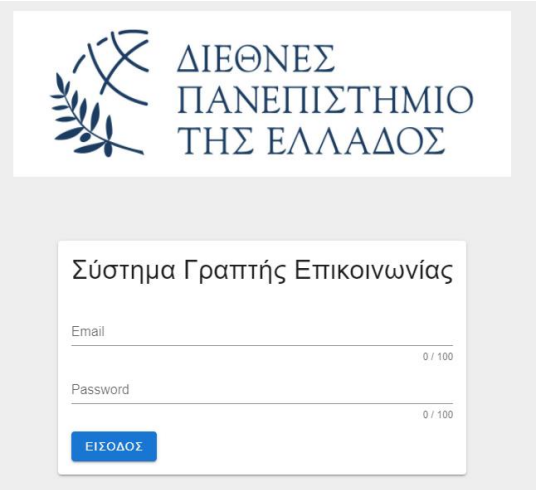
ΔΙΕΘΝΕΣ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΤΗΣ ΕΛΛΑΔΟΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

«Δημιουργία εφαρμογής για σύγχρονη γραπτή επικοινωνία εξ' αποστάσεως»



Φοιτητές

ΑΙΚΑΤΕΡΙΝΗ ΛΟΥΛΟΥ **514082**

ΒΑΣΙΛΕΙΟΣ ΘΕΟΔΩΡΟΓΛΑΚΗΣ **513054**

Επιβλέπων

Δρ. Κυριάκος Τσιακμάκης

Φεβρουάριος 2021

Δημιουργία εφαρμογής για σύγχρονη γραπτή επικοινωνία εξ' αποστάσεως

Κωδικός: 20181

Φοιτήτρια: Αικατερίνη Λούλου

Φοιτητής: Βασίλειος Θεοδωρογλάκης

Εισηγητής: Δρ Κυριάκος Τσιακμάκης

Ημερομηνία ανάληψης Π.Ε. 10-07-2020

Ημερομηνία περάτωσης Π.Ε. 01-02-2021

Βεβαιώνω ότι είμαι ο συγγραφέας αυτής της εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, έχω καταγράψει τις όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών, εικόνων και κειμένου, είτε αυτές αναφέρονται ακριβώς είτε παραφρασμένες. Επιπλέον, βεβαιώνω ότι αυτή η εργασία προετοιμάστηκε από εμένα προσωπικά, ειδικά ως πτυχιακή εργασία, στο Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του ΔΙ.ΠΑ.Ε.

Η παρούσα εργασία αποτελεί πνευματική ιδιοκτησία της φοιτήτριας Αικατερίνη Λούλου και *του φοιτητή* Βασίλειου Θεοδωρογλάκη *που την εκπόνησε/αν. Στο πλαίσιο της πολιτικής ανοικτής πρόσβασης, ο συγγραφέας/δημιουργός εκχωρεί στο Διεθνές Πανεπιστήμιο της Ελλάδος άδεια χρήσης του δικαιώματος αναπαραγωγής, δανεισμού, παρουσίασης στο κοινό και ψηφιακής διάχυσης της εργασίας διεθνώς, σε ηλεκτρονική μορφή και σε οποιοδήποτε μέσο, για διδακτικούς και ερευνητικούς σκοπούς, άνευ ανταλλάγματος. Η ανοικτή πρόσβαση στο πλήρες κείμενο της εργασίας, δεν σημαίνει καθ' οιονδήποτε τρόπο παραχώρηση δικαιωμάτων διανοητικής ιδιοκτησίας του συγγραφέα/δημιουργού, ούτε επιτρέπει την αναπαραγωγή, αναδημοσίευση, αντιγραφή, πώληση, εμπορική χρήση, διανομή, έκδοση, μεταφόρτωση (downloading), ανάρτηση (uploading), μετάφραση, τροποποίηση με οποιονδήποτε τρόπο, τμηματικά ή περιληπτικά της εργασίας, χωρίς τη ρητή προηγούμενη έγγραφη συναίνεση του συγγραφέα/δημιουργού.*

Η έγκριση της πτυχιακής εργασίας από το Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του Διεθνούς Πανεπιστημίου της Ελλάδος, δεν υποδηλώνει απαραιτήτως και αποδοχή των απόψεων του συγγραφέα, εκ μέρους του Τμήματος.

Περίληψη

Η εργασία αυτή αφορά τη δημιουργία εφαρμογής για σύγχρονη γραπτή επικοινωνία εξ' αποστάσεως μεταξύ δύο χρηστών ή περισσότερων όταν επικοινωνούν μέσω ενός δωματίου. Για την εξυπηρέτηση ενός τμήματος ενός ιδρύματος ή μιας υπηρεσίας υπάρχει ένα κοινό δωμάτιο που μπορούν να επικοινωνήσουν όλοι οι χρήστες μεταξύ τους που είναι εγγεγραμμένοι και συνδεδεμένοι στο σύστημα-εφαρμογή. Επιπλέον κάθε χρήστης μπορεί να δημιουργήσει δωμάτιο και να συμπεριλάβει μέλη-χρήστες σε αυτό για να επικοινωνούν μεταξύ τους μόνοι όσοι ανήκουν σε αυτό. Η εφαρμογή υλοποιήθηκε σε περιβάλλον node.js με websockets, vue.js και βάση δεδομένων mySql.

« Message application for real-time, bidirectional and event-based communication »

Abstract

This work concerns the creation of an application for real-time, bidirectional and event-based communication between two users or more when communicating through a room. To serve a part of an institution or a work place there is a common room where all users who are registered and connected to the application system can communicate with each other. In addition, each user can create a room and include member-users in it to communicate with each other only those who belong to it. The application was implemented in node.js environment with websockets, vue.js and mySql database.

Ευχαριστίες

Θέλουμε να ευχαριστήσουμε τους γονείς μας για τη συμπαράσταση τους και τον επιβλέπων κ. Τσιακμάκη Κυριάκο για τη συνεχή επιστημονική και παιδαγωγική του καθοδήγηση και τη συμβολή του σε μέρη του κώδικα.

Περιεχόμενα

Κατάλογος Σχημάτων

Κεφάλαιο 1ο: Εισαγωγή

1.1 Εισαγωγή

Η εργασία αυτή αφορά τη δημιουργία εφαρμογής για σύγχρονη γραπτή επικοινωνία εξ' αποστάσεως μεταξύ δύο χρηστών ή περισσότερων όταν επικοινωνούν μέσω ενός δωματίου. Για την εξυπηρέτηση ενός τμήματος ενός ιδρύματος ή μιας υπηρεσίας υπάρχει ένα κοινό δωμάτιο που μπορούν να επικοινωνήσουν όλοι οι χρήστες μεταξύ τους που είναι εγγεγραμμένοι και συνδεδεμένοι στο σύστημα-εφαρμογή. Επιπλέον κάθε χρήστης μπορεί να δημιουργήσει δωμάτιο και να συμπεριλάβει μέλη-χρήστες σε αυτό για να επικοινωνούν μεταξύ τους μόνοι όσοι ανήκουν σε αυτό. Η εφαρμογή υλοποιήθηκε σε περιβάλλον node.js[1] με websockets[2], vue.js[3] και βάση δεδομένων mySql[4].

Η εφαρμογή είναι ένα website και ο χρήστης μπορεί να τη χρησιμοποιήσει μέσω ενός περιηγητή. Διαχωρίζεται σε δύο βασικά μέρη, το πρώτο αφορά το κομμάτι front-end που αφορά το επίπεδο παρουσίασης και προβολής των γραφικών στοιχείων, δηλαδή η ιστοσελίδα που κάνουμε περιήγηση. Το δεύτερο αφορά το κομμάτι του server που εξυπηρετεί το front-end και εκτελεί ενέργειες που έχουν σχέση περισσότερο με τη διαχείριση της βάσης. Η εφαρμογή υλοποιήθηκε σε περιβάλλον node.js με websockets, vue.js και βάση δεδομένων mySql.

Ο κύριος στόχος αυτής της εργασίας είναι η σύγχρονη γραπτή επικοινωνία εξ' αποστάσεως έχοντας αμφίδρομη επικοινωνία πραγματικού χρόνου μέσω διαδικτύου χρησιμοποιώντας websockets.

Συγκεκριμένα, οι επιμέρους στόχοι της εργασίας είναι:

Οι χρήστες που ανήκουν σε ένα εργασιακό χώρο να μπορούν να συνομιλήσουν όλοι μαζί σε ένα κοινό κανάλι-δωμάτιο.

Οι χρήστες να μπορούν να δημιουργήσουν κανάλια-δωμάτια και να συμπεριλάβουν άλλους χρήστες σε αυτά και να συνομιλήσουν όλοι μαζί σε πραγματικό χρόνο.

Οι χρήστες μπορούν να διαγράψουν τον εαυτό τους από κάποιο δωμάτιο.

Ο χρήστης που χρησιμοποιεί την ιστοσελίδα πρόσβασης στο σύστημα είναι εξουσιοδοτημένος.

1.1.1 Βιβλιογραφική Ανασκόπηση

Προτού αναλυθεί το σύστημα που υλοποιήθηκε στην εργασία θα αναφερθούν παρόμοια συστήματα που προτείνονται στη βιβλιογραφία και έχουν υλοποιηθεί.

Υπάρχουν πολλά chat συστήματα που υποστηρίζονται από μεγάλες εταιρείες όπως είναι το Viber, το Google Messages ή Allo, το facebook messenger - instagram, snapchat, whatsapp, discord κ.α.

Επίσης, υπάρχουν πάρα πολλές παρόμοιες εφαρμογές ανοιχτού κώδικα στα αποθετήρια σε διάφορες γλώσσες προγραμματισμού. Συγκεκριμένα για το framework vue.js με συνδυασμό sockets δεν υπάρχουν πολλά και θα αναφερθούμε σε τρία συγκεκριμένα.

Το πρώτο [5] χρησιμοποιεί απλώς το Socket.io για να στέλνει μηνύματα σε πραγματικό χρόνο αμφίδρομη επικοινωνία που βασίζεται σε συμβάντα. Το vue.js χρησιμοποιείται για τη σύνδεση δεδομένων στο DOM[6] και για τον εύκολο χειρισμό των δεδομένων.

Το δεύτερο [7] είναι ένα μικρό έργο με vue.js και sockets.io εξηγώντας τα πολύ βασικά και πολύ χρήσιμα πρώτα βήματα για την υλοποίηση ενός πρώτου chat επικοινωνίας.

Το τρίτο [8] προσεγγίζει ένα ολοκληρωμένα σύστημα γραπτής επικοινωνίας χρησιμοποιώντας Vue.js, Node.js (Express.js[9]), Nuxt.js[10], Vue-Socket.IO[11] (Socket.IO[12]), Vuetify.js[13]. Το κυριότερο χαρακτηριστικό του και η μεγάλη διαφορά με το σύστημα που παρουσιάζεται σε αυτήν την εργασία είναι ότι χρησιμοποιεί Nuxt.js για back-end server το οποίο είναι υλοποιημένο με vue.js. Στην εργασία μας χρησιμοποιούμε server που υλοποιείται από Express.js και socket.io.

1.2 Συνεισφορά της εργασίας

Στην εργασία παρουσιάζεται μια εφαρμογή-website για σύγχρονη γραπτή επικοινωνία εξ' αποστάσεως μεταξύ δύο χρηστών ή περισσότερων όταν επικοινωνούν μέσω ενός δωματίου. Η εργασία σχεδιάστηκε με τέτοιον τρόπο ώστε να χρησιμοποιηθεί αποκλειστικά από ένα Τμήμα ή Σχολή για την άμεση επικοινωνία μεταξύ των μελών του. Να μπορεί να υπάρχει ένα δωμάτιο για κάθε μάθημα που δημιουργείται από κάποιον εκπαιδευτικό και να συμπεριλάβει φοιτητές που τους ενδιαφέρει το μάθημα. Με αυτόν τον τρόπο θα μπορούσε να υπάρχει άμεση ενημέρωση των φοιτητών που παρακολουθούν-ανήκουν σε αυτό το μάθημα. Επίσης, θα μπορούσαν να καταθέσουν τη δική τους απορία ή να επικοινωνήσουν άμεσα με κάποιον εκπαιδευτικό μέσω ιδιωτικού μηνύματος.

Η εργασία μέσω του κώδικα και της ανάλυσης παρέχει χρήση web sockets μέσω της βιβλιοθήκης socket.io και πως μπορεί να συνδυαστεί με το vue.js. Παρέχει πληροφορίες και τεχνογνωσία για τη σύνδεση του front-end website με το back-end API server-side και πως αυτά επικοινωνούν μεταξύ τους μέσω sockets αλλά και url requests. Τέλος, δίνει μια λύση στη δημιουργία και στη χρήση των δωματίων-rooms στο socket.io, με ποιο τρόπο μπορούν να μιλήσουν σε ένα κοινό τους δωμάτιο και την ασφαλή επικοινωνία μέσω tokens.

1.3 Δομή της εργασίας

Στο πρώτο κεφάλαιο παρουσιάζεται μια μικρή εισαγωγή της εργασίας και οι στόχοι της και βιβλιογραφική ανασκόπηση παρόμοιων συστημάτων. Επιπρόσθετα, αναφέρεται η συνεισφορά της εργασίας και σε ποια σημεία επικεντρώνεται η ανάλυση της.

Στο δεύτερο κεφάλαιο περιγράφεται η τεχνολογία που χρησιμοποιήθηκε, το κύριο περιβάλλον node.js το framework vue.js και όλα τα υποστηρικτικά και βοηθητικά εργαλεία που ενσωματώνονται σε αυτό.

Στο τρίτο κεφάλαιο αναλύεται η εφαρμογή με σχήματα και κομμάτια κώδικα και διαγράμματα ενώ στο τέλος παρουσιάζεται και η βάση που χρησιμοποιήθηκε.

Στο τέταρτο κεφάλαιο παρουσιάζονται τα συμπεράσματα της εργασίας και θέματα για μελλοντική έρευνα ενώ στο τέλος της εργασίας παρατίθεται το παράρτημα με τους κώδικες που χρησιμοποιήθηκαν στην εργασία.

Κεφάλαιο 2ο: Τεχνολογία, πλατφόρμες και εργαλεία

2.1 Εισαγωγή

Η τεχνολογία που χρησιμοποιήθηκε είναι το κύριο περιβάλλον node.js και τα frameworks-πλατφόρμες που επιλέχθηκαν είναι το vue.js και το Express.js.

2.2 Node.js

Το Node.js είναι ένα περιβάλλον ανοιχτού κώδικα, που εκτελεί κώδικα JavaScript εκτός του προγράμματος περιήγησης. Συνήθως, το JavaScript χρησιμοποιείται κυρίως για scripting από την πλευρά του πελάτη, στο οποίο προγράμματα είναι γραμμένα σε JavaScript και ενσωματώνονται στην HTML της ιστοσελίδας και εκτελούνται από την πλευρά του πελάτη από μια μηχανή JavaScript στον περιηγητή ιστού του χρήστη.

Σχήμα 2.1: Node Λογότυπο

Το Node.js επιτρέπει στους προγραμματιστές να χρησιμοποιούν JavaScript για να γράψουν κώδικα στη γραμμή εντολών και για server-side scripting-running scripts και για να παράγουν δυναμικό περιεχόμενο ιστοσελίδας πριν η σελίδα φορτωθεί στο πρόγραμμα περιήγησης του χρήστη. Συνεπώς, το Node.js αντιπροσωπεύει ένα παράδειγμα <<JavaScript παντού>>, ενοποιώντας την ανάπτυξη web εφαρμογών γύρω από μια ενιαία γλώσσα προγραμματισμού, αντί για διαφορετικές γλώσσες για τις σελίδες διακομιστή και τις σελίδες πελάτη.

Παρόλο που η .js είναι η συμβατική επέκταση αρχείου για κώδικα JavaScript, το όνομα "Node.js" δεν αναφέρεται σε ένα συγκεκριμένο αρχείο και είναι απλώς το όνομα του προϊόντος. Το Node.js έχει μια αρχιτεκτονική που βασίζεται σε events-συμβάντα και είναι ικανή για ασύγχρονα I/O (input-output). Αυτές οι επιλογές σχεδιασμού στοχεύουν στη βελτιστοποίηση της απόδοσης και χρήσης σε εφαρμογές ιστού με πολλές λειτουργίες εισόδου / εξόδου, καθώς και για εφαρμογές Web σε πραγματικό χρόνο (π.χ. προγράμματα επικοινωνίας σε πραγματικό χρόνο και παιχνίδια με προγράμματα περιήγησης.

Το έργο Node.js, το οποίο διανέμεται από το Node.js Foundation, υποστηρίζεται από πρόγραμμα συνεργασίας με το Linux Foundation.

2.2.1 Που μπορεί να χρησιμοποιηθεί

Μερικές από τις περιοχές που μπορεί να χρησιμοποιηθεί είναι:

• Εφαρμογές συνδεδεμένες με I/O

• Εφαρμογές ροής δεδομένων

• Εφαρμογές δεδομένων σε πραγματικό χρόνο

• Εφαρμογές που βασίζονται σε API JSON

• Εφαρμογές μίας σελίδας

Δεν προτείνεται να χρησιμοποιηθεί από εφαρμογές που χρησιμοποιούν συνέχεια-εντατικά τον επεξεργαστή.

2.2.2 Ιστορία

Το Node.js γράφτηκε αρχικά από τον Ryan Dahl το 2009. Η αρχική έκδοση υποστηρίζει μόνο το Linux και το Mac OS X. Η ανάπτυξη και η συντήρησή του καθοδηγούνται από τον Dahl και αργότερα από τον Joyent. Ο Dahl εμπνεύστηκε για να δημιουργήσει το Node.js αφού είδε μια μπάρα προόδου upload file στο Flickr. Το πρόγραμμα περιήγησης δεν γνώριζε πόσο από το αρχείο είχε ανεβεί και έπρεπε να ερωτήσει τον διακομιστή Web. Ο Ντάλ επιθυμούσε έναν ευκολότερο τρόπο.

Ο Dahl επέκρινε τις περιορισμένες δυνατότητες του πιο δημοφιλέστερου διακομιστή ιστού το 2009, του Apache HTTP Server, για να χειριστεί πολλές ταυτόχρονες συνδέσεις (έως και 10.000) και τον πιο συνηθισμένο τρόπο δημιουργίας κώδικα (διαδοχικός προγραμματισμός). Το Node.js συνδυάζει τη μηχανή JavaScript V8 της Google, έναν βρόχο συμβάντων και ένα χαμηλού επιπέδου I/O API.

Μετά την πρώτη παρουσίαση το 2009, τον Ιανουάριο του 2010 έβαλε έναν διαχειριστή πακέτων για το περιβάλλον Node.js που ονομάζεται npm. [16] Ο διαχειριστής πακέτων διευκολύνει τους προγραμματιστές να δημοσιεύουν και να μοιράζονται τον πηγαίο κώδικα των βιβλιοθηκών Node.js και έχει σχεδιαστεί για να απλοποιεί την εγκατάσταση, την ενημέρωση και την απεγκατάσταση των βιβλιοθηκών. Η πρώτη έκδοση του Node.js που υποστηρίζει τα Windows κυκλοφόρησε τον Ιούλιο του 2011.

Τον Ιανουάριο του 2012, ο Dahl παραιτήθηκε, προωθώντας τον συνεργάτη και τον δημιουργό του Isaac Schlueter να διαχειριστεί το έργο. Τον Ιανουάριο του 2014, ο Schlueter ανακοίνωσε ότι ο Timothy J. Fontaine θα ηγηθεί του σχεδίου. Τον Φεβρουάριο του 2015 ανακοινώθηκε η πρόθεση να σχηματιστεί

ένα ουδέτερο Node.js Foundation. Μέχρι τον Ιούνιο του 2015, οι κοινότητες Node.js και io.js, μια παραλλαγή του, ψήφισαν να συνεργαστούν στο πλαίσιο του Node.js Foundation.

2.2.3 Αρχιτεκτονική

Το Node.js υποστηρίζει προγραμματισμό που βασίζεται σε γεγονότα-events σε διακομιστές ιστού - servers, επιτρέποντας την ανάπτυξη διακομιστών γρήγορου web με JavaScript. Οι προγραμματιστές μπορούν να δημιουργήσουν εξαιρετικά κλιμακούμενους διακομιστές χρησιμοποιώντας απλοποιημένο μοντέλο προγραμματισμού που βασίζεται σε συμβάντα και χρησιμοποιεί ανακλήσεις-callbacks για να σηματοδοτήσει την ολοκλήρωση μιας εργασίας.

Το Node.js χτίστηκε στη μηχανή JavaScript Google V8[14] αφού ήταν ανοιχτό με βάση την άδεια BSD[15]. Είναι εξαιρετικά γρήγορη και ικανή με βασικά στοιχεία του διαδικτύου όπως το HTTP. Επίσης, η JavaScript είναι μια πολύ γνωστή γλώσσα, καθιστώντας το Node.js άμεσα προσβάσιμο σε ολόκληρη την κοινότητα ανάπτυξης ιστού.

Υπάρχουν χιλιάδες βιβλιοθήκες ανοιχτού κώδικα για το Node.js, οι περισσότεροι από τους οποίους φιλοξενούνται στον ιστότοπο npm. [16]

Το Node.js είναι ένα περιβάλλον διακομιστή ανοιχτού κώδικα

Το Node.js είναι δωρεάν

Το Node.js εκτελείται σε διάφορες πλατφόρμες (Windows, Linux, Unix, Mac OS X κ.λπ.)

Το Node.js χρησιμοποιεί το JavaScript στον διακομιστή

Το Node.js είναι μια πολύ ισχυρή πλατφόρμα που βασίζεται στη μηχανή JavaScript του Google Chrome V8. Χρησιμοποιείται για την ανάπτυξη εφαρμογών Ιστού εισόδου / εξόδου, ιστότοπων ροής βίντεο, εφαρμογών μίας σελίδας και άλλων εφαρμογών ιστού. Το Node.js είναι ανοιχτού κώδικα, εντελώς δωρεάν και χρησιμοποιείται από χιλιάδες προγραμματιστές σε όλο τον κόσμο.

Το Node.js είναι open source, και συμβατό με όλες σχεδόν τις πλατφόρμες για την ανάπτυξη εφαρμογών διακομιστή και δικτύωσης. Οι εφαρμογές Node.js γράφονται σε JavaScript και μπορούν να εκτελεστούν στο Node.js σε OS X, Microsoft Windows και Linux.

Το Node.js παρέχει επίσης μια πλούσια βιβλιοθήκη διαφόρων ενοτήτων JavaScript που απλοποιεί την ανάπτυξη εφαρμογών ιστού χρησιμοποιώντας το Node.js σε μεγάλο βαθμό.

Τα πιο σημαντικά χαρακτηριστικά του node.js είναι:

Asynchronous and Event Driven

Όλα τα API[17] της βιβλιοθήκης Node.js είναι ασύγχρονα, δηλαδή δεν αποκλείονται. Αυτό ουσιαστικά σημαίνει ότι ένας διακομιστής που βασίζεται στο Node.js δεν περιμένει ποτέ ένα API για την επιστροφή δεδομένων. Ο διακομιστής μετακινείται στο επόμενο API αφού τον καλέσει και ένας μηχανισμός ειδοποιήσεων για τα συμβάντα του Node.js βοηθά τον διακομιστή να λάβει απάντηση από την προηγούμενη κλήση API.

Single Threaded but Highly Scalable

Το Node.js χρησιμοποιεί ένα μονό νήμα-thread με events-συμβάντα. Ο μηχανισμός συμβάντων βοηθά τον διακομιστή να αποκρίνεται με τρόπο χωρίς αποκλεισμό και καθιστά τον διακομιστή εξαιρετικά επεκτάσιμο σε αντίθεση με τους παραδοσιακούς διακομιστές που δημιουργούν περιορισμένα νήματα για τον χειρισμό αιτημάτων. Το Node.js χρησιμοποιεί ένα πρόγραμμα με ένα μόνο νήμα και το ίδιο πρόγραμμα μπορεί να παρέχει υπηρεσίες σε πολύ μεγαλύτερο αριθμό αιτημάτων από τους παραδοσιακούς διακομιστές όπως ο διακομιστής HTTP Apache

No Buffering

Οι εφαρμογές Node.js δεν αποθηκεύουν ποτέ δεδομένα. Αυτές οι εφαρμογές απλώς εξάγουν τα δεδομένα σε τεμάχια.

2.2.4 Οδηγός εγκατάστασης και χρήσης

1. Κατεβάζουμε το αρχείο node από την κεντρική ιστοσελίδα του. [18]

2. Δημιουργούμε ένα js αρχείο με όνομα main.js στον υπολογιστή (Windows or Linux) και γράφουμε τον ακόλουθο κώδικα.

3.

```
console.log("Hello, World!")
```

4.

Εκτελούμε το πρόγραμμα γράφοντας σε console το παρακάτω:

```
node main.js
```

Θα δούμε στην οθόνη το εξής αποτέλεσμα:

Hello, World!

Στη συνέχεια μπορούμε να δημιουργήσουμε έναν απλό server.

Για αυτό το λόγο πρέπει να συμπεριλάβουμε ένα συγκεκριμένο module, το http.

Στην αρχή του κώδικα γράφουμε:

var http = require("http");

Και στη συνέχεια γράφουμε τον παρακάτω κώδικα για τη δημιουργία server.

```
http.createServer(function (request, response) {
  response.writeHead(200, {'Content-Type': 'text/plain'});

  // Επιστροφή ως response body το κείμενο "Hello World"
  response.end('Hello World\n');
}).listen(8081);

// Θα εκτυπωθεί στην οθόνη Console
console.log('Server running at http://127.0.0.1:8081/');
```

Ανοίγουμε το http://127.0.0.1:8081/ ή http://localhost:8081/ και παρατηρούμε το αποτέλεσμα.

Αυτό θα είναι όπως το παρακάτω

Σχήμα 2.2: Σελίδα όταν φορτώνουμε τον κώδικα του παραδείγματος

Έχουμε δημιουργήσει έναν τοπικό server που εξυπηρετεί στη Θύρα 8081 .

Αν θέλουμε να εγκαταστήσουμε μια επιπλέον βιβλιοθήκη θα πρέπει να χρησιμοποιήσουμε τον npm.

Μια από τις πιο σημαντικές βιβλιοθήκες είναι η express.js.

2.3 Express.js

Το Express.js είναι ένα ελάχιστο και ευέλικτο πλαίσιο εφαρμογών ιστού με Node.js που παρέχει ένα ισχυρό σύνολο χαρακτηριστικών για την ανάπτυξη εφαρμογών ιστού και κινητών. Διευκολύνει την ταχεία ανάπτυξη εφαρμογών Web ιδίως για υλοποίηση server.

Ακολουθούν μερικά από τα βασικά χαρακτηριστικά του πλαισίου Express.

- Επιτρέπει τη ρύθμιση των ενδιάμεσων λογισμικών για την απόκριση σε αιτήματα HTTP.
- Ορίζει έναν πίνακα δρομολόγησης που χρησιμοποιείται για την εκτέλεση διαφορετικών ενεργειών με βάση τη μέθοδο HTTP και τη διεύθυνση URL.
- Επιτρέπει τη δυναμική απόδοση σελίδων HTML με βάση τη μετάδοση ορισμάτων σε πρότυπα.

Εγκατάσταση Express

Πρώτον, εγκαθιστούμε το πλαίσιο Express χρησιμοποιώντας NPM, έτσι ώστε να μπορεί να χρησιμοποιηθεί για τη δημιουργία μιας εφαρμογής ιστού χρησιμοποιώντας τερματικό.

$ npm εγκατάσταση express –save

Η παραπάνω εντολή αποθηκεύει την εγκατάσταση τοπικά στον κατάλογο node_modules και δημιουργεί έναν κατάλογο express μέσα στο node_modules.

Πρέπει να εγκαταστήσουμε τις ακόλουθες σημαντικές ενότητες μαζί με το express

- body-parser - Αυτό είναι ένα ενδιάμεσο λογισμικό node.js για το χειρισμό δεδομένων φόρμας κωδικοποιημένων σε JSON, Raw, Text και URL.
- cookie-parser - Parse Cookie header και παράγει req.cookies
- multer - Πρόκειται για ένα ενδιάμεσο λογισμικό node.js για το χειρισμό πολλαπλών στοιχείων / μορφών δεδομένων.

Οι εντολές είναι:

$ npm install body-parser --save

$ npm install cookie-parser --save

$ npm install multer –save

Ένα πακέτο μπορεί να επαναεγκατασταθεί με την παρακάτω εντολή

npm uninstall express

Ένα πακέτο μπορεί να γίνει update(όταν υπάρχει διαθέσιμο) με την παρακάτω εντολή

$ npm update express

Χρησιμοποιώντας τη βιβλιοθήκη express μπορούμε να δημιουργήσουμε έναν server πολύ απλά

Γράφοντας τα παρακάτω κομμάτι κώδικα σε ένα αρχείο server.js

```
var express = require('express');
var app = express();

app.get('/', function (req, res) {
  res.send('Hello World');
})
```

```
var server = app.listen(8081, function () {
  var host = server.address().address
  var port = server.address().port

  console.log("Example app listening at http://%s:%s", host, port)
})
```

Και να τον τρέξουμε με την εντολή

node server.js

Το express παρέχει πάρα πολλές ευκολίες όσο αφορά την διαχείριση των routers, αξιοποίησης των παραμέτρων από τα Get και Post requests.

Η εφαρμογή Express χρησιμοποιεί μια συνάρτηση callback της οποίας οι παράμετροι είναι αντικείμενα request και response.

```
app.get('/', function (req, res) {
  // --
})
```

Request Object - Το αντικείμενο αίτησης αντιπροσωπεύει το αίτημα HTTP και έχει ιδιότητες για τη συμβολοσειρά ερωτήματος, παραμέτρους, σώμα, κεφαλίδες HTTP.

Response Object - Το αντικείμενο απόκρισης αντιπροσωπεύει την απόκριση HTTP που στέλνει μια εφαρμογή Express όταν λαμβάνει ένα αίτημα HTTP.

Έχουμε δει μια βασική εφαρμογή που εξυπηρετεί Request αίτημα HTTP για την αρχική σελίδα. Η δρομολόγηση αναφέρεται στον προσδιορισμό του τρόπου με τον οποίο μια εφαρμογή ανταποκρίνεται σε ένα αίτημα πελάτη σε ένα συγκεκριμένο τελικό σημείο, το οποίο είναι ένα URI (ή διαδρομή) και μια συγκεκριμένη μέθοδο αιτήματος HTTP (GET, POST και ούτω καθεξής).

Θα επεκτείνουμε το πρόγραμμα Hello World για να χειριστούμε περισσότερους τύπους Request HTTP.

```
var express = require('express');
var app = express();
```

```
// Οταν καλούμε το http://127.0.0.1:8081/ στο περιηγητή
app.get('/', function (req, res) {
//η επιστροφή
  res.send('Hello GET');
})

// Οταν καλούμε το http://127.0.0.1:8081/users στο περιηγητή
app.get('/users', function (req, res) {
  var userlist = [];
  res.send(userlist);
})

// Οταν καλούμε το http://127.0.0.1:8081/user στο περιηγητή
app.post('/user', function (req, res) {
  //με post δημιουργούμε
  res.send("Success");
})


var server = app.listen(8081, function () {
  var host = server.address().address
  var port = server.address().port
port)
})
```

Επειδή δεν θέλουμε να αναλύσουμε περισσότερο το express ο αναγνώστης μπορεί να επισκεφθεί ιστοχώρους με επιπλέον παραδείγματα όπως είναι ο [19].

2.4 Vue.js

2.4.1 Εισαγωγή

Για να δημιουργήσουμε ιστοσελίδες με το node υπάρχουν πολλά frameworks. Ένα από αυτά είναι το Vue.js.

Το Vue.js είναι ένα ανοιχτού κώδικα μοντέλο - view-model και είναι front end JavaScript framework για την κατασκευή διεπαφών χρήστη και εφαρμογών μίας σελίδας. Δημιουργήθηκε από τον Evan You και διατηρείται από αυτόν και τα υπόλοιπα ενεργά μέλη της ομάδας.

Το Vue.js διαθέτει μια σταδιακά προσαρμόσιμη αρχιτεκτονική που εστιάζει στο δηλωτικό rendering και τη σύνθεση των συστατικών. Η βασική βιβλιοθήκη εστιάζεται μόνο στο επίπεδο προβολής. Προηγμένες δυνατότητες που απαιτούνται για σύνθετες εφαρμογές, όπως δρομολόγηση, διαχείριση κατάστασης και εργαλεία κατασκευής προσφέρονται μέσω επίσημα υποστηρικτικών βιβλιοθηκών και πακέτων όπως είναι το Nuxt.js ως μία από τις πιο δημοφιλείς λύσεις.

Το Vue.js σάς επιτρέπει να επεκτείνετε HTML με χαρακτηριστικά HTML που ονομάζονται directives. Οι directives προσφέρουν λειτουργικότητα σε εφαρμογές HTML και έρχονται είτε ως ενσωματωμένες είτε καθοριζόμενες από τον χρήστη.

Το Vue δημιουργήθηκε από τον Evan You αφού εργάστηκε για το Google χρησιμοποιώντας το AngularJS[20] σε διάφορα έργα. Η Vue κυκλοφόρησε για πρώτη φορά τον επόμενο Φεβρουάριο, το 2014.

Templates

Η Vue χρησιμοποιεί μια σύνταξη προτύπου που βασίζεται σε HTML που επιτρέπει τη σύνδεση του DOM που αποδίδεται στα δεδομένα της Vue. Όλα τα πρότυπα Vue είναι έγκυρα HTML που μπορούν να αναλυθούν από προγράμματα περιήγησης συμβατά με προδιαγραφές και προγράμματα ανάλυσης HTML. Η Vue συγκεντρώνει τα πρότυπα σε λειτουργίες DOM. Ένα εικονικό μοντέλο αντικειμένου εγγράφου (ή "DOM") επιτρέπει στο Vue να αποδίδει στοιχεία στη μνήμη του πριν από την ενημέρωση του προγράμματος περιήγησης. Σε συνδυασμό με το σύστημα αντιδραστικότητας, η Vue είναι σε θέση να υπολογίσει τον ελάχιστο αριθμό στοιχείων για εκ νέου απόδοση και εφαρμογή του ελάχιστου αριθμού χειρισμών DOM όταν αλλάζει η κατάσταση της εφαρμογής.

Οι χρήστες της Vue μπορούν να χρησιμοποιήσουν σύνταξη προτύπου ή να επιλέξουν να γράψουν απευθείας συναρτήσεις απόδοσης χρησιμοποιώντας το JSX. [21]

Reactivity

Το Vue διαθέτει ένα σύστημα αντιδραστικότητας που χρησιμοποιεί απλά αντικείμενα JavaScript και βελτιστοποιημένη rendering. Κάθε στοιχείο παρακολουθεί τις εξαρτήσεις του κατά τη διάρκεια του rendering, έτσι το σύστημα γνωρίζει με ακρίβεια πότε πρέπει να επανασχεδιάσει και ποια στοιχεία πρέπει να κάνει rendering ξανά.

Transitions

Η Vue παρέχει μια ποικιλία κλάσεων για την εφαρμογή των εφέ μετάβασης κατά την εισαγωγή, ενημέρωση ή αφαίρεση στοιχείων από το DOM.

Αυτό περιλαμβάνει εργαλεία για:

Εφαρμογή αυτοματισμών για μεταβάσεις και κινούμενες εικόνες με CSS

Βιβλιοθήκες CSS τρίτων κατασκευαστών, όπως το Animate.css[22]

Χρήση JavaScript για άμεσο χειρισμό του DOM κατά τη διάρκεια.

Ενσωμάτωση βιβλιοθηκών κινούμενων σχεδίων τρίτων κατασκευαστών, όπως το Velocity.js[23]

Όταν ένα στοιχείο εισάγεται ή αφαιρείται, τότε το Vue θα καταλαβαίνει αυτόματα εάν το στοιχείο αλλάζει σε CSS ιδιότητες.

Routing

Ένα συνηθισμένο μειονέκτημα των εφαρμογών μίας σελίδας (SPA [24]) είναι η αδυναμία κοινής χρήσης συνδέσμων προς μια σελίδα sub (κάποια εσωτερική σελίδα). Επειδή οι σελίδες SPA εξυπηρετούν στους χρήστες τους μόνο ένα response που βασίζεται στη διεύθυνση URL (συνήθως εξυπηρετεί index.html ή index.vue), η κοινή χρήση συνδέσμων σε συγκεκριμένες ενότητες είναι συνήθως δύσκολη, αν όχι αδύνατη. Για την επίλυση αυτού του προβλήματος, πολλοί δρομολογητές από την πλευρά του πελάτη οριοθετούν τις δυναμικές διευθύνσεις URL τους με ένα "hashbang" (#!), Π.χ. σελίδα.com/#!/. Ωστόσο, με το HTML5 τα περισσότερα σύγχρονα προγράμματα περιήγησης υποστηρίζουν δρομολόγηση χωρίς κατακερματισμούς.

Η Vue παρέχει μια διεπαφή για να αλλάξει αυτό που εμφανίζεται στη σελίδα με βάση την τρέχουσα διαδρομή URL - ανεξάρτητα από το πώς άλλαξε (είτε μέσω συνδέσμου μέσω email, ανανέωσης ή συνδέσμων εντός σελίδας). Επιπλέον, η χρήση ενός δρομολογητή front-end επιτρέπει τη μετάβαση της διαδρομής του προγράμματος περιήγησης όταν συμβαίνουν συγκεκριμένα συμβάντα προγράμματος περιήγησης (δηλαδή κλικ) σε κουμπιά ή συνδέσμους. Το Vue δεν έρχεται με κατακερματισμένη δρομολόγηση front-end. Αλλά το πακέτο ανοιχτού κώδικα "vue-router" [25] παρέχει ένα API για την ενημέρωση της διεύθυνσης URL της εφαρμογής, υποστηρίζει το κουμπί επιστροφής (ιστορικό πλοήγησης).

2.5 Vuex

2.5.1 Εισαγωγή

Το Vuex[26] είναι ένα μοτίβο διαχείρισης κατάστασης + βιβλιοθήκη για εφαρμογές Vue.js. Χρησιμεύει ως κεντρικό store για όλα τα στοιχεία της εφαρμογής, με κανόνες που διασφαλίζουν ότι η κατάσταση μπορεί να αλλάξει μόνο με συγκεκριμένο τρόπο. Ενσωματώνεται επίσης με την επίσημη επέκταση devtools της Vue (ανοίγει νέο παράθυρο) για να παρέχει προηγμένες δυνατότητες όπως εντοπισμός σφαλμάτων time-travel μηδενικής διαμόρφωσης και εξαγωγή / εισαγωγή στιγμιότυπου.

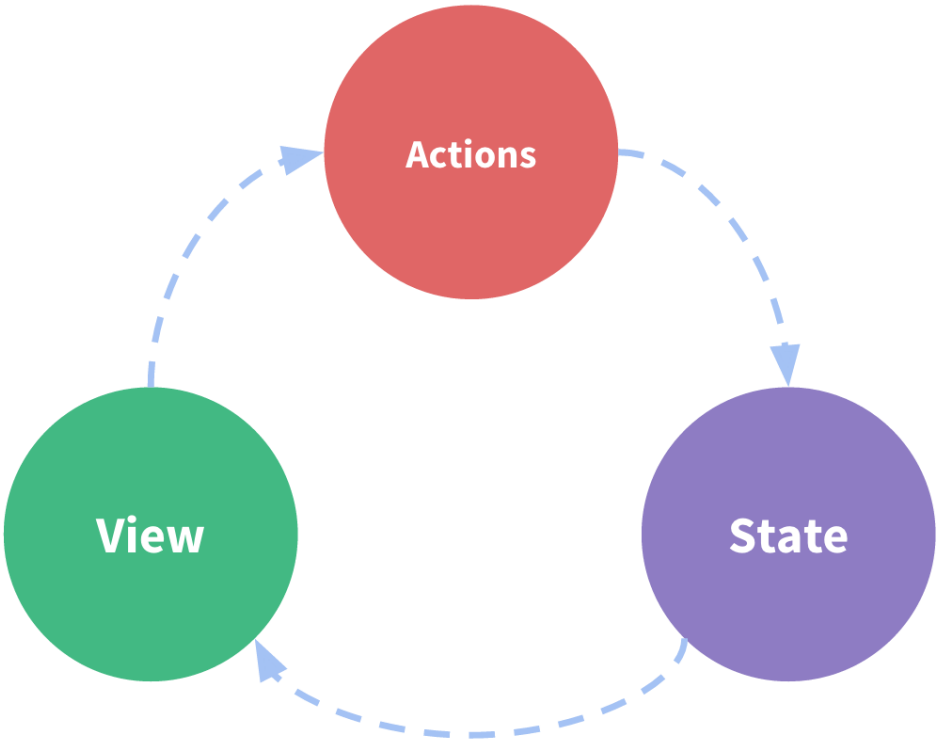
Είναι μια εφαρμογή με τα ακόλουθα μέρη:

Το **state**, η πηγή που οδηγεί την εφαρμογή μας.

Το **view**, η χαρτογράφηση του state

Τα **actions**, οι πιθανοί τρόποι με τους οποίους η κατάσταση - state θα μπορούσε να αλλάξει ως αντίδραση στις εισόδους του χρήστη.

Αυτή είναι μια απλή αναπαράσταση της έννοιας της "one-way data flow":



Σχήμα 2.3: Vuex μέρη

Ωστόσο, η απλότητα εξαφανίζεται γρήγορα όταν έχουμε πολλά στοιχεία που μοιράζονται ένα κοινό state.

Τα διάφορα Views ενδέχεται να εξαρτώνται από το ίδιο state.

Τα actions από διαφορετικά Views θα χρειαστούν επιπλέον συναρτήσεις για να έχουν πρόσβαση στο ίδιο κομμάτι του state.

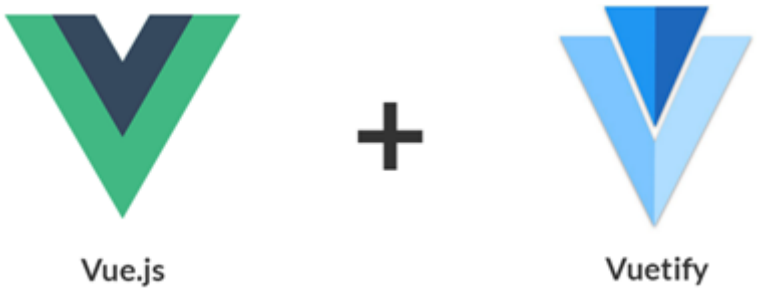
2.6 Vuetify

2.6.1 Εισαγωγή

Το Vuetify[13] είναι ένα πλήρες πλαίσιο διεπαφής χρήστη (UI framework) που δημιουργήθηκε πάνω από το Vue.js. Ο στόχος του έργου είναι να παρέχει στους προγραμματιστές τα εργαλεία που χρειάζονται για να δημιουργήσουν πλούσιες και ενδιαφέρουσες εμπειρίες χρηστών. Σε αντίθεση με άλλα πλαίσια, το Vuetify έχει σχεδιαστεί από τη βάση προς τα πάνω ώστε να είναι εύκολο κάποιος να

το μάθει. Περιέχει πάρα πολλά γραφικά εξαρτήματα που μπορούν να εισαχθούν εύκολα σε μια ιστοσελίδα και το σημαντικότερο είναι ότι έχουν responsive ιδιότητες.

Το Vuetify ακολουθεί μια πρώτη προσέγγιση για σχεδίαση για κινητά, πράγμα που σημαίνει ότι η εφαρμογή σας λειτουργεί out of box, είτε πρόκειται για τηλέφωνο, tablet ή επιτραπέζιο υπολογιστή.



Σχήμα 2.4: Το Vuetify δημιουργήθηκε για το Vue.js

Από την αρχική του κυκλοφορία το 2014, το Vue.js έχει εξελιχθεί σε ένα από τα πιο δημοφιλή πλαίσια JavaScript στον κόσμο. Ένας από τους λόγους αυτής της δημοτικότητας είναι η ευρεία χρήση στοιχείων που επιτρέπουν στους προγραμματιστές να δημιουργήσουν συνοπτικές modules που θα χρησιμοποιηθούν και θα επαναχρησιμοποιηθούν σε όλη την εφαρμογή τους. Οι βιβλιοθήκες UI είναι συλλογές αυτών των modules που εφαρμόζουν μια συγκεκριμένη οδηγία στυλ και παρέχουν τα απαραίτητα εργαλεία για τη δημιουργία εκτεταμένων εφαρμογών ιστού.

Το Vuetify έχει αναπτυχθεί ακριβώς σύμφωνα με τις προδιαγραφές του Material Design με κάθε συστατικό σχολαστικά κατασκευασμένο ώστε να είναι responsive και αποτελεσματικό. Προσαρμόστε την εφαρμογή σας με μοναδικές και δυναμικές διατάξεις και προσαρμόστε τα στυλ των στοιχείων σας χρησιμοποιώντας μεταβλητές SASS.

Το Vuetify έχει έναν πολύ ενεργό κύκλο ανάπτυξης και διορθώνεται εβδομαδιαίως, ανταποκρινόμενο σε ζητήματα και αναφορές της κοινότητας με πρωτοποριακή ταχύτητα, επιτρέποντάς να χειρίζεστε συχνά τις διορθώσεις σφαλμάτων και τις βελτιώσεις. Επιπλέον, κάθε σημαντική έκδοση συνοδεύεται από μακροπρόθεσμη υποστήριξη 18 μηνών για την προηγούμενη δευτερεύουσα έκδοση.

Περιέχει μια πολύ μεγάλη δωρεάν συλλογή από UI Components που εφαρμόζονται στο Vue.

2.7 Sockets.io

Το Socket.IO[12] είναι μια βιβλιοθήκη JavaScript για εφαρμογές ιστού σε πραγματικό χρόνο. Επιτρέπει σε πραγματικό χρόνο, αμφίδρομη επικοινωνία μεταξύ πελατών και διακομιστών Ιστού. Διαθέτει δύο μέρη: μια βιβλιοθήκη από την πλευρά του πελάτη που εκτελείται στο πρόγραμμα περιήγησης και μια βιβλιοθήκη από την πλευρά του διακομιστή για το Node.js. Και τα δύο συστατικά έχουν σχεδόν πανομοιότυπο API. Όπως το Node.js, βασίζεται σε συμβάντα.

Το Socket.IO χρησιμοποιεί κυρίως το πρωτόκολλο WebSocket με pooling ως εναλλακτική λύση, [3] παρέχοντας ταυτόχρονα την ίδια διεπαφή. Παρόλο που μπορεί να χρησιμοποιηθεί ως απλό ‘περιτύλιγμα’ για το WebSocket, παρέχει πολλές ακόμη δυνατότητες, όπως μετάδοση σε πολλά sockets, με αποθήκευση δεδομένων που σχετίζονται με κάθε πελάτη και ασύγχρονο I/O. Μπορεί να εγκατασταθεί με το εργαλείο npm.

Το Socket.IO παρέχει τη δυνατότητα εφαρμογής σε πραγματικό χρόνο αναλύσεων, streaming, άμεσων μηνυμάτων και συνεργασίας εγγράφων. Κάποιοι αξιοσημείωτοι χρήστες είναι το Microsoft Office, το Yammer [27] και το Zendesk [28]. Το Socket.IO χειρίζεται τη σύνδεση μεταξύ client-server στο background. Μπορεί να γίνει αυτόματα αναβάθμιση σε WebSocket, όταν απαιτείται. Επίσης, απαιτεί από τον προγραμματιστή να έχει μόνο γνώσεις Socket.IO.

Το Socket.IO δεν είναι βιβλιοθήκη WebSocket [29] με εναλλακτικές επιλογές σε άλλα πρωτόκολλα πραγματικού χρόνου. Είναι μια προσαρμοσμένη εφαρμογή πρωτοκόλλου μεταφοράς σε πραγματικό χρόνο πάνω από άλλα πρωτόκολλα πραγματικού χρόνου. Ένας διακομιστής υλοποίησης Socket.IO δεν μπορεί να συνδεθεί σε πελάτη που δεν είναι Socket.IO WebSocket Ένας πελάτης με Socket.IO δεν μπορεί να μιλήσει σε διακομιστή που δεν είναι Socket.IO WebSocket. Το Socket.IO απαιτεί τη χρήση βιβλιοθηκών Socket.IO τόσο από πλευράς πελάτη όσο και από διακομιστή.

Από την έκδοση 2.0, το Socket.IO χρησιμοποιεί το WebSockets ως την βασική βιβλιοθήκη WebSocket.

Κεφάλαιο 3ο: Η εφαρμογή - Το σύστημα επικοινωνίας

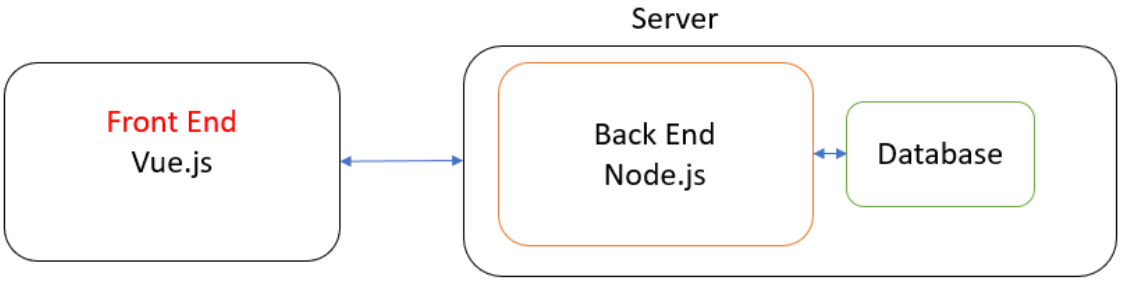
3.1 Εισαγωγή

Το σύστημα επικοινωνίας διακρίνεται σε δύο μέρη. Το πρώτο αφορά το front-end μέρος, το οποίο είναι η ιστοσελίδα που βλέπει ο χρήστης στον browser και το δεύτερο αφορά το back-end σύστημα με το οποίο επικοινωνεί το front με αυτό και εκτελεί τις σημαντικές λειτουργίες όπως είναι η εξουσιοδότηση, κρυπτογράφηση και η επικοινωνία με τη βάση.

Το front εκτελείται μόνο στον browser του χρήστη, δηλαδή από τη μεριά του πελάτη.

Το back εκτελείται στον server.

Το node.js χρησιμοποιήθηκε και για τα δύο και αποτελεί τη βάση και τη τεχνολογία για το έργο.



Σχήμα 3.1: Το σύστημα – η εφαρμογή

Για το front χρησιμοποιήθηκε το Vue.js.

Οι τεχνολογίες που χρησιμοποιήθηκαν και αναφέρθηκαν στο προηγούμενο κεφάλαιο είναι το

Vuetify – για χρήση όμορφων, εύχρηστων, responsive εξαρτημάτων (όπως κουμπιών, στήλες, διαλόγων κτλ)

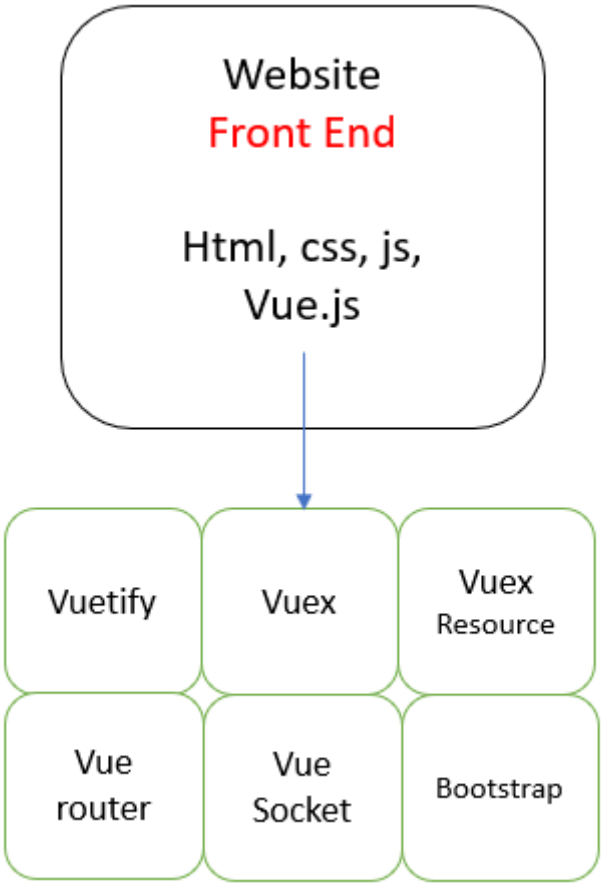
Vuex – για διαμοιρασμό των μεταβλητών μεταξύ σελίδων και χρήση κοινών συναρτήσεων

Vue Resource – παρέχει υπηρεσίες για να δημιουργούμε requests

Vue Router – για τον εύκολο χειρισμό των διαδρομών url - routes

Vue Socket – για τον χειρισμό του socket.io μέσα στην vue.js

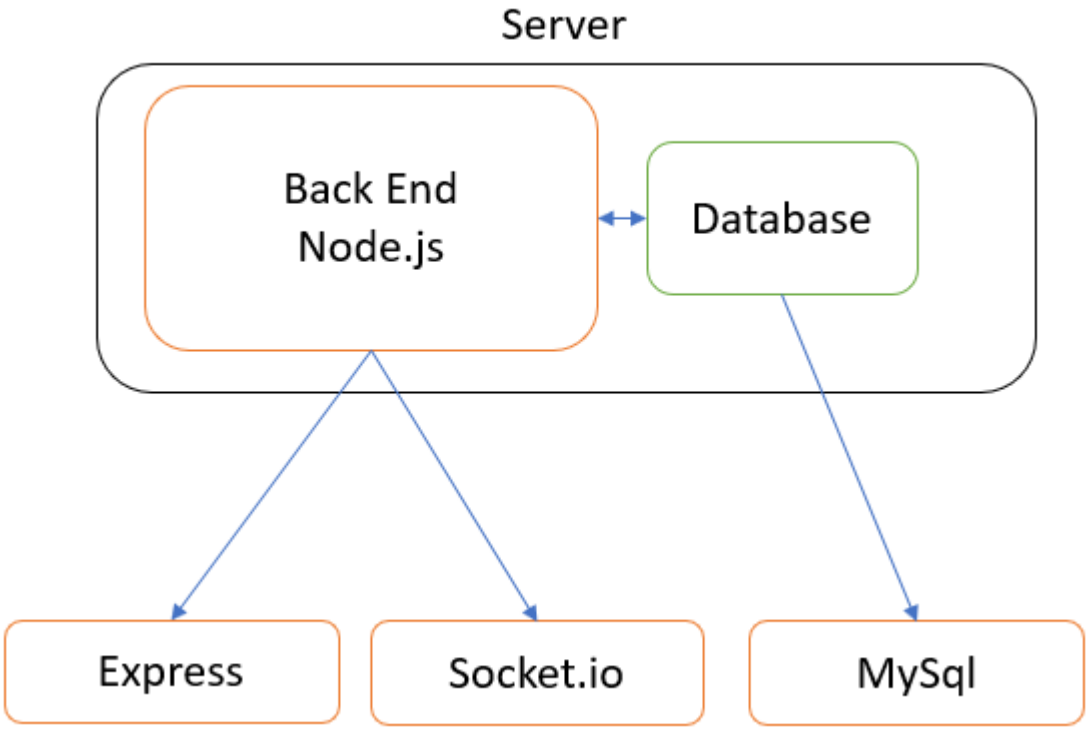
Boostrap [30] - Όπως το Vuetify είναι μια συλλογή εργαλείων ανοιχτού κώδικα (ελεύθερο λογισμικό) για τη δημιουργία ιστοσελίδων και διαδικτυακών εφαρμογών. Περιέχει HTML και CSS για τις μορφές τυπογραφίας, κουμπιά πλοήγησης και άλλων στοιχείων του περιβάλλοντος, καθώς και προαιρετικές επεκτάσεις JavaScript.



Σχήμα 3.2: Οι τεχνολογίες που χρησιμοποιήθηκαν για την κατασκευή της ιστοσελίδας

Το back-end πρόγραμμα δημιουργήθηκε με node.js και συγκεκριμένα χρησιμοποιήθηκε το Express framework για εύκολη διαχείρισή του server και η βιβλιοθήκη socket.io για τα websockets.

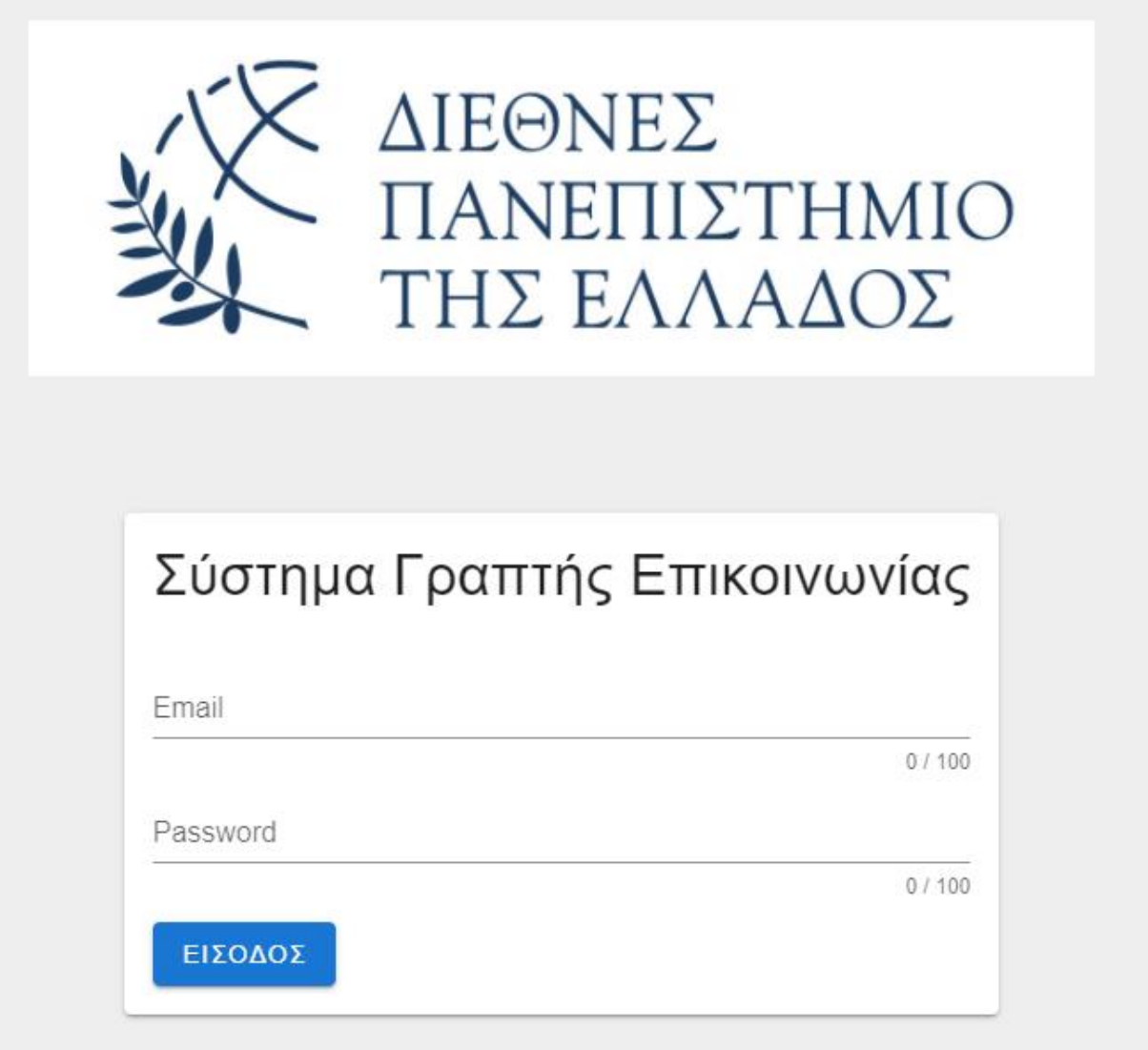
Επίσης χρησιμοποιήθηκε η mysql για την εγκατάσταση της βάσης δεδομένων.



Σχήμα 3.3: Οι τεχνολογίες που χρησιμοποιήθηκαν για την κατασκευή του server

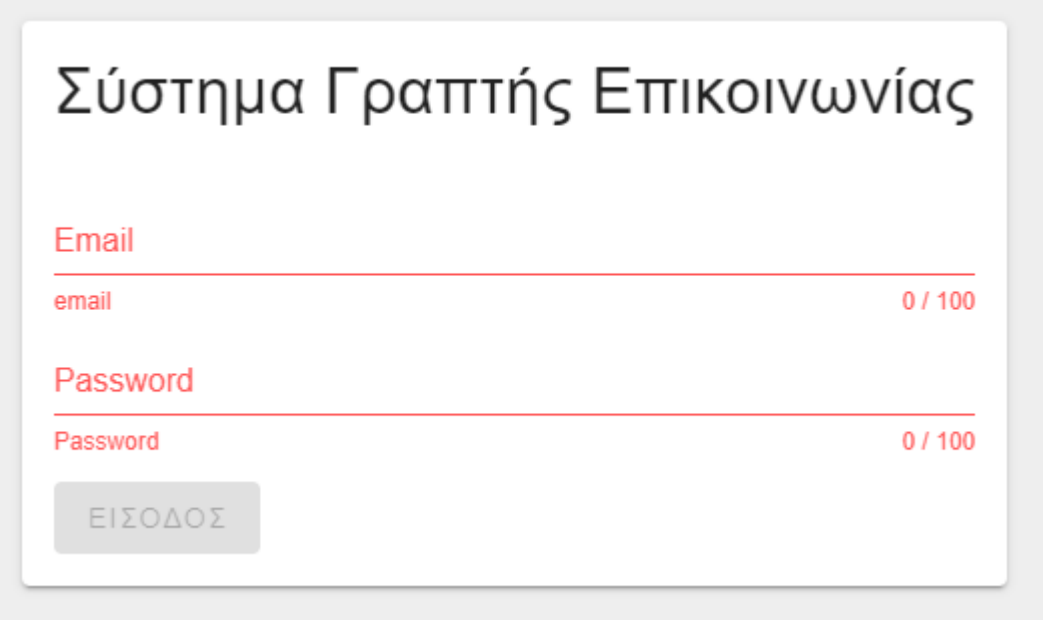
3.2 Ο ιστοχώρος - Προβολή

Στο Σχήμα 3.4 παρουσιάζεται η σελίδα που βλέπει ο μη συνδεδεμένος χρήστης για να κάνει σύνδεση.

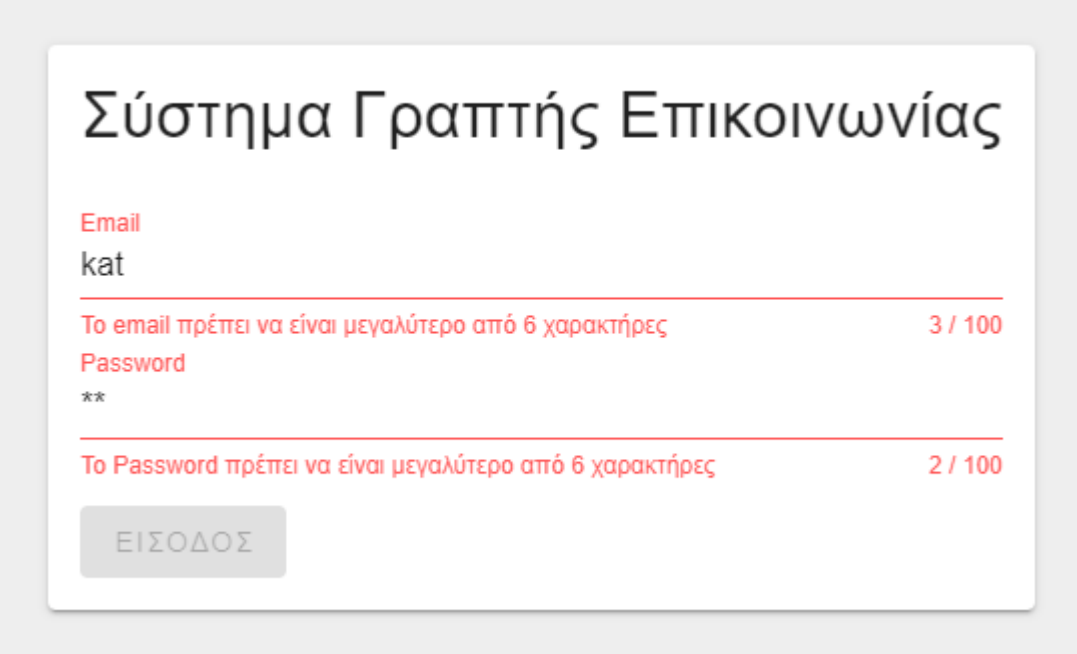


Σχήμα 3.4: Η σελίδα που βλέπει ο μη συνδεδεμένος χρήστης για να κάνει σύνδεση

Στα Σχήματα 3.5 και 3.6 παρουσιάζονται οι σελίδες που βλέπει χρήστης όταν δεν ικανοποιούνται οι προϋποθέσεις εισαγωγής στα πεδία.

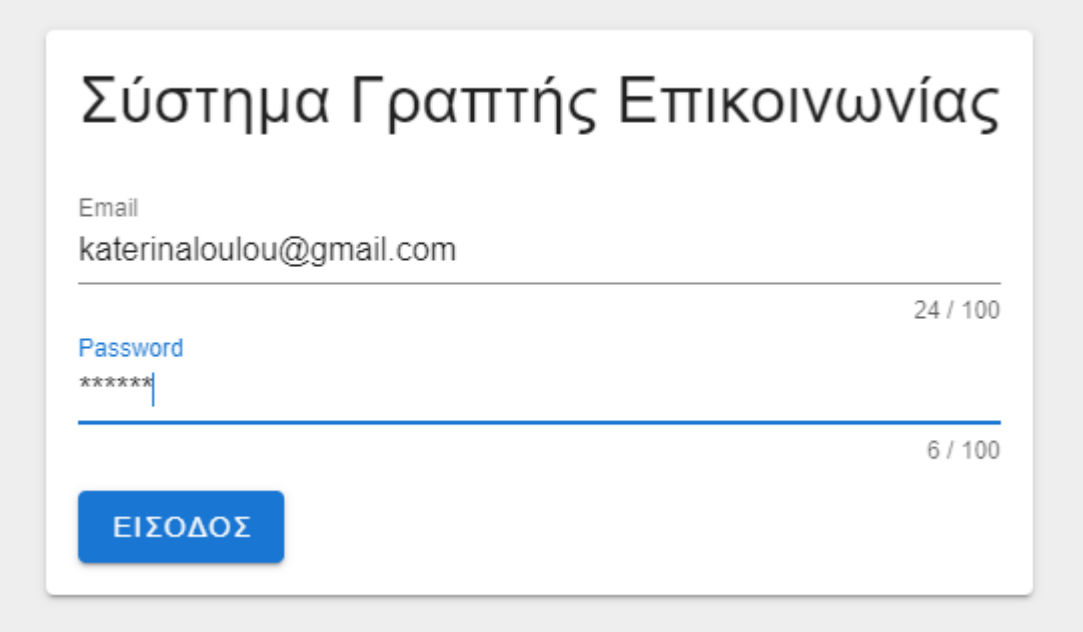


Σχήμα 3.5: Η σελίδα που βλέπει χρήστης όταν δεν ικανοποιούνται οι προϋποθέσεις εισαγωγής στα πεδία – Κενά πεδία



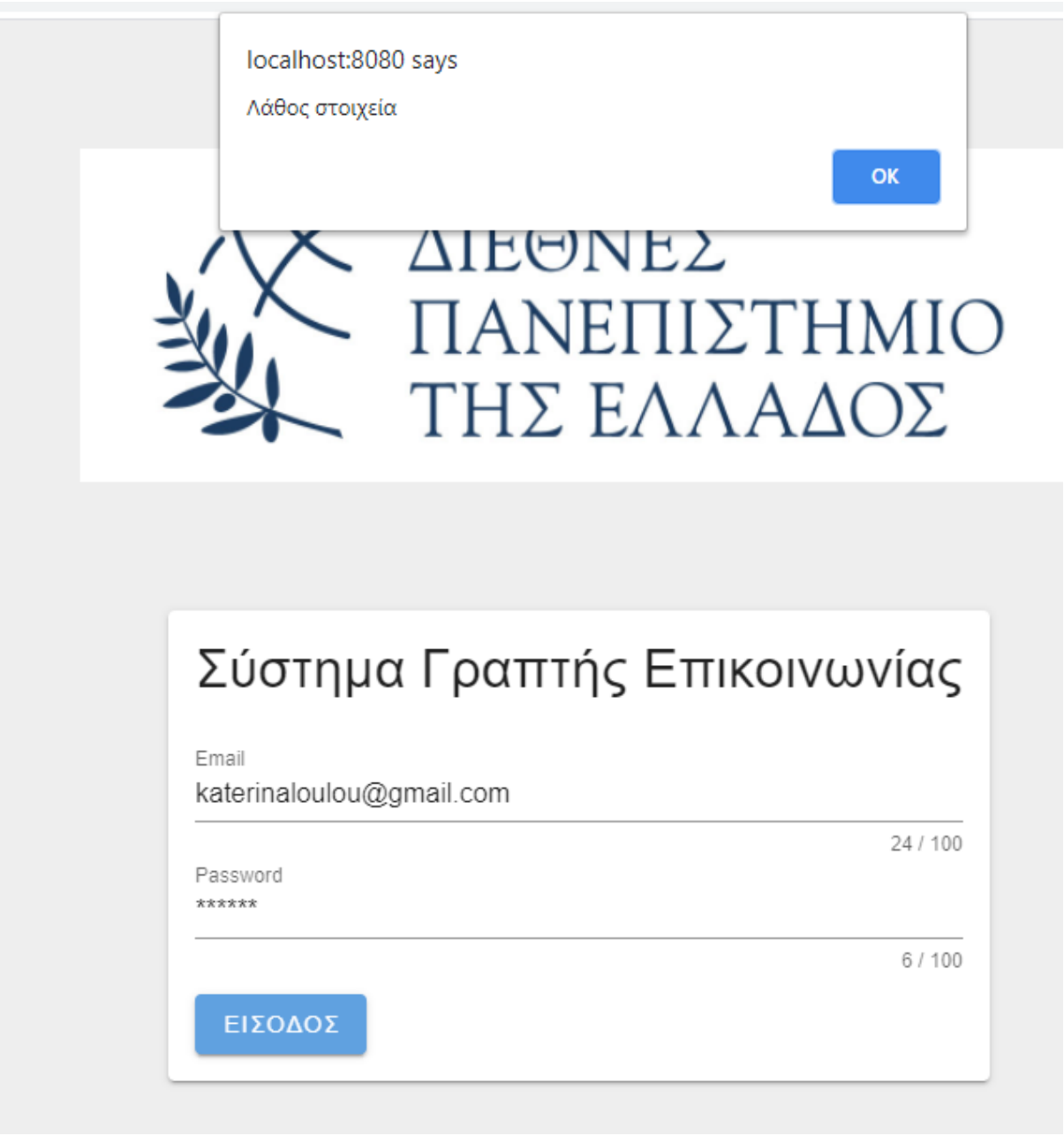
Σχήμα 3.6: Η σελίδα που βλέπει χρήστης όταν δεν ικανοποιούνται οι προϋποθέσεις εισαγωγής στα πεδία

Στο Σχήμα 3.7 παρουσιάζεται η σελίδα που βλέπει χρήστης όταν ικανοποιούνται οι προϋποθέσεις εισαγωγής στα πεδία.



Σχήμα 3.7: Η σελίδα που βλέπει χρήστης όταν ικανοποιούνται οι προϋποθέσεις εισαγωγής στα πεδία

Στο Σχήμα 3.8 παρουσιάζεται η σελίδα που βλέπει χρήστης όταν πατάει λανθασμένα τα στοιχεία ταυτοποίησης για την είσοδο στο σύστημα.



Σχήμα 3.8: Η σελίδα που βλέπει χρήστης όταν πατάει λανθασμένα τα στοιχεία ταυτοποίησης για την είσοδο στο σύστημα

Όταν ο χρήστης εισάγει τα σωστά στοιχεία του – email και password τότε δρομολογείται στη κεντρική σελίδα του συστήματος που βλέπει την chat σελίδα όπως φαίνεται στο Σχήμα 3.9.

Σχήμα 3.9: Η κεντρική σελίδα του συστήματος

Σχήμα 3.10: Η περιοχή εμφάνισης των μηνυμάτων και του πεδίου γραφής νέου μηνύματος

Η κεντρική σελίδα χωρίζεται σε τρείς βασικές περιοχές.

Η πρώτη και μεσαία περιοχή φαίνεται στο Σχήμα 3.10. Σε αυτήν παρουσιάζονται τα μηνύματα αλλά περιέχει και το πεδίο όπου ο χρήστης γράφει ένα νέο μήνυμα που θέλει να στείλει.

Σχήμα 3.11: Περιοχή με τους χρήστες

Η αριστερή περιοχή, όπως φαίνεται στο Σχήμα 3.11 περιέχει:

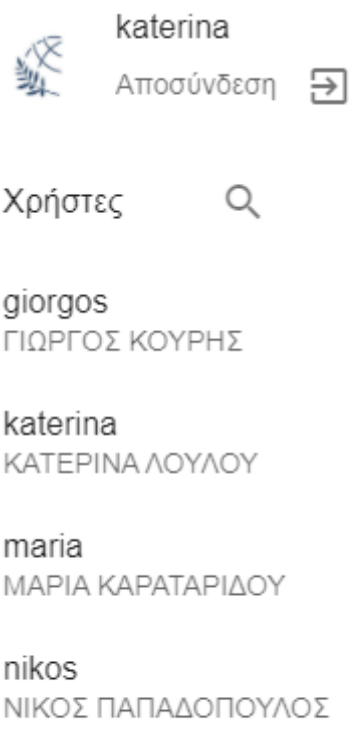
- Ποιος χρήστης είναι συνδεδεμένος
- Το κουμπί αποσύνδεσης
- Τη λίστα με του χρήστες – φίλους
- Το κουμπί για αναζήτηση χρήστη

‘Όπως φαίνεται στο Σχήμα 3.12 η λίστα περιέχει τους φίλους του χρήστη, username-alias και ονοματεπώνυμο με αλφαβητική σειρά όσον αφορά τα ονόματα.

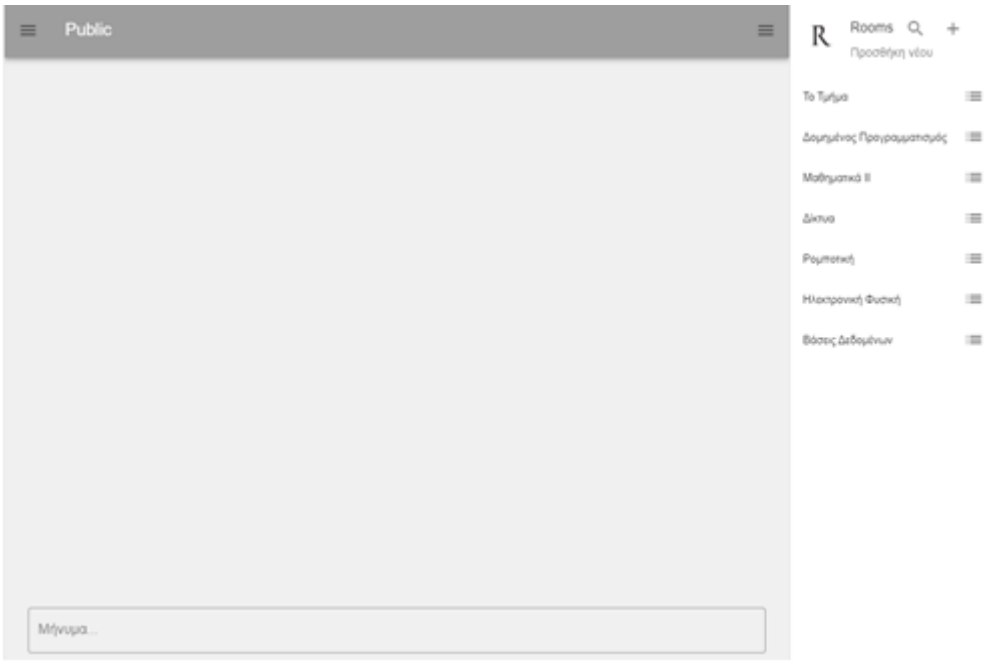
Η δεξιά περιοχή, όπως φαίνεται στο Σχήμα 3.13 περιέχει:

- Το κεντρικό δωμάτιο που είναι όλοι συνδεδεμένοι – Το Τμήμα
- Τα δωμάτια του χρήστη
- Τα δωμάτια που είναