην προβολή στο πρότυπο MVC. Μπορεί να χρησιμοποιηθεί με συνδυασμό άλλων βιβλιοθηκών ή πλαισίων JavaScript, όπως το Angular JS στο MVC.

Οι λόγοι επιλογής της React για την εφαρμογή μας ήταν αρκετοί. Θα μπορούσαμε να διαλέξουμε κάποια άλλη γλώσσα με διαφορετική βιβλιοθήκη. Αυτό όμως που μας κέρδισε είναι ότι είναι απλή, χρησιμοποιεί μια ειδική σύνταξη που ονομάζεται JSX που αποτελείται HTML με JavaScript. Σε σύγκριση με παρόμοιες βιβλιοθήκες όπως η Angular είναι πολύ εύχρηστη και εύκολα επεκτάσιμη. Χρειάζεται μόνο γνώσεις CSS και HTML ώστε κάποιος να μπορεί να φτιάξει εφαρμογές για όλων των ειδών συσκευές. Επιπλέον οι επιδόσεις της εφαρμογής μας μπορούν να τροποποιηθούν όπως επιθυμούμε και να γίνει το απαραίτητο τεστ ώστε να είναι σταθερή η εφαρμογή χωρίς σφάλματα.

3.2. Bootstrap [4][5]



Εικ. 12 Λογότυπο Bootstrap

Το Bootstrap είναι ένας συνδυασμός HTML, CSS και Javascript που έχουν σχεδιαστεί ως πλαίσιο ανοιχτού κώδικα για τη δημιουργία καθαρών διεπαφών χρήστη. Το Bootstrap δημιουργήθηκε από, και διατηρείται ενεργά από την ομάδα του Twitter. Το Bootstrap έχει γίνει ένα από τις πιο δημοφιλείς βιβλιοθήκες front-end και open source στον κόσμο.

Μετά την κυκλοφορία του ανοιχτού κώδικα το 2011, το Bootstrap έγινε δημοφιλές πολύ γρήγορα και όχι χωρίς λόγο. Οι σχεδιαστές ιστοσελίδων και οι προγραμματιστές ιστού επιλέγουν το Bootstrap επειδή είναι ευέλικτο και εύκολο στη χρήση. Τα κύρια πλεονεκτήματά του είναι ότι ανταποκρίνεται στη σχεδίαση, διατηρεί ευρεία συμβατότητα με το πρόγραμμα περιήγησης, προσφέρει συνεπή σχεδίαση χρησιμοποιώντας επαναχρησιμοποιήσιμα στοιχεία και είναι πολύ εύκολο στη χρήση και γρήγορο να μάθει. Προσφέρει πλούσια επεκτασιμότητα χρησιμοποιώντας JavaScript, συνοδευόμενη από ενσωματωμένη υποστήριξη για προσθήκες jQuery και προγραμματικό JavaScript API.

Το Bootstrap μας βοήθησε πάρα πολύ στην εφαρμογή μας ώστε να έχουμε έναν κορμό στην σχεδίαση μας. Δηλαδή χρησιμοποιώντας τα προφίλ του μπορέσαμε να στήσουμε σχεδιαστικά μία βάση και έχοντας έτοιμα εργαλεία να αναπτύξουμε πιο εύκολα τις λειτουργίες της εφαρμογής. Στην συνέχεια χρησιμοποιήσαμε CSS για να φέρουμε τις δικές μας σχεδιαστικές αλλαγές για να δημιουργήσουμε ένα πιο οικείο περιβάλλον.

3.3. CSS



Εικ. 13 Λογότυπο CSS

Η CSS (Cascading Style Sheets - Διαδοχικά Φύλλα Ύφους) ή (αλληλουχία φύλλων ύφους) είναι μια γλώσσα υπολογιστή που ανήκει στην κατηγορία των γλωσσών φύλλων ύφους που χρησιμοποιείται για τον έλεγχο της εμφάνισης ενός εγγράφου που έχει γραφτεί με μια γλώσσα σήμανσης.

Χρησιμοποιείται δηλαδή για τον έλεγχο της εμφάνισης ενός εγγράφου που γράφτηκε στις γλώσσες HTML και XHTML, δηλαδή για τον έλεγχο της εμφάνισης μιας ιστοσελίδας και γενικότερα ενός ιστότοπου. Η CSS είναι μια γλώσσα υπολογιστή προορισμένη να αναπτύσσει στιλιστικά μια ιστοσελίδα δηλαδή να διαμορφώνει περισσότερα χαρακτηριστικά, χρώματα, στοίχιση και δίνει περισσότερες δυνατότητες σε σχέση με την html. Για μια όμορφη και καλοσχεδιασμένη ιστοσελίδα η χρήση της CSS κρίνεται ως απαραίτητη.

3.4. MySQL



Εικ. 14 Λογότυπο MySQL

Η MySQL είναι ένα σύστημα διαχείρισης σχεσιακών βάσεων δεδομένων ανοικτού κώδικα που χρησιμοποιεί την SQL (Structured Query Language). Οποιοσδήποτε μπορεί να κατεβάσει την MySQL και να την διαμορφώσει με βάση τις ανάγκες του.

Η MySQL είναι γνωστή για την ταχύτητα, την αξιοπιστία και επεκτασιμότητα που παρέχει. Μπορεί να λειτουργήσει σε περιβάλλοντα Windows, Unix, Linux στα οποία τρέχει ένας εξυπηρετητής δίνοντας πρόσβαση στους χρήστες σε ένα σύνολο βάσεων δεδομένων. Χρησιμοποιείτε από δημοφιλείς ιστοσελίδες και διαδικτυακά προγράμματα και υπηρεσίες όπως το Facebook, Google, Wikipedia. Τα δεδομένα αποθηκεύονται σε πίνακες και αποτελούνται από στήλες και γραμμές.

Η διαχείριση των δεδομένων γίνεται από εντολές queries. Με την MySQL μπορούμε να αναζητήσουμε, να προσθέσουμε, διαγράψουμε και να τροποποιήσουμε τα δεδομένα μίας βάσης δεδομένων και να τα επιστρέψουμε στην ζήτηση από τα queries.

Η βάση για την εφαρμογή μας στήθηκε σε MySQL λόγω της πλήρους συμβατότητας με τα άλλα συστήματα. Έχει την βέλτιστη απόδοση για τις λειτουργίες που επιθυμούμε και προσφέρει μεγάλο πλήθος χαρακτηριστικών ασφαλείας που ήταν αναγκαία.

3.5. Keycloak [2]



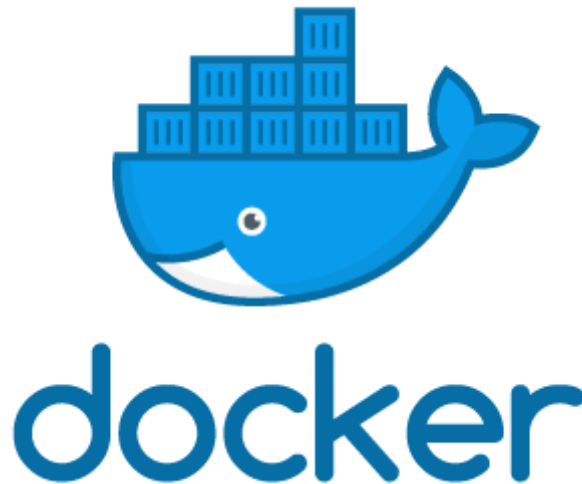
Εικ. 15 Λογότυπο Keycloak

Το Keycloak είναι ένα σύστημα διαχείρισης ταυτότητας και πρόσβασης, ανοιχτού κώδικα που στοχεύει σε σύγχρονες εφαρμογές και υπηρεσίες. Διευκολύνει την ασφάλεια εφαρμογών και υπηρεσιών με λίγο έως καθόλου κώδικα. Οι χρήστες κάνουν έλεγχο ταυτότητας με το Keycloak και όχι μεμονωμένες εφαρμογές. Αυτό σημαίνει ότι οι εφαρμογές δεν χρειάζεται να ασχολούνται με φόρμες σύνδεσης, έλεγχο ταυτότητας χρηστών και αποθήκευση χρηστών. Μόλις γίνει η σύνδεση στο Keycloak, οι χρήστες δεν χρειάζεται να συνδεθούν ξανά για να αποκτήσουν πρόσβαση σε άλλη εφαρμογή.

Προσφέρει πάρα πολλά χαρακτηριστικά στους διαχειριστές. Μπορούν να ενεργοποιήσουν σύνδεση με κοινωνικά δίκτυα και άλλες πλατφόρμες ταυτοποίησης. Συνδέετε σε ήδη υπάρχων διακομιστές, όπως και με ξεχωριστούς παρόχους αρκεί να υπάρχει μία σχεσιακή βάση δεδομένων. Από την κονσόλα του διαχειριστή μπορούν να παραμετροποιηθούν όλα τα χαρακτηριστικά και οι λειτουργίες του Keycloak, να γίνει η διαχείριση των χρηστών και η προσθήκη στοιχείων σε χρήστες.

Το Keycloak επιλέχθηκε ως σύστημα διαχείρισης χρηστών κυρίως για λόγους ασφαλείας. Λειτουργεί με πρωτόκολλα τα οποία μας διαβεβαιώνουν για την διασφάλιση των δεδομένων των χρηστών μας χωρίς να χρειάζεται την δική μας παρέμβαση. Μπορούμε να ορίσουμε εύκολα ρόλους για τους χρήστες, να ρυθμίσουμε τις προσβάσεις αυτών και να προσθέσουμε παραπάνω πληροφορίες πάνω τους. Είναι πλήρως συμβατό και επεκτάσιμο με τα υπόλοιπα συστήματά μας και παρέχει συνεχείς ενημερώσεις και νέες λειτουργίες συνέχεια.

3.6. Docker [14]



Εικ. 16 Λογότυπο Docker

Η ανάπτυξη εφαρμογών σήμερα απαιτεί πολύ περισσότερα από το γράψιμο κώδικα. Πολλές γλώσσες, πλαίσια, αρχιτεκτονικές και ασυνεχείς διεπαφές μεταξύ εργαλείων για κάθε στάδιο κύκλου ζωής δημιουργούν τεράστια πολυπλοκότητα. Το Docker απλοποιεί και επιταχύνει τη ροή εργασίας σας, ενώ δίνει στους προγραμματιστές την ελευθερία να καινοτομούν με την επιλογή των εργαλείων, των στοιβών εφαρμογών και των περιβαλλόντων ανάπτυξης για κάθε έργο. Χρησιμοποιεί container που βασίζονται στο host λειτουργικό. Χρησιμοποιεί τον πυρήνα του host. Δεν έχει δικό του πυρήνα. Με αποτέλεσμα να είναι πολύ ελαφρύ και να ξοδεύει μόνο τα αναγκαία resources, για να τρέξουν τα services που θέλει. Κάθε container είναι πλήρως απομονωμένο από το host λειτουργικό και από άλλα containers. Με το Docker μπορούμε να στήσουμε την εφαρμογή μας για πολλές χρήσεις περιγράφοντας το container και όλες τις υπηρεσίες που χρειάζεται για να είναι λειτουργική.

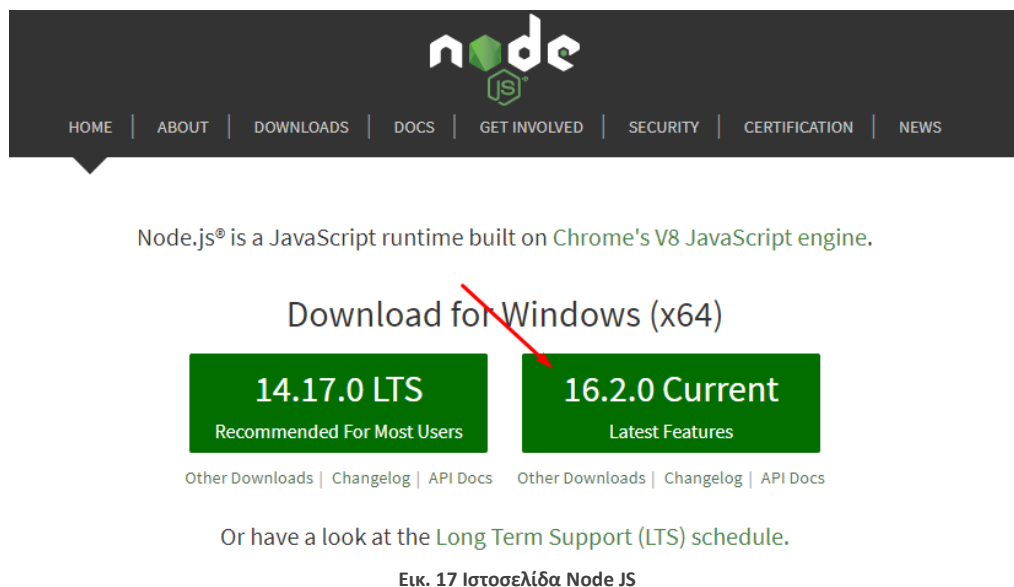
Το Docker είναι ο βασική πλατφόρμα που στήνουμε τα συστήματά μας. Έτσι ανεβαίνουν στο cloud και είναι προσβάσιμα συνεχώς. Μπορούμε να δούμε την λειτουργία τους ανά πάσα στιγμή και να κάνουμε αλλαγές στα τεστ περιβάλλοντα που έχουμε δημιουργήσει χωρίς να επηρεάζουμε την βασική εφαρμογή. Το ένα σύστημα συνδέεται με το άλλο εύκολα και γρήγορα ώστε να μπορεί να χρησιμοποιεί τις λειτουργίες του. Επιπλέον παρέχει πολύ γρήγορες ταχύτητες στην προσπέλαση των συστημάτων λόγω της λογικής των containers, αλλά και πολύ εύκολη αποσφαλμάτωση έχοντας στην διάθεση του την επιλογή της έκδοσης της υπηρεσίας που χρειάζεται αλλά και της απομόνωσης αυτού..

4. Κεφάλαιο 4^ο Συστατικά στοιχεία και Επεξήγηση

4.1. Εγκατάσταση Απαιτούμενων Προγραμμάτων

4.1.1. Εγκατάσταση Node JS [12]

Για να ξεκινήσουμε την ανάπτυξη της εφαρμογής μας χρειαζόμαστε το Node JS. Το Node JS είναι ο βασικός κορμός για να στήσουμε για αρχή τοπικά την εφαρμογή μας. Για να εγκαταστήσουμε το Node JS σε υπολογιστή με Windows κατευθυνόμαστε στην ιστοσελίδα του και επιλέγουμε την έκδοση που επιθυμούμε. Εμείς επιλέξαμε την τελευταία 16.2 που κυκλοφορεί



Εικ. 17 Ιστοσελίδα Node JS

Αφού ολοκληρωθεί η λήψη κάνουμε την εγκατάσταση ακολουθώντας τα βήματα του οδηγού και στο τέλος επανεκκίνηση του υπολογιστή. Με αυτή την διαδικασία έχουμε ολοκληρώσει την εγκατάσταση και το Node JS είναι διαθέσιμο.

Για επαλήθευση ανοίγουμε την γραμμή εντολών και γράφουμε την εντολή:

```
node -v
```

Και εμφανίζεται η εγκατεστημένη έκδοση.

4.1.2. Εγκατάσταση React [1]

Αφού ολοκληρώσαμε την εγκατάσταση του Node JS μπορούμε να δημιουργήσουμε την React εφαρμογή μας. Έτσι θα μπορέσουμε να ξεκινήσουμε την υλοποίηση.

Για να το κάνουμε αυτό πρέπει να κατευθυνθούμε στον φάκελο που θέλουμε να δημιουργηθεί να ανοίξουμε την γραμμή εντολών μέσα του. Έπειτα πρέπει να τρέξουμε την εντολή:

```
npx create-react-app appointment-app
```

Όπου appointment-app το όνομα της εφαρμογής και του φακέλου που θα δημιουργήσει. Με την εντολή αυτή γίνεται εγκατάσταση όλων των απαραίτητων στοιχείων που χρειάζεται για να λειτουργήσει ένα project React. Αφού ολοκληρωθεί τρέχουμε την εντολή για να μπούμε στον φάκελο και να τρέξουμε την εφαρμογή.

```
cd my-app  
npm start
```

Έτσι θα τρέξει η εφαρμογή και μπορούμε να την δούμε από το πρόγραμμα περιήγησης μας στην διεύθυνση <http://localhost:3000>.



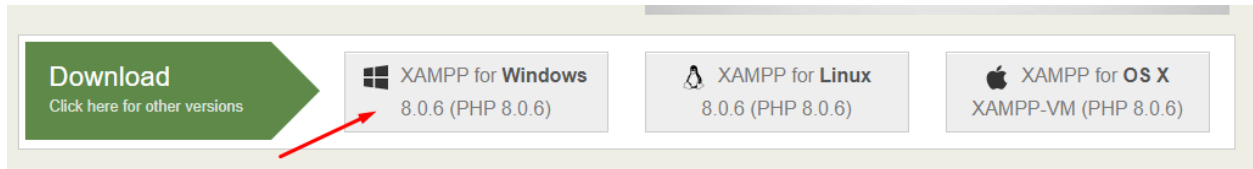
Εικ. 18 Αρχική σελίδα νέας εφαρμογής React

Έτσι μπορούμε να ξεκινήσουμε την υλοποίηση μας για το περιβάλλον χρήσης.

Με την ίδια διαδικασία δημιουργήσαμε και μία ξεχωριστή εφαρμογή React για το API της εφαρμογής μας. Το API είναι υπεύθυνο ώστε να μπορεί να φέρνει με τα κατάλληλα queries από την βάση της πληροφορίες που του ζητάμε και να τις περνάει στο περιβάλλον χρήσης. Έτσι μπορούμε να τις προβάλουμε στους χρήστες και να ανανεώνουμε συνεχώς τι βλέπουν με τις επιλογές τους.

4.1.3. Εγκατάσταση MySQL και phpMyAdmin

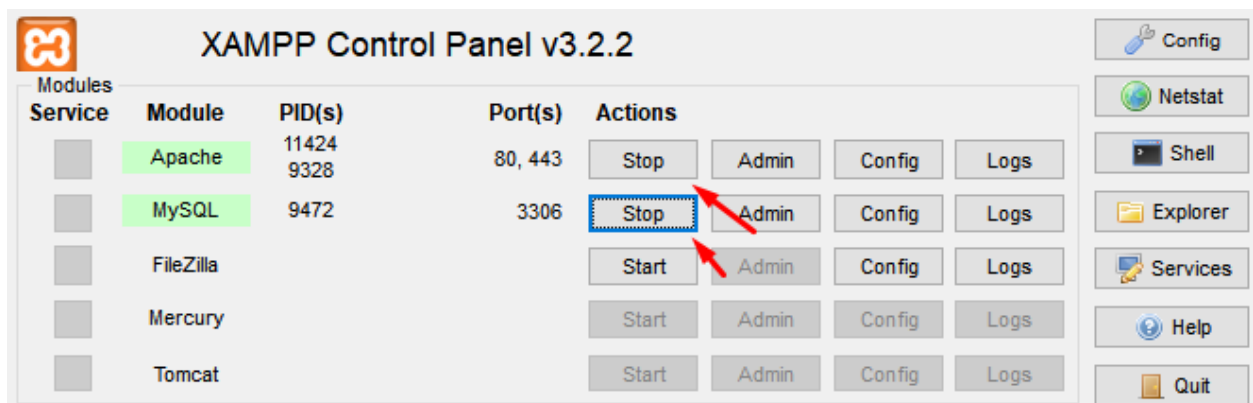
Για την εγκατάσταση MySQL σε τοπικό περιβάλλον Windows επιλέξαμε την χρήση του Xampp. Το λογισμικό Xampp είναι ένα έτοιμο εργαλείο ανάπτυξης για γλώσσα PHP, στην δική μας όμως περίπτωση περιλαμβάνει την MySQL και το phpMyAdmin για εύκολη διαχείριση της βάσης. Για να κάνουμε την εγκατάσταση κατευθυνόμαστε στην ιστοσελίδα του και κατεβάζουμε τον οδηγό εγκατάστασης.



Εικ. 19 Ιστοσελίδα Xampp

Ακολουθούμε τα βήματα της εγκατάστασης και μόλις ολοκληρωθεί ανοίγουμε το Xampp.

Στον πίνακα ελέγχου που εμφανίζεται πατάμε “start” στο Apache και MySQL και η βάση μας είναι ενεργοποιημένη.



Εικ. 20 Πίνακας ελέγχου Xampp

4.1.4. Εγκατάσταση Docker [15]

Το Docker είναι υπεύθυνο αφού ολοκληρωθεί η ανάπτυξη της εφαρμογής μας στο τοπικό περιβάλλον να το εγκαταστήσουμε και στο cloud. Μπορούμε όμως να εγκαταστήσουμε τοπικά στον υπολογιστή μας για να κάνουμε τον απαραίτητο έλεγχο πριν το ανεβάσουμε σε κάποιον διακομιστή.

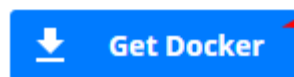
Για την εγκατάσταση του σε περιβάλλον Windows είναι απαραίτητη η εγκατάσταση του WSL2. Το WSL2 γίνεται εγκατάσταση από το Microsoft Store και λειτουργεί στο παρασκήνιο όταν κληθεί από εφαρμογές. Σε συστήματα UNIX η εγκατάσταση είναι αρκετά πιο εύκολη λόγω της υψηλότερης συμβατότητας. Έπειτα για την εγκατάσταση του Docker κατευθυνόμαστε στην ιστοσελίδα του και κατεβάζουμε τον οδηγό εγκατάστασης.

Get Docker Desktop for Windows

Docker Desktop for Windows is available for free.

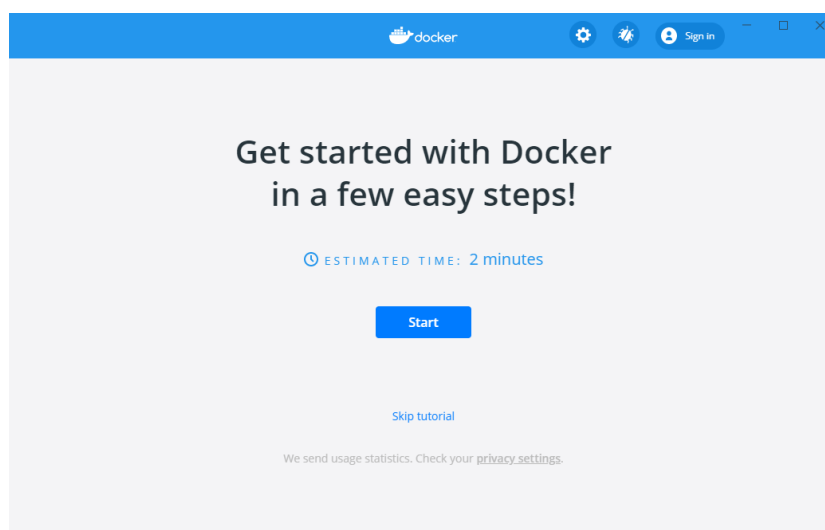
Requires Microsoft Windows 10 Professional or Enterprise 64-bit, or Windows 10 Home 64-bit with WSL 2.

By downloading this, you agree to the terms of the [Docker Software End User License Agreement](#) and the [Docker Data Processing Agreement \(DPA\)](#).



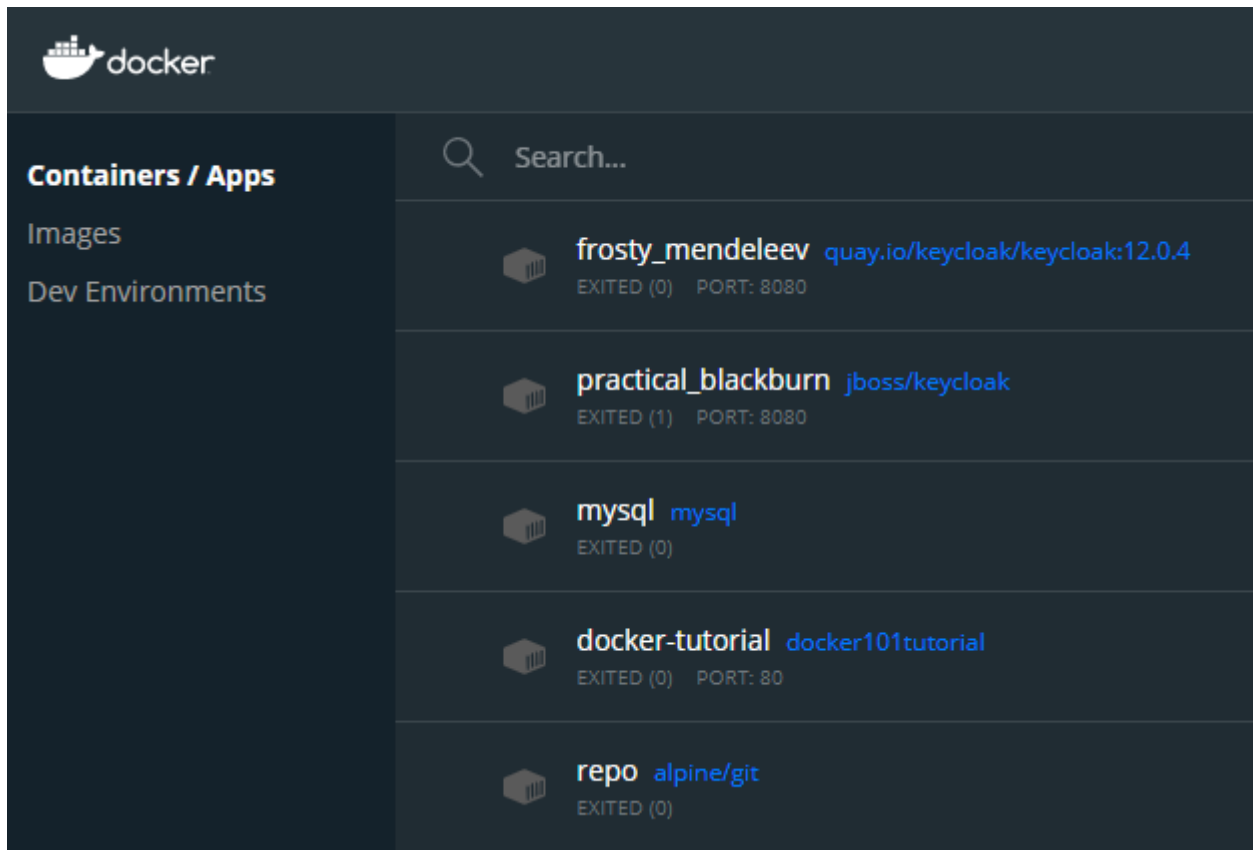
Εικ. 21 Ιστοσελίδα Docker

Τρέχουμε τον οδηγό και ακολουθώντας τα βήματα ολοκληρώνουμε την εγκατάσταση. Έπειτα ανοίγουμε την εφαρμογή Docker Desktop και ολοκληρώνουμε την εγκατάσταση.



Εικ. 22 Αρχική οθόνη εφαρμογής Docker Desktop

Αφού ολοκληρωθούν και οι ρυθμίσεις της εφαρμογής το Docker είναι έτοιμο. Μπορούμε να κάνουμε εγκατάσταση img από το Docker Hub για τα containers που επιθυμούμε. Συγκεκριμένα αφού ολοκληρώσουμε την υλοποίηση στο τοπικό περιβάλλον θα χρησιμοποιήσουμε τα δικά μας custom container που θα περιέχουν τα δικά μας συστήματα.

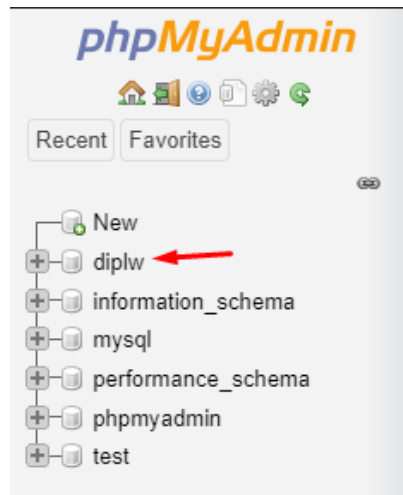


Εικ. 23 Containers Docker Desktop

4.2. Δημιουργία Βάσης Δεδομένων

4.2.1. Δημιουργία Βάσης

Αρχικά πρέπει να δημιουργήσουμε μία βάση δεδομένων που θα στεγάσει τους απαραίτητους πίνακες. Αφού κάνουμε είσοδο στο phpMyAdmin, πατάμε πάνω στο κουμπί Νέα που βρίσκεται στα αριστερά της οθόνης. Έπειτα δίνοντας ένα όνομα και επιλέγοντας την σύνθεση που επιθυμούμε πατάμε στο κουμπί Δημιουργία και η βάση μας είναι έτοιμη. Η βάση μας είναι η diplw.



Εικ. 24 Στιγμιότυπο πλαϊνού μενού phpMyAdmin

4.2.2. Δημιουργία Πινάκων

4.2.2.1. Πίνακας Appointments

Επιλέγοντας την βάση μας από τα αριστερά μας και έπειτα επιλέγοντας κώδικας SQL μπορούμε να πληκτρολογήσουμε την εντολή:

```
CREATE TABLE `appointments` (  
  `id` int(11) NOT NULL AUTO_INCREMENT,  
  `slot` datetime NOT NULL,  
  `isconfirmed` int(2) NOT NULL,  
  `carrierid` varchar(255) NOT NULL,  
  `userid` varchar(255) NOT NULL,  
  `username` varchar(255) NOT NULL,  
  `comment` longtext NOT NULL,  
  `file` varchar(255) NOT NULL,  
  PRIMARY KEY (`id`)  
) ENGINE=InnoDB AUTO_INCREMENT=82 DEFAULT CHARSET=utf8mb4
```

Για να δημιουργήσουμε το πρώτο πίνακα Appointments ο οποίος είναι υπεύθυνος για την αποθήκευση των ραντεβού. Σε αυτό τον πίνακα θα αποθηκεύονται όλα τα στοιχεία που εμφανίζονται με το ραντεβού. Κάθε γραμμή του πίνακα περιλαμβάνει ένα ραντεβού μαζί με τα στοιχεία του χρήστη που το έκλεισε, την ώρα και τον φορέα που απευθύνεται.

4.2.2.2. Πίνακας Carriers

Ο πίνακας Carriers έχει τον ρόλο της αποθήκευσης των φορέων που στεγάζονται στο σύστημα μας. Τα πεδία του πίνακα αποτελούνται από το id αυτού, το όνομα και το email. Έτσι ώστε όταν ένας υπάλληλος ενός φορέα συνδεθεί υπάρχει αντιστοίχιση με αυτόν τον πίνακα μέσω του μοναδικού αριθμού του id.

Ο πίνακας αυτός διαμορφώνεται με την εντολή:

```
CREATE TABLE `carriers` (  
  `id` int(11) NOT NULL AUTO_INCREMENT,  
  `name` varchar(255) NOT NULL,  
  `email` varchar(255) NOT NULL,  
  PRIMARY KEY (`id`)  
) ENGINE=InnoDB AUTO_INCREMENT=3 DEFAULT CHARSET=utf8mb4
```

4.3. Εγκατάσταση και ρύθμιση Keycloak [2]

Το Keycloak είναι ο διαχειριστής των χρηστών του συστήματος (χρήστες, υπάλληλοι). Είναι αναγκαία η εγκατάσταση και η ρύθμιση του πριν την υλοποίηση του περιβάλλοντος ώστε να μπορέσουμε να διαχειριστούμε τις δικές του οθόνες λειτουργίας και να τις ενσωματώσουμε αργότερα.

4.3.1. Εγκατάσταση Keycloak Container

Το Keycloak μπορεί να εγκατασταθεί είτε τοπικά σε περιβάλλον UNIX είτε σε δικό του container στο cloud. Η εφαρμογή μας είναι συμβατή με το cloud και έχοντας ήδη εγκαταστήσει το Docker τοπικά επιλέξαμε να εγκαταστήσουμε το Keycloak σε container. Η εγκατάσταση σε container είναι πολύ εύκολη λόγω του ότι υπάρχουν έτοιμες εικόνες του συστήματος από την RedHat στο GitHub. Για να γίνει η εγκατάσταση του container και η εκκίνηση αυτού τρέχουμε από ένα τερματικό του υπολογιστή μας μία εντολή με παραμέτρους. Η εντολή είναι:

```
docker run -p 8080:8080 -e KEYCLOAK_USER=admin -e KEYCLOAK_PASSWORD=admin  
quay.io/keycloak/keycloak:13.0.1
```

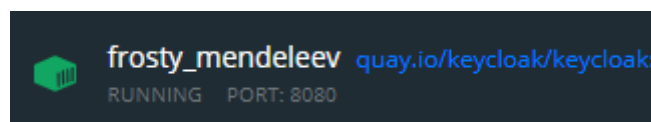
8080:8080 = το εύρος των θυρών που τρέχει το Keycloak

KEYCLOAK_USER = όνομα διαχειριστή του Keycloak

KEYCLOAK_PASSWORD = κωδικός διαχειριστή

quay.io/keycloak/keycloak:13.0.1 = η διεύθυνση από την οποία λαμβάνει τη εικόνα του συστήματος για το Keycloak. Ο αριθμός 13.0.1 είναι η έκδοση της εικόνας του συστήματος, στην συγκεκριμένη περίπτωση και είναι και η τελευταία έκδοση που κυκλοφορεί.

Αφού ολοκληρωθεί η διαδικασία λήψης, εγκατάστασης και εκκίνησης του container για το Keycloak είναι έτοιμο για την ρύθμιση του.



Εικ. 25 Container Keycloak

4.3.2. Βασικές ρυθμίσεις Keycloak

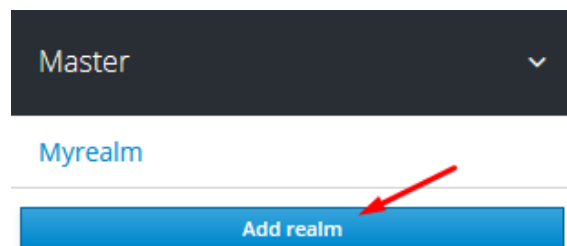
4.3.2.1. Είσοδος στον πίνακα ελέγχου

Για να εισέλθουμε στον πίνακα ελέγχου για τους διαχειριστές του Keycloak ανοίγουμε ένα πρόγραμμα περιήγησης και πληκτρολογούμε την διεύθυνση <http://localhost:8080/auth/admin>. Συνδεόμαστε με τα στοιχεία του διαχειριστή που θέσαμε στην εγκατάσταση και μας εμφανίζεται ο πίνακας διαχείρισης του Keycloak.

4.3.2.2. Δημιουργία νέου περιβάλλοντος

Στο Keycloak μας επιτρέπεται να δημιουργήσουμε πολλά περιβάλλοντα για κάθε εφαρμογή χωρίς να χρειαζόμαστε νέα εγκατάσταση κάθε φορά. Αυτό γίνεται με την δημιουργία ενός Realm για την εφαρμογή που θέλουμε να το συσχετίσουμε.

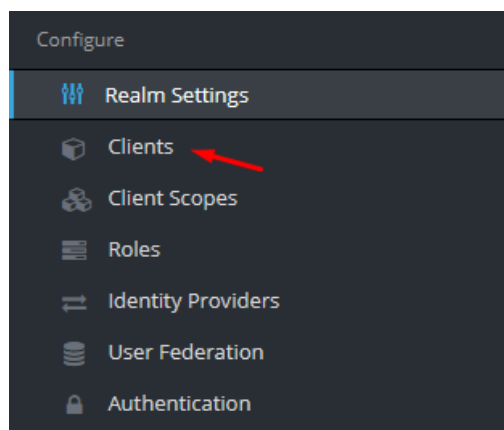
Για να δημιουργήσουμε ένα νέο περιβάλλον πατάμε το κουμπί “Add realm” πάνω αριστερά. Εισάγουμε το όνομα και πατάμε “Create”. Το νέο περιβάλλον έχει δημιουργηθεί.



Εικ. 26 Προσθήκη περιβάλλοντος στο Keycloak

4.3.2.3. Ασφάλιση εφαρμογής

Για να ασφαλίσουμε την εφαρμογή που θέλουμε να συνδέσουμε με το Keycloak πατάμε την επιλογή “Clients” στα αριστερά.



Εικ. 27 Επιλογή Clients στο Keycloak

Έπειτα “Create” πάνω δεξιά στην λίστα. Συμπληρώνουμε τα πεδία:

Client ID = το όνομα του client που συνδέουμε το περιβάλλον

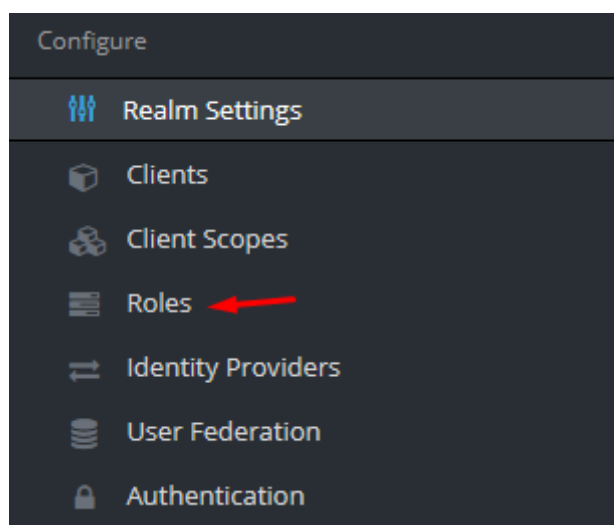
Client Protocol = το πρωτόκολλο για την σύνδεση (προτείνεται openid)

Root URL = η αρχική σελίδα της εφαρμογής που συνδέουμε(η προεπιλεγμένη διεύθυνση για την React εφαρμογή μας είναι <http://localhost:3000/>)

Τέλος πατάμε “Create” και το Keycloak έχει ρυθμιστεί να δέχεται συνδέσεις από την εφαρμογή μας.

4.3.2.4. Δημιουργία Roles

Για την δημιουργία ενός ρόλου για τους χρήστες του Keycloak κατευθυνόμαστε στην επιλογή “Roles”

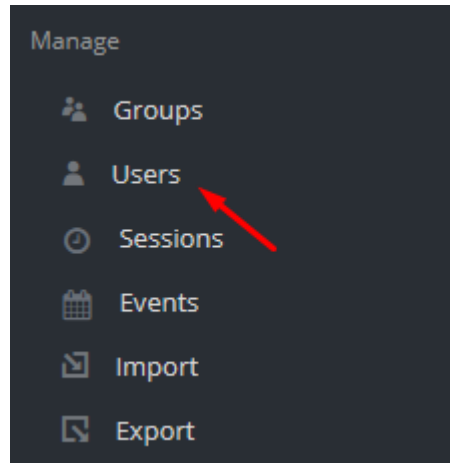


Εικ. 28 Επιλογή Roles στο Keycloak

Στην λίστα μας εμφανίζονται όλοι οι διαθέσιμοι ρόλοι για τους χρήστες του επιλεγμένου περιβάλλοντος. Για να δημιουργήσουμε έναν νέο, πατάμε την επιλογή “Add Role”, συμπληρώνουμε το όνομα για τον ρόλο και μία περιγραφή αυτού, τέλος “Save”. Ο ρόλος έχει δημιουργηθεί. Μπορούμε να προσθέσουμε όσους ρόλους επιθυμούμε για τον περιβάλλον μας. Αυτό θα μας βοηθήσει αργότερα ώστε να αλλάζουμε το περιεχόμενο της εφαρμογής με βάση τον ρόλο του χρήστη.

4.3.2.5. Δημιουργία Users

Μπορούμε να δημιουργήσουμε κατευθείαν χρήστες από το περιβάλλον του Keycloak. Για να το κάνουμε αυτό κατευθυνόμαστε στην επιλογή “Users”



Εικ. 29 Επιλογή Users στο Keycloak

Πατάμε στην επιλογή “Add User”. Συμπληρώνουμε τα πεδία που επιθυμούμε για τον χρήστη. Βασική προϋπόθεση η συμπλήρωση του πεδίου Username. Αν θέλουμε μπορούμε να συμπληρώσουμε ονοματεπώνυμο, email, αν ο χρήστης χρειάζεται να ενημερώσει τον κωδικό του με το που συνδεθεί και πολλά άλλα. Αφού ολοκληρώσουμε την συμπλήρωση πατάμε “Save”. Ο χρήστης μας έχει δημιουργηθεί και εμφανίζονται οι πληροφορίες του. Μπορούμε να δημιουργήσουμε και κωδικό πρόσβασης πατώντας στην καρτέλα “Credentials”. Εισάγουμε τον κωδικό του και την επαλήθευση στα πεδία και πατάμε “Set Password”. Αν θέλουμε ενεργοποιούμε την επιλογή Temporary ώστε ο χρήστης να αλλάξει τον κωδικό όταν εισέλθει πρώτη φορά στο σύστημα, αλλιώς το απενεργοποιούμε.

A form titled 'Set Password'. It has three input fields: 'Password', 'Password Confirmation', and 'Temporary'. The 'Temporary' field is a toggle switch currently set to 'ON'. Below the fields is a 'Set Password' button. Red arrows point to each of the three input fields and the button.

Εικ. 30 Δημιουργία κωδικού πρόσβασης για χρήστη

Ο χρήστης μπορεί να εισέλθει στο σύστημα με τα στοιχεία (username, password) που του δημιουργήσαμε.

4.4. Δημιουργία API Server [10]

Ο API Server είναι υπεύθυνος για να περιέχει όλα τα API που χρειάζεται το περιβάλλον για να μας εμφανίσει τα δεδομένα. Δηλαδή είναι υπεύθυνος για την σύνδεση της βάσης δεδομένων με το βασικό περιβάλλον. Όταν ζητηθεί από την εφαρμογή μία πληροφορία «χτυπάει» το κατάλληλο URL του Server για να την λάβει. Με την σειρά ο Server λαμβάνει τις πληροφορίες από την βάση δεδομένων και τις στέλνει πίσω στην εφαρμογή ώστε να τις εμφανίσει στον χρήστη.

Για τον API Server έχουμε δημιουργήσει ένα νέο React project με το όνομα “Backend”. Οι βιβλιοθήκες που χρειάζεται είναι:

- Express
- CORS
- MySQL
- Nodemailer
- Multer

Για να δημιουργήσουμε τα API χρησιμοποιούμε την Express. Η Express παρέχει έναν εύκολο τρόπο για να δημιουργήσουμε συναρτήσεις POST και GET για να επικοινωνεί με την βάση. Τις περισσότερες φορές χρησιμοποιούμε POST στην περίπτωση που έχουμε παραπάνω μεταβλητές για τα queries προς την βάση και GET για τα queries που δεν χρειάζονται μεταβλητές όπως να προσπελάσουμε όλα τα ραντεβού.

Επιπλέον χρειάζεται την βιβλιοθήκη της MySQL για να γίνει η σύνδεση της Express με την βάση και να εκτελεστούν τα queries. Όπως επίσης και το CORS για να ενεργοποιήσει την αντίστοιχη λειτουργία.

Μία βασική συνάρτηση GET που χρησιμοποιούμε είναι η appointmentsrecent η οποία φέρνει όλα τα πρόσφατα ραντεβού της βάσης. Ο κώδικάς της είναι:

```
app.get("/appointmentsrecent", function (req, res) {
  var sql = "SELECT * FROM appointments WHERE slot >= CURDATE() ORDER BY slot ASC;";
  con.query(sql, function (err, rows) {
    if (err) {
      res.json({ Error: true, Message: "Error Execute Sql", err });
    } else {
      res.json(rows);
    }
  });
});
```

Άλλο ένα παράδειγμα συνάρτησης POST που μας επιτρέπει επίσης την εγγραφή στοιχείων στην βάση είναι η συνάρτηση createappointment που είναι υπεύθυνη για την εγγραφή του ραντεβού του χρήστη στην βάση:

```
app.post("/createappointment", function (req, res) {
  var sql =
    "INSERT INTO appointments (slot,isconfirmed,carrierid,userid,username,comment,file)
VALUES (?, ?, ?, ?, ?, ?)";
  var body =
    [req.body.slot, req.body.isconfirmed, req.body.carrierid, req.body.userid, req.body.username, req
    .body.comment, req.body.file];
  con.query(sql, body, function (err, rows) {
    if (err) {
      res.json({ Error: true, Message: "Error Execute Sql", err });
    } else {
      res.json({ Error: false, Message: "Success" });
    }
  });
});
```

Σε κάθε μέθοδο υπάρχουν έλεγχοι ώστε να εξετάσουμε και να ενημερωθούμε αν υπάρχουν σφάλματα κατά την λειτουργία των συναρτήσεων. Και επιλέγουμε να στέλνουμε τα αποτελέσματα από τα queries σε μορφή json γιατί είναι μία μορφή εύκολα διαχειρίσιμη από το περιβάλλον.

Επιπλέον χρησιμοποιήσαμε το Nodemailer για να μπορούμε να στέλνουμε τις ειδοποιήσεις προς τους χρήστες με email. Το Nodemailer συνδέει έναν email server SMTP με την εφαρμογή και αποστέλλει από αυτόν emails με το κατάλληλο περιεχόμενο που ορίζουμε εμείς.

```
app.post('/sendmail', (req, res, next) => {
  var email = req.body.email;
  var message = req.body.message;
  var content = `email: ${email} \n message: ${message}`;

  var mail = {
    from: 'notification@carriercenter.gr',
    to: email,
    subject: "Ακύρωση ραντεβού",
    message: message,
    text: content
  }

  transporter.sendMail(mail, (err, data) => {
    if (err) {
      res.json({
        status: 'fail'
      })
    }
  })
});
```

```
} else {  
  res.json({  
    status: 'success'  
  })  
}  
})  
})
```

Η Multer είναι μία βιβλιοθήκη η οποία διαχειρίζεται τα δεδομένα από τις φόρμες. Στην δική μας περίπτωση διαχειρίζεται τα αρχεία που ανεβάζουν οι χρήστες στα ραντεβού τους και επιτρέπει την αποστολή στον server και στο περιβάλλον χρήσης.

```
app.post("/uploadfile", function (req, res) {  
  var uploadedfilename;  
  var storage = multer.diskStorage({  
    destination: function (req, file, cb) {  
      cb(null, "public");  
    },  
    filename: function (req, file, cb) {  
      cb(null, Date.now() + "-" + file.originalname);  
      uploadedfilename = Date.now() + "-" + file.originalname;  
    },  
  });  
  
  var upload = multer({ storage: storage }).single("file");  
  
  upload(req, res, function (err) {  
    if (err instanceof multer.MulterError) {  
      return res.status(500).json(err);  
    } else if (err) {  
      return res.status(500).json(err);  
    }  
    res.json({filename: uploadedfilename});  
  });  
});
```

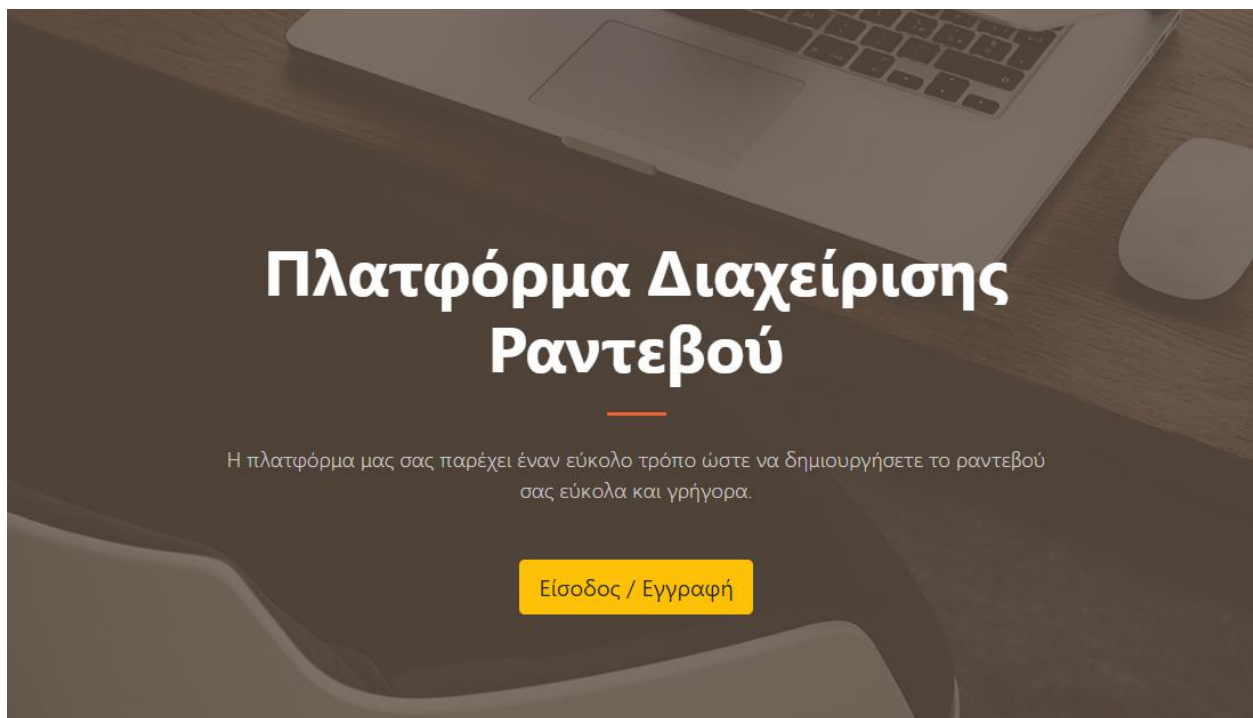
Όλες οι παραπάνω συναρτήσεις είναι υπεύθυνες για την μεταφορά των δεδομένων στο περιβάλλον των χρηστών. Οι κλήσεις αυτών γίνονται από μεθόδους AJAX κάνοντας κλήση στο κατάλληλο URL της κάθε μεθόδου και αποστέλλοντας μαζί τα απαραίτητα δεδομένα για την κάθε μία ώστε να λειτουργήσει σωστά.

4.5. Δημιουργία εφαρμογής

Η εφαρμογή μας αποτελείται από μικρό πλήθος σελίδων αλλά μεγάλο πλήθος λειτουργιών. Οι σελίδες που έχει ο χρήστης περιλαμβάνουν αρκετές λειτουργίες ώστε να μην χρειάζεται να πλοηγείτε σε πολλά μενού και να χάνετε μέσα σε επιλογές. Έχουμε δημιουργήσει το περιβάλλον για την χρήση της εφαρμογής με γνώμονα να μπορεί να χρησιμοποιηθεί εύκολα από πολλούς χρήστες και να εμφανίζονται ακριβώς οι πληροφορίες που ζητάει να δει εκείνη την στιγμή. Παρακάτω θα δούμε τις περιέχουν οι σελίδες του συστήματος και πως υλοποιήθηκαν κάποιες βασικές λειτουργίες αυτών.

4.5.1. Αρχική σελίδα

Η αρχική σελίδα είναι η σελίδα που βλέπει ο χρήστης μόλις μπει στην εφαρμογή μας. Του παρουσιάζονται κάποιες βασικές πληροφορίες για την εφαρμογή και το κουμπί για την είσοδο ή την εγγραφή του στο σύστημα.



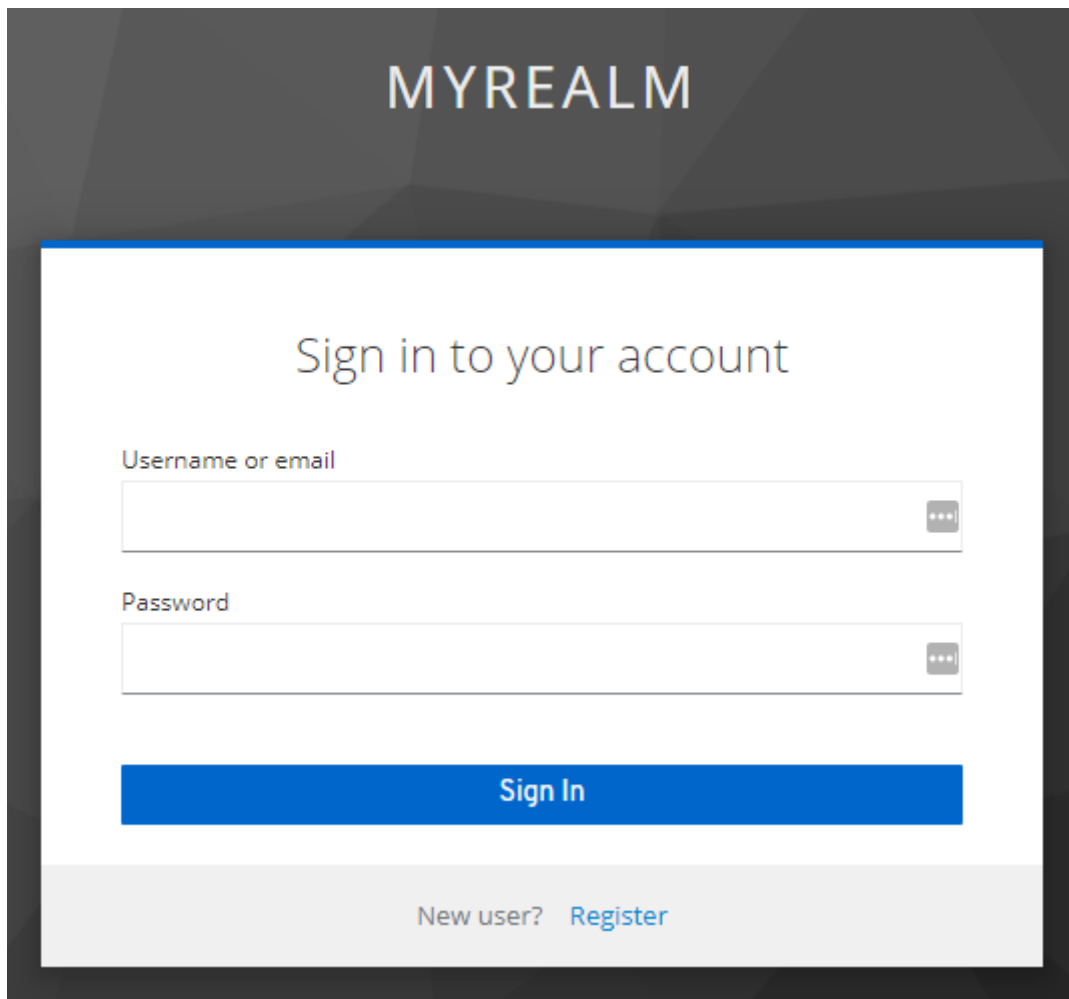
Εικ. 31 Αρχική σελίδα εφαρμογής

Χρησιμοποιώντας το Bootstrap για να διαχειρίζεται το layout και το σχεδιαστικό (CSS) μπορέσαμε με συγκεκριμένες «κλάσεις» στα στοιχεία της σελίδας να φέρουμε το επιθυμητό αποτέλεσμα. Έτσι με λίγες γραμμές κώδικα μπορούμε να ορίσουμε τα χρώματα, τις γραμματοσειρές και την στοίχιση των στοιχείων.

```
<div class="container h-100">
  <div class="row h-100 align-items-center justify-content-center text-center">
    <div class="col-lg-10 align-self-end">
      <h1 class="text-white font-weight-bold">
```

4.5.2. Σελίδα Εισόδου / Εγγραφής

Η σελίδα εισόδου / εγγραφής έχει δημιουργηθεί δυναμικά από το Keycloak κάνοντας κλικ ο χρήστης στο κουμπί της αρχικής σελίδας τον ανακατευθύνει αυτόματα σε αυτή την σελίδα.



Εικ. 32 Σελίδα εισόδου στο σύστημα

Για να δημιουργήσουμε την ανακατεύθυνση αυτή έχουμε δημιουργήσει ένα component το οποίο εξετάζει αν ο χρήστης έχει εισέλθει στο σύστημα αλλιώς του ζητάει να συνδεθεί.

```
constructor(props) {  
  super(props);  
  this.state = { keycloak: null, authenticated: false };  
}  
  
componentDidMount() {  
  const keycloak = Keycloak({  
    url: 'http://localhost:8080/auth',  
    realm: 'myrealm',  
    clientId: 'my-react-client'  
  });  
  keycloak.init({onLoad: 'login-required'}).then(authenticated => {
```

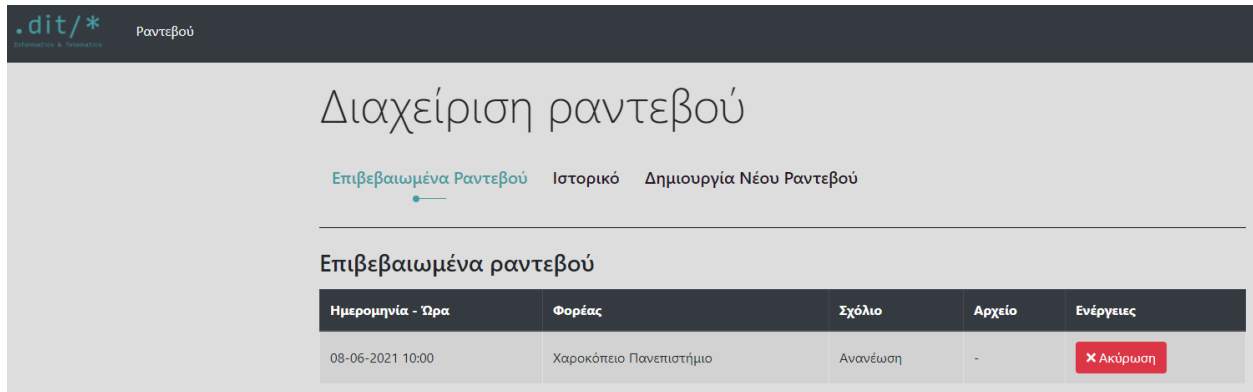
```
    this.setState({ keycloak: keycloak, authenticated: authenticated })
  })
}

render() {
  if (this.state.keycloak) {
    const userrole = this.state.keycloak.tokenParsed.realm_access.roles;
    // const userid = this.state.keycloak.tokenParsed.sub;
    if (this.state.authenticated) {
      if (userrole.includes("user")) {
        return <Redirect to="/user" />;
      } else if (userrole.includes("carrier")){
        return <Redirect to="/carrier" />;
      } else {
        return <Redirect to="/" />;
      }
    } else {
      return <div>Unable to authenticate!</div>;
    }
  }
  return <div>Loading...</div>;
}
```

Αν έχει συνδεθεί τον ανακατευθύνει μόνο του στην αντίστοιχη σελίδα για τον ρόλο που του αντιστοιχεί.

4.5.3. Σελίδα Χρήστη

Η σελίδα του χρήστη απευθύνεται στους χρήστες του συστήματος. Οι χρήστες μπορούν να δουν τα ραντεβού τους, το ιστορικό τους και να δημιουργήσουν ένα νέο ραντεβού.



Εικ. 33 Σελίδα Χρήστη

Στην σελίδα που εμφανίζεται από το μενού μπορούν να επιλέξουν την καρτέλα που θέλουν να τους εμφανιστεί. Η κάθε καρτέλα έχει δημιουργηθεί ως ξεχωριστή λειτουργία και ανανεώνεται αυτόματα με κάθε αλλαγή. Δηλαδή αν ένας χρήστης δεν έχει κανένα ραντεβού και κλείσει ένα νέο αυτόματα εμφανίζεται η καρτέλα για τα ραντεβού. Όπως και αν ακυρώσει ένα ραντεβού. Για να γίνει αυτή η λειτουργία εμφάνισης των ραντεβού πρέπει να χτυπάμε τον API Server με τις πληροφορίες που θέλουμε. Για παράδειγμα για να κλείσει ο χρήστης ραντεβού, επιλέγει από την λίστα τον φορέα που επιθυμεί. Καλούμε το API για τους φορείς και εμφανίζουμε τους συνεργαζόμενους. Έπειτα καλούμε ξανά με την επιλογή το API ώστε να φέρει τις διαθέσιμες ώρες για τον συγκεκριμένο φορέα. Επιλέγει ο χρήστης την ημερομηνία που θέλει συμπληρώνει αν επιθυμεί το σχόλιο και το αρχείο και με το που πατήσει «Αποστολή» καλείται το API για να εισάγει τα δεδομένα του ραντεβού στην βάση και απενεργοποιεί η ώρα για την οποία έκλεισε το ραντεβού.

4.5.3.1. Εμφάνιση Ραντεβού [11]

Οι κλήσεις γίνονται με την χρήση AJAX μεθόδων. Για να εμφανίσουμε τα ραντεβού του χρήστη καλούμε πρώτα την μέθοδο του API `appointmentsbyuserconfirmed` δίνοντας της το `userid` για τον συγκεκριμένο χρήστη.

```
async confirmedappointments(currentuserid) {
  await this.setState({ confirmedappointments: [] });
  await axios
    .post("http://localhost:3001/appointmentsbyuserconfirmed", {
      userid: currentuserid,
    })
    .then(
      function (response) {
        if (response.data.length !== 0) {
```

```

        response.data.forEach((element) => {
            element.start = moment(element.slot).format("DD-MM-YYYY HH:mm");
            var joined = this.state.confirmedappointments.concat(element);
            this.setState({ confirmedappointments: joined });
            this.setState({ showconfirmed: true });
        });
    }
    }.bind(this)
)
.catch(function (error) {
    console.log(error);
});
}

```

Αυτή η μέθοδος μας επιστρέφει τα ραντεβού του συγκεκριμένου χρήστη εμείς τα περνάμε σε ένα react state και έπειτα τα διαβάζουμε και τα τυπώνουμε ώστε να εμφανιστούν στον πίνακα μας. Υπάρχει επίσης έλεγχος αν δεν υπάρχει καμία εγγραφή στα αποτελέσματα να μην εμφανίζεται καθόλου.

```

{this.state.confirmedappointments.map(
    (listitem) => (
        <tr
            id={listitem.id}
            key={"pending" + listitem.id}
        >
            <td className="align-middle">
                {listitem.start}
            </td>
            <td className="align-middle">
                {listitem.name}
            </td>
            <td className="align-middle">
                {listitem.comment}
                ? listitem.comment
                : "-"
            </td>
            <td className="align-middle">
                {listitem.file ? (
                    <a
                        href={
                            "http://localhost:3001/" +
                            listitem.file
                        }
                        target="_blank"
                        rel="noreferrer"
                        className="btn btn-info"
                        download
                    >

```



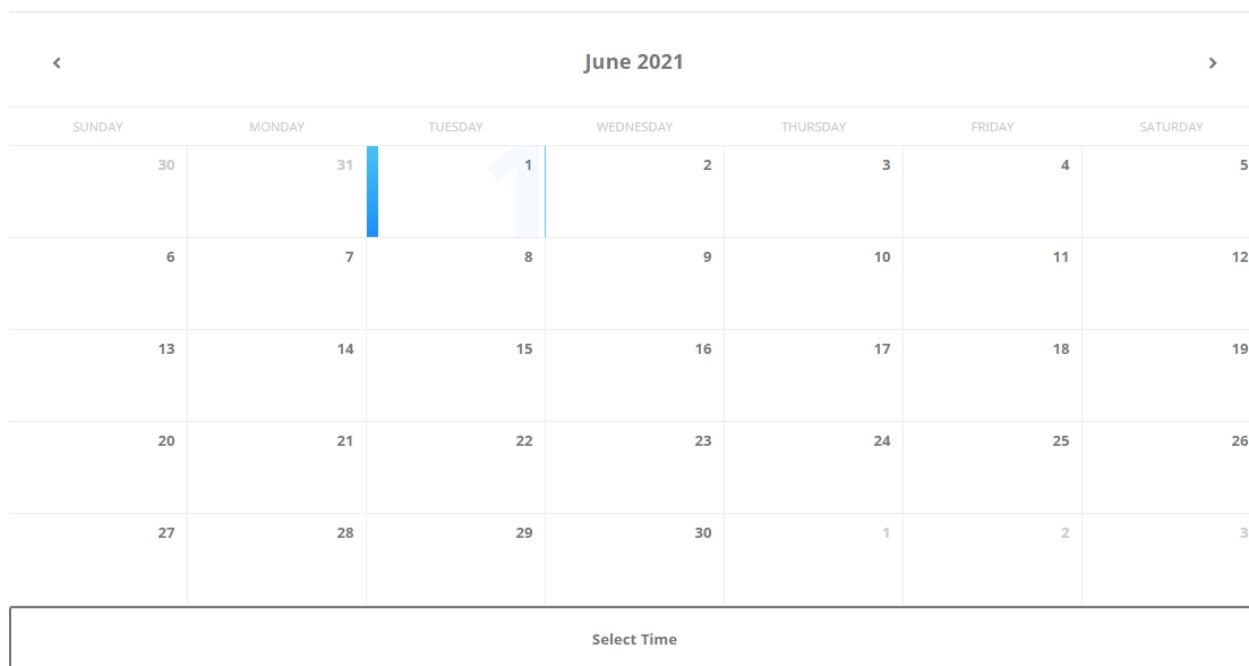
```

        <FontAwesomeIcon icon={faFile} />
      </a>
    ) : (
      "_"
    )}
    { /* {listitem.file} */ }
  </td>
  <td className="align-middle">
    <Button
      variant="danger"
      onClick={() =>
        this.showdeletemodal(listitem.id)
      }
    >
      <FontAwesomeIcon
        icon={faTimes}
        className="mr-1"
      />
      Ακύρωση
    </Button>
  </td>
</tr>
)
)}

```

Με την ίδια λογική εμφανίζουμε το Ιστορικό των ραντεβού του καλώντας τα κατάλληλα API για να εμφανίζει όλα τα ραντεβού του χρήστη.

4.5.3.2. Ημερολόγιο Ραντεβού [6]



Εικ. 34 Ημερολόγιο για νέο ραντεβού

Για να εμφανίσουμε το ημερολόγιο με τα διαθέσιμα ραντεβού των χρηστών χρησιμοποιούμε μία βιβλιοθήκη της React με όνομα Time Calendar. Αυτή η βιβλιοθήκη παρέχει το δικό της component στο οποίο με παραμέτρους ορίζουμε τον χρόνο για το κάθε ραντεβού, τις ώρες που υπάρχουν διαθέσιμα ραντεβού, τις ώρες που είναι κλεισμένα ραντεβού και όταν επιλέξει μία συγκεκριμένη ώρα τι θα κάνει.

```
openHours: [  
  [9, 20],  
  [9, 14],  
],  
  
<TimeCalendar  
  disableHistory  
  clickable  
  timeSlot={30}  
  bookings={bookings}  
  openHours={openHours}  
  onTimeClick={this.showcreateappmodal}  
>
```

4.5.3.3. Δημιουργία Ραντεβού

Για να κλείσει νέο ραντεβού, όταν πατήσει το κουμπί «Αποστολή» καλούμε την συνάρτηση για την δημιουργία του. Λαμβάνει την ημερομηνία και ώρα που επιλέχθηκε, αποθηκεύει προσωρινά αν υπάρχει αρχείο και έπειτα περνάει όλα τα στοιχεία σε μία μέθοδο POST καλώντας τα API για να προσθέσει το ραντεβού στην βάση. Έπειτα καλεί όλες τις υπόλοιπες μεθόδους ώστε να ανανεώσει τα ραντεβού και να κλείσει το παράθυρο.

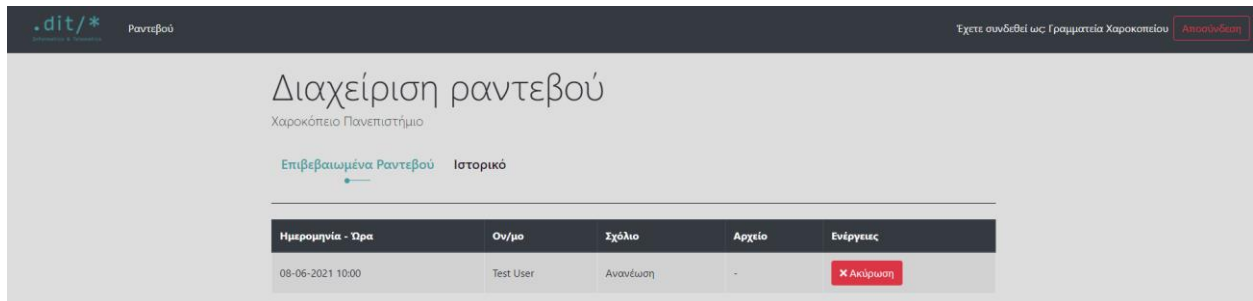
```
async createAppointment() {
  // console.log(time);
  var slottime = moment(this.state.selectedtime).format(
    "YYYY-MM-DD HH:mm:ss"
  );

  const data = new FormData();
  data.append("file", this.state.selectedFile);
  if (this.state.selectedFile) {
    await this.sendfile(data);
  }
  var comment = "";
  if (this.state.val) {
    // console.log("a");
    comment = this.state.val;
  }
  await axios
    .post("http://localhost:3001/createappointment", {
      slot: slottime,
      isconfirmed: "1",
      carrierid: this.state.selectedcarrier,
      comment: comment,
      userid: this.state.currentuserid,
      username: this.state.currentusername,
      file: this.state.filename,
    })
    .then((response) => {
    })
    .catch(function (error) {
      console.log(error);
    });

  this.getappointments();
  this.pendingappointments(this.state.currentuserid);
  this.confirmedappointments(this.state.currentuserid);
  this.handleCreateClose();
}
```

4.5.4. Σελίδα Φορέα

Η σελίδα φορέα είναι υπεύθυνη για την εμφάνιση στους υπαλλήλους του φορέα των ραντεβού που αντιστοιχούν σε αυτόν. Μπορούν να δουν τα ραντεβού του φορέα και να ακυρώσουν ραντεβού αν χρειαστεί.



Εικ. 35 Σελίδα φορέα

Στην σελίδα του φορέα υπάρχει η ίδια λογική με την σελίδα του χρήστη. Οι καρτέλες στο μενού αλλάζουν την προβολή για τους πίνακες με βάση την επιλογή. Εμφανίζεται δυναμικά το όνομα του τρέχων φορέα κάτω από τον τίτλο και οι λειτουργίες είναι σχεδόν οι ίδιες με παραλλαγές στις συναρτήσεις που γίνονται κλήση στο API.

4.5.4.1. Διαχωρισμός Φορέα

Υπάρχουν πολλοί υπάλληλοι σε κάθε φορέα και ξεχωρίζουν από το χαρακτηριστικό id του φορέα, που υπάρχει στο προφίλ τους από το Keycloak. Λαμβάνουμε αυτό το id κατά την σύνδεση του χρήστη στο σύστημα και έπειτα καλούμε τις συναρτήσεις ώστε να φέρουν τα αντίστοιχα δεδομένα του.

```
keycloak.loadUserProfile().then((profile) => {
  if (keycloak.tokenParsed.realm_access.roles == "carrier") {
    currentcarrierid = profile.attributes.carrierid[0];
    axios
      .post("http://localhost:3001/carriers", {
        carrierid: currentcarrierid,
      }).then(
        function (response) {
          if (response.data.length !== 0) {
            this.setState({ carriername: response.data[0].name });
          }
        }.bind(this)
      ).catch(function (error) {
        console.log(error);
      });
  } else {
    currentcarrierid = 0;
  }
});
```

```

    }
    this.setState({
      currentcarrierid: currentcarrierid,
      carriername: carriername,
    });
    this.pendingappointments(currentcarrierid);
    this.confirmedappointments(currentcarrierid);
    this.appointmentshistory(currentcarrierid);
  });

```

4.5.4.2. Ακύρωση Ραντεβού

Όταν ακυρώσει ο φορέας ένα ραντεβού μπορεί να συμπληρώσει τον λόγο για τον οποίο το ακυρώνει. Πατώντας επιβεβαίωση ακύρωσης καλείται μία συνάρτηση η οποία είναι υπεύθυνη για την διαγραφή του ραντεβού από την βάση και την αποστολή email [7] στον χρήστη με τον λόγο της ακύρωσης.

```

deleteTask = async (appid) => {
  var userid = "";
  var useremail = "";
  await axios
    .post("http://localhost:3001/appointments", {
      appid: appid,
    }).then(async function (response) {
      // console.log(response.data[0]);
      // this.setState({ appoi: response.data[0].userid });
      userid = response.data[0].userid;
      useremail = response.data[0].email;
    });
  await axios
    .post("http://localhost:3001/sendmail", {
      email: useremail,
      message: this.state.val,
    }).then(function (response) {
      console.log(response);
    });
  await axios.post('http://localhost:3001/deleteappointment', {
    appid: appid
  }).then(function (response) {
    console.log(response);
  })
  this.handleCreateClose();
  this.pendingappointments(this.state.currentcarrierid);
  this.confirmedappointments(this.state.currentcarrierid);
};

```

4.5.5. Αποσύνδεση

Έχετε συνδεθεί ως: Γραμματεία Χαροκοπείου

Αποσύνδεση

Εικ. 36 Όνομα χρήστη και κουμπί αποσύνδεσης

Για την αποσύνδεση των χρηστών γίνεται κλήση σε μία μέθοδο που καταργεί την συγκεκριμένη συνεδρία του χρήστη και το επιστρέφει στην αρχική σελίδα. Επιπλέον καλεί την μέθοδο logout του Keycloak ώστε να διακόψει την σύνδεση του χρήστη με αυτό.

```
logout() {  
  this.props.history.push('/');  
  this.props.keycloak.logout();  
}
```

5. Κεφάλαιο 5^ο Σενάρια Χρήσης

Σε αυτό το κεφάλαιο θα δούμε πιο αναλυτικά το πως ο κάθε χρήστης μπορεί να χρησιμοποιήσει το σύστημα και να εκτελέσει τις λειτουργίες αυτού με παραδείγματα. Όπως επίσης πως γίνονται σωστά οι λειτουργίες με τις απαραίτητες οδηγίες.

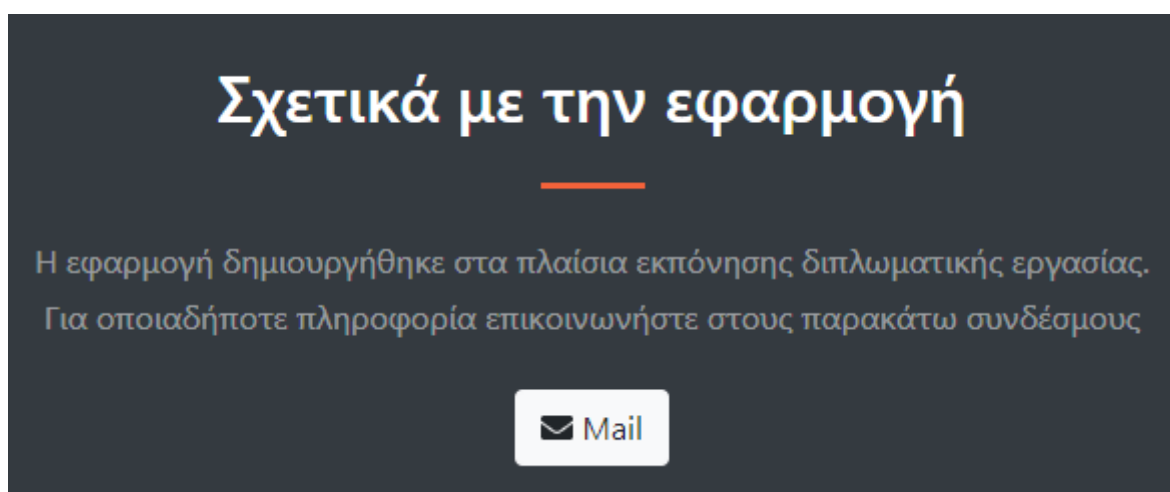
5.1. Αρχική Σελίδα

Ο χρήστης καθώς μπαίνει στην αρχική σελίδα μπορεί να δει την περιγραφή της εφαρμογής και το κουμπί εισόδου/εγγραφής.



Εικ. 37 Βασική οθόνη εφαρμογής

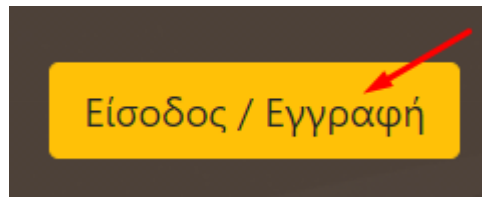
Κάνοντας scroll κάτω εμφανίζονται παραπάνω πληροφορίες και το κουμπί επικοινωνίας.



Εικ. 38 Πληροφορίες εφαρμογής

5.2. Είσοδος χρήστη/φορέα στο σύστημα

Για να συνδεθεί ο χρήστης ή ο υπάλληλος του φορέα στο σύστημα πρέπει να πατήσει στο κουμπί «Είσοδο / Εγγραφή» στην αρχική σελίδα.



Εικ. 39 Κουμπί εισόδου/εγγραφής

Έπειτα εισάγει τα στοιχεία εισόδου του, όνομα χρήστη και κωδικός πρόσβασης, στην φόρμα που εμφανίζεται και πατάει “Sign In”.

Sign in to your account

Username or email

1

Password

2

3

Sign In

Εικ. 40 Συμπλήρωση φόρμας εισόδου

Αν συμπληρωθούν σωστά τα στοιχεία θα του εμφανιστεί η αντίστοιχη σελίδα βάση τον ρόλο του χρήστη του. Στην περίπτωση που τα στοιχεία του είναι λάθος ή ο χρήστης του δεν υπάρχει θα του εμφανιστεί μήνυμα λάθους.

Sign in to your account

Username or email

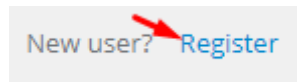
Invalid username or password.

Password

Εικ. 41 Μήνυμα λάθους κατά την είσοδο

5.3. Εγγραφή Χρήστη

Οι χρήστες που δεν έχουν λογαριασμό μπορούν εύκολα να δημιουργήσουν έναν κάνοντας κλικ στην επιλογή “Register” στην σελίδα εισόδου.



Εικ. 42 Κουμπί εγγραφής

Έτσι τους εμφανίζεται η φόρμα εγγραφής που πρέπει να συμπληρώσουν τα στοιχεία τους. Αφού συμπληρωθούν τα πεδία και πατήσουν “Register” γίνεται αυτόματη ανακατεύθυνση στην σελίδα του χρήστη που μπορούν να δουν τα ραντεβού και να κλείσουν νέο.

Register

First name

Last name

Email

Username

Password

Confirm password

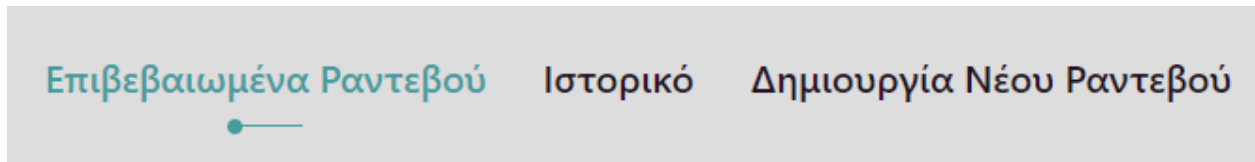
[« Back to Login](#)

Register

Εικ. 43 Φόρμα εγγραφής

5.4. Σελίδα Χρήστη

Στην σελίδα του χρήστη εμφανίζονται όλα τα ραντεβού του. Επιβεβαιωμένα, αν υπάρχουν μη επιβεβαιωμένα και το ιστορικό των παλαιότερων ραντεβού του. Επιπλέον υπάρχει και η επιλογή να κλείσει νέο ραντεβού. Πατώντας από τις επιλογές στο μενού αλλάζει η προβολή από κάτω.

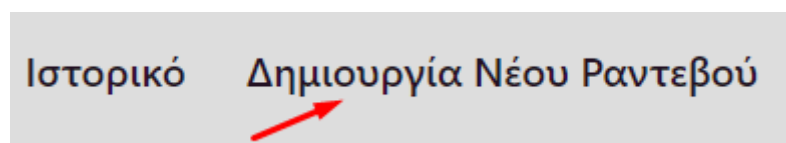


Εικ. 44 Μενού χρήστη

Έτσι ο χρήστης μπορεί να δει όλες τις απαραίτητες πληροφορίες σε μία σελίδα χωρίς να χρειάζεται την αλλαγή τους.

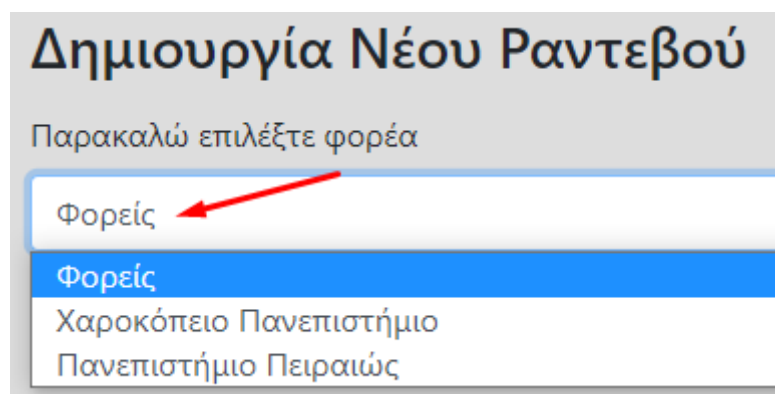
5.5. Δημιουργία Νέου Ραντεβού

Για την δημιουργία νέου ραντεβού πρέπει να έχουμε εισέλθει στο σύστημα ως χρήστης και να πατήσουμε στην επιλογή «Δημιουργία Νέου Ραντεβού»



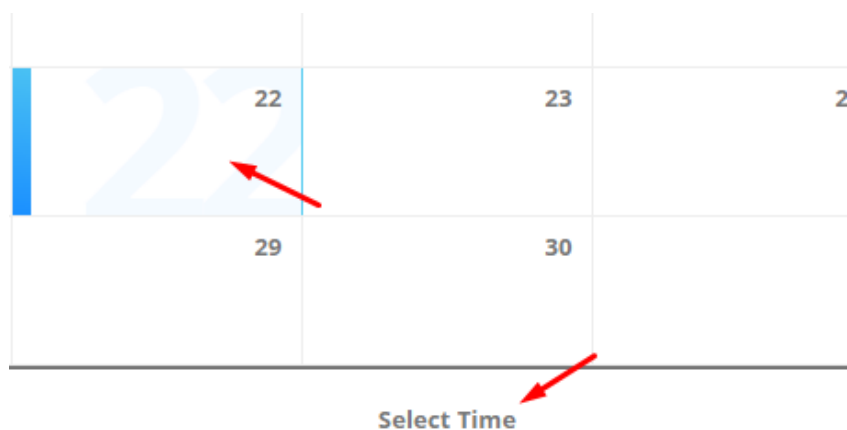
Εικ. 45 Επιλογή δημιουργία νέου ραντεβού

Η προβολή αλλάζει και ο χρήστης πρέπει να επιλέξει τον φορέα για τον οποίο θέλει να κλείσει το ραντεβού του.



Εικ. 46 Επιλογή φορέα

Στην συνέχεια εμφανίζεται το ημερολόγιο με τις διαθέσιμες ημέρες. Ο χρήστης επιλέγει την ημερομηνία που επιθυμεί και πατάει το κουμπί «Select Time»



Εικ. 47 Επιλογή ημερομηνίας ραντεβού

Εμφανίζονται όλες οι ώρες της συγκεκριμένης ημέρας. Οι ώρες που είναι με γκρι αχνό χρώμα είναι αυτές που είναι κλεισμένες, μη διαθέσιμες και άμα πατήσει ο χρήστης πάνω δεν γίνεται τίποτα, ενώ αυτές που είναι με πιο έντονο γκρι είναι οι διαθέσιμες και όταν πατηθούν εμφανίζεται το παράθυρο για το νέο ραντεβού.

11-00	11-30
13-00	13-30

Εικ. 48 Διαθέσιμες και μη ώρες ραντεβού

Αφού ο χρήστης επιλέξει την διαθέσιμη ώρα που επιθυμεί εμφανίζεται το παράθυρο για να κλείσει το ραντεβού του. Υπάρχουν δύο πεδία που αν θέλει μπορεί να συμπληρώσει, το πεδίο για το σχόλιο του ραντεβού που συμπληρώνει την αιτιολογία για το ραντεβού του και το πεδίο για το αρχείο. Στο πεδίο του αρχείου μπορεί να ανεβάσει ένα δικαιολογητικό όπως την ταυτότητα του, κάποιο έγγραφο ή να το αφήσει κενό αν δεν χρειάζεται.

Νέο ραντεβού

×

Σχόλιο

Επιλογή αρχείου

Δεν επιλέχθηκε κανένα αρχείο.

Πίσω

Αποστολή

Εικ. 49 Φόρμα νέου ραντεβού

Τέλος πατάει «Αποστολή» για να στείλει το ραντεβού στον φορέα. Αφού σταλθεί το ραντεβού του εμφανίζεται στα επιβεβαιωμένα ραντεβού του και επίσης κλείνει αυτόματα από το σύστημα η ώρα που διάλεξε ώστε να μην κλείσει άλλος χρήστης.



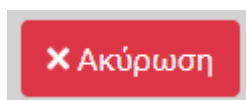
Εικ. 50 Κουμπί αποστολής

Επιβεβαιωμένα ραντεβού				
Ημερομηνία - Ώρα	Φορέας	Σχόλιο	Αρχείο	Ενέργειες
08-06-2021 10:00	Χαροκόπειο Πανεπιστήμιο	Ανανέωση	-	✕ Ακύρωση

Εικ. 51 Λίστα επιβεβαιωμένων ραντεβού

5.6. Ακύρωση Ραντεβού

Για να ακυρώσει ένας χρήστης το ραντεβού του από τις λίστες των επιβεβαιωμένων ραντεβού και μη επιβεβαιωμένων αρκεί να πατήσει το κουμπί «Ακύρωση»



Εικ. 52 Κουμπί ακύρωσης

Έτσι του εμφανίζεται η φόρμα για την ακύρωση του ραντεβού. Συμπληρώνει τον λόγο ακύρωσης και πατάει το κουμπί της επιβεβαίωσης. Το ραντεβού του έχει πλέον ακυρωθεί. Ο φορέας ενημερώνεται με email για την ακύρωση μαζί με τον λόγο της ακύρωσης και ελευθερώνεται η ώρα από το ημερολόγιο για να το κλείσει κάποιος άλλος χρήστης.

Ακύρωση ραντεβού

✕

1

2

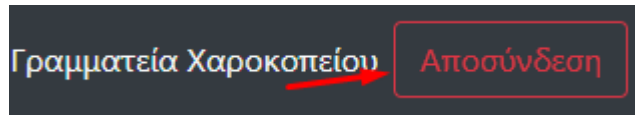
Πίσω

Επιβεβαίωση ακύρωσης

Εικ. 53 Φόρμα ακύρωσης ραντεβού

5.7. Αποσύνδεση Χρήστη

Ο χρήστης μπορεί να αποσυνδεθεί οποιαδήποτε στιγμή θελήσει από το σύστημα πατώντας το κουμπί αποσύνδεσης στο πάνω δεξιό μέρος της οθόνης.



Εικ. 54 Κουμπί αποσύνδεσης

5.8. Σελίδα Φορέα

Στην σελίδα του υπαλλήλου φορέα εμφανίζονται όλα τα ραντεβού σχετικά με τον φορέα για τον οποίο εργάζεται. Επιβεβαιωμένα, αν υπάρχουν μη επιβεβαιωμένα και το ιστορικό των παλαιότερων ραντεβού του.

Διαχείριση ραντεβού

Χαροκόπειο Πανεπιστήμιο

Επιβεβαιωμένα Ραντεβού Ιστορικό

Ημερομηνία - Ώρα	Ον/μο	Σχόλιο	Αρχείο	Ενέργειες
08-06-2021 10:00	Test User	Ανανέωση	-	✕ Ακύρωση

Εικ. 55 Σελίδα φορέα

Οι διαδικασίες για την προβολή είναι οι ίδιες με το χρήστη του συστήματος. Δηλαδή διαλέγει από το μενού την επιλογή για τα ραντεβού που τον ενδιαφέρει και η προβολή αλλάζει για να του δείξει τα επιθυμητά. Επιπλέον η ίδια διαδικασία ισχύει για την ακύρωση των ραντεβού, ο φορέας συμπληρώνει τον λόγο ακύρωσης και ο χρήστης ενημερώνεται αυτόματα με email, καθαρίζοντας την ώρα από το ημερολόγιο.

5.9. Επιβεβαίωση Ραντεβού

Αν υπάρξει κάποιο σφάλμα στο κλείσιμο ραντεβού από τους χρήστες και κλείσουν δύο χρήστες την ίδια ώρα το ραντεβού που έκλεισε δευτερόλεπτα μετά παίρνει την σήμανση μη επιβεβαιωμένο. Στην περίπτωση αυτή ο φορέας πρέπει να εξετάσει αν μπορεί να εξυπηρετήσει τα ραντεβού αυτά και να επιβεβαιώσει αυτό που δεν είναι επιβεβαιωμένο. Αυτό γίνεται πηγαίνοντας στην επιλογή «Μη επιβεβαιωμένα ραντεβού» αν υπάρχουν και στο ραντεβού που επιθυμεί να πατήσει το κουμπί επιβεβαίωση.

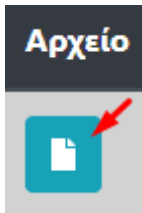
Μη επιβεβαιωμένα ραντεβού		Ιστορικό		
Ημερομηνία - Ώρα	Ον/μο	Σχόλιο	Αρχείο	Ενέργειες
08-06-2021 10:00	Test User	Ανανέωση	-	✕ Ακύρωση ✓ Επιβεβαίωση

Εικ. 56 Επιβεβαίωση ραντεβού

Αν δεν μπορεί να εξυπηρετήσει τα ραντεβού αυτά μαζί μπορεί να το ακυρώσει.

5.10. Προβολή Αρχείου Ραντεβού

Τα ραντεβού έχουν την επιλογή να περιέχουν αρχεία από τον χρήστη. Τα αρχεία αυτά βρίσκονται στις λίστες στην στήλη «Αρχείο». Για να γίνει προβολή ή λήψη του αρχείου για προβολή, ο χρήστης πρέπει να πατήσει στο κουμπί του αρχείου.



Εικ. 57 Κουμπί προβολής αρχείου

Αν δεν υπάρχει από τον χρήστη εμφανίζεται μία παύλα στην θέση του κουμπιού.

6. Κεφάλαιο 6^ο Μελλοντική Εξέλιξη και Συμπεράσματα

6.1. Μελλοντική Εξέλιξη

Βασιζόμενοι στο σύστημα που δημιουργήσαμε με τις ήδη εγκατεστημένες τεχνολογίες, μπορούν να αλλάξουν πολλά. Έχοντας στο νου μας να δημιουργήσουμε κάτι το οποίο θα είναι εύκολα επεκτάσιμο στο μέλλον, φτιάξαμε με τέτοιο τρόπο το σύστημα ώστε να μπορεί ο επόμενος προγραμματιστής του να το αλλάξει.

Για παράδειγμα, αν θέλει να αλλάξει το περιβάλλον χρήσης του θα μπορεί να το κάνει πολύ εύκολα αλλάζοντας μερικά κομμάτια του κώδικα. Αν θέλει να προσθέσει την δυνατότητα σύνδεσης με κοινωνικά μέσα, δεν έχει παρά να δημιουργήσει τις κατάλληλες παραμέτρους ώστε να το ενσωματώσει. Θα μπορούσε το ίδιο το σύστημα να αντιγραφεί και να λειτουργήσει εκτός από φορείς και υπηρεσίες και στο ιδιωτικό τομέα σε κάποια επιχείρηση με ραντεβού.

Θα μπορούσε ακόμη να αλλάξει εντελώς την δομή του συστήματος ώστε να επεκταθεί με πολλές παραπάνω λειτουργίες για την εξυπηρέτηση και χρηστών άλλων χωρών. Παραδείγματα για το πως μπορεί να αξιοποιηθεί το σύστημα που δημιουργήσαμε είναι πολλά. Μπορεί να χρησιμοποιηθεί η ίδια λογική όπως το σκεφτήκαμε εμείς ή μία τελείως διαφορετική.

Ο επόμενος προγραμματιστής θα έχει την δυνατότητα να αναλύσει τις απαιτήσεις που θα προκύψουν στο μέλλον και να το υλοποιήσει με την δικιά του σύμφωνα με τις δυνατότητες του συστήματος.

6.2. Συμπεράσματα

Έχοντας λοιπόν υλοποιήσει την εφαρμογή και τα συστήματα που επικοινωνούν μεταξύ, αξιοποιώντας τεχνολογίες όπως React, Keycloak, Docker κλπ. Καταλήξαμε στα εξής συμπεράσματα:

- Το σύστημα αυτό κατάφερε με επιτυχία να καλύψει τις ανάγκες της επιλογής φορέα. Ο χρήστης μπορεί να διαλέξει τον φορέα που επιθυμεί πριν κλείσει το ραντεβού του.
- Οι διαθέσιμες ώρες ανανεώνονται δυναμικά με τα νέα ραντεβού και τις ακυρώσεις. Το ημερολόγιο επιλογής εμφανίζει εύκολα και ευδιάκριτα τις διαθέσιμες ώρες.
- Υπάρχει δυνατότητα προσθήκης σχολίου και αρχείου στην δημιουργία ραντεβού. Έτσι και ο χρήστης και ο φορέας γνωρίζουν περί τίνος πρόκειται και λαμβάνουν κατευθείαν από το σύστημα τα απαραίτητα δικαιολογητικά.
- Οι χρήστες δεν δεσμεύονται με την δημιουργία του ραντεβού. Μπορούν είτε οι ίδιου να το ακυρώσουν οποιαδήποτε στιγμή επιθυμούν λόγω της άμεσης ενημέρωσης προς τους φορείς. Το ίδιο ισχύει και για τους φορείς, μπορούν να ακυρώσουν τα ραντεβού λόγω της ενημέρωσης των χρηστών.
- Τα δεδομένα των ραντεβού δεν διαγράφονται χωρίς ειδοποίηση. Ο μόνος τρόπος για να διαγραφεί μία πληροφορία για τα ραντεβού είναι από την ακύρωση που υπάρχει ενημέρωση την στιγμή που γίνεται. Όλα τα υπόλοιπα στοιχεία αποθηκεύονται και μένουν στην βάση. Κάθε χρήστης μπορεί να λάβει αυτές τις πληροφορίες από το

ιστορικό του. Στο ιστορικό όμως δεν εμφανίζονται τα ραντεβού τα οποία δεν έχουν περάσει για να μην αγνοηθούν άθελα.

- Με τις τεχνολογίες που χρησιμοποιήσαμε δημιουργήσαμε ένα περιβάλλον ελαφρύ και εύκολο στην χρήση. Μπορεί να χρησιμοποιηθεί από υπολογιστές, tablet ακόμα και κινητά ώστε ο χρήστης να έχει πρόσβαση χωρίς περιορισμούς.
- Το περιβάλλον χρήσης είναι απλό και εύκολα κατανοητό. Υπάρχουν ταμπέλες και κείμενα επεξήγησης για όλες τις λειτουργίες του που κατευθύνουν ακριβώς τον χρήστη εκεί που θέλει.
- Τα στοιχεία του χρήστη αποθηκεύονται με ασφάλεια στο σύστημα μας. Χρησιμοποιούμε κωδικοποίηση ως προς αυτά και δεν μπορούν να ληφθούν από άλλους χρήστες. Αν κάποιος χρήστης δεν έχει τα κατάλληλα διαπιστευτήρια δεν μπορεί να μεταβάλει το σύστημα.
- Ο τρόπος με τον οποίο λειτουργήσαμε καθιστά το σύστημα και το περιβάλλον του πλήρως επεκτάσιμα. Δηλαδή να μπορούν να γίνουν εύκολα αλλαγές, βελτιώσεις και προσθήκες στο σύστημα χωρίς μεγάλο κόπο.

6.3. Συνεισφορά

Παρόμοια συστήματα διαχείρισης ραντεβού αρχίζουν να εμφανίζονται όλο και περισσότερο στις μέρες μας. Τέτοια συστήματα χρησιμοποιούνται και στο δημόσιο και στον ιδιωτικό τομέα λόγω της κατάστασης της εποχής. Άλλα συστήματα πιο σύνθετα με μεγάλο πλήθος λειτουργιών και άλλα πιο απλά με την χρήση εύκολων τεχνολογιών. Μέσα από την δημιουργία αυτής της πτυχιακής εργασίας αποκομίσαμε τις απαραίτητες πληροφορίες και γνώσεις για την υλοποίηση αυτών των συστημάτων.

Για παράδειγμα τι υλικά χρειάζονται για να το φτιάξουμε, πως μπορούμε να το υλοποιήσουμε με βάση τις ανάγκες και τις απαιτήσεις των χρηστών που θα το χρησιμοποιούν καθημερινά, με ποιον τρόπο γίνεται η υλοποίηση αυτού και πως μπορεί να χρησιμοποιηθεί από τον μέσο χρήστη. Σε αυτά τα ερωτήματα κληθήκαμε να απαντήσουμε και να έχουμε το δικό μας παρόμοιο σύστημα, διευρύνοντας έτσι τις ήδη υπάρχουσες γνώσεις μας.

Ανακαλύψαμε νέες τεχνολογίες και δομές που μπορούμε να χρησιμοποιήσουμε και σε άλλα Project. Χρησιμοποιώντας τις τεχνολογίες της React και του Bootstrap είδαμε το πως μπορεί κάποιος να τις αξιοποιήσει για να φτιάξει εφαρμογές δικτύου αρκετά εύκολα με βασικές γνώσεις. Με την βάση δεδομένων MySQL μπορούμε να κάνουμε εύκολη διαχείριση των δεδομένων που έχουμε. Με το Keycloak ανακαλύψαμε ένα εύκολο και γρήγορο τρόπο να διαχειριζόμαστε τους χρήστες των εφαρμογών μας με πολλές χρήσεις για αρκετά συστήματα.

Έχοντας όλες αυτές τις γενικές γνώσεις προσπαθήσαμε να δείξουμε το πως κάποιος ακολουθώντας τις δικές μας οδηγίες μπορεί να φτάσει στο σημείο να υλοποιήσει το δικό του σύστημα για ραντεβού. Μπορεί να χρησιμοποιήσει τις δικές μας έτοιμες σελίδες και με τις γνώσεις του να το μετατρέψει σύμφωνα με τις ανάγκες του. Επεκτείνουμε έτσι την λειτουργικότητα του σε διάφορους τομείς.

Θέλαμε να δημιουργήσουμε ένα είδος εγχειριδίου για το πως μπορεί να στηθεί ένα σύστημα ραντεβού αρχίζοντας από το μηδέν. Παρουσιάζουμε στον ενδιαφερόμενο το τρόπο με τον οποίο στήσαμε το λειτουργικό μας, πως ρυθμίσαμε τις τεχνολογίες του και πως κληθήκαμε με τις ανάγκες των δικών μας χρηστών να προγραμματίσουμε το περιβάλλον του ώστε να τις ικανοποιεί.

Εν κατακλείδι θεωρούμε πως θέσαμε τις βάσεις για ένα σύστημα ραντεβού το οποίο θα μπορεί να χρησιμοποιήσει ο μέσος χρήστης πληροφορικής ώστε να κάνει την καθημερινότητα του πιο εύκολη.

7. Βιβλιογραφία

- [1] React, «React Documentation & Tutorial,» 2014. [Ηλεκτρονικό]. <https://reactjs.org>
- [2] Keycloak, «Keycloak Documentation & Tutorial,» 2014. [Ηλεκτρονικό]. <https://www.keycloak.org/>
- [3] Dasniko, «Keycloak Reactjs Demo,» 2018. [Ηλεκτρονικό]. <https://github.com/dasniko/keycloak-reactjs-demo>
- [4] Bootstrap, «Bootstrap,» 2011. [Ηλεκτρονικό]. <https://getbootstrap.com/>
- [5] R. Bootstrap, «React Bootstrap,» 2018. [Ηλεκτρονικό]. <https://react-bootstrap.github.io/>
- [6] J. Sidford, «Time Calendar,» 2019. [Ηλεκτρονικό]. <https://www.npmjs.com/package/react-timecalendar>
- [7] Nodemailer, «Nodemailer,» 2014. [Ηλεκτρονικό]. <https://nodemailer.com/about/>
- [8] M. Hamedani, «React file upload: proper and easy way, with NodeJS!,» 2019. [Ηλεκτρονικό]. <https://programmingwithmosh.com/javascript/react-file-upload-proper-server-side-nodejs-easy/>
- [9] ExpressJS, «Multer,» 2014. [Ηλεκτρονικό]. <https://www.npmjs.com/package/multer>
- [10] ExpressJS, «Express,» 2010. [Ηλεκτρονικό]. <https://expressjs.com/>
- [11] Axios, «Axios,» 2018. [Ηλεκτρονικό]. <https://www.npmjs.com/package/axios>
- [12] Npm, «Npm,» 2010. [Ηλεκτρονικό]. <https://www.npmjs.com/>
- [13] V. Paradigm, «Visual Paradigm,» 2002. [Ηλεκτρονικό]. <https://www.visual-paradigm.com/>
- [14] Docker, «Docker,» 2013. [Ηλεκτρονικό]. <https://www.docker.com/>
- [15] Docker, «Docker Installation for Windows,» 2020. [Ηλεκτρονικό]. <https://docs.docker.com/docker-for-windows/install/>
- [16] Kubernetes, «Kubernetes,» 2014. [Ηλεκτρονικό]. <https://kubernetes.io/>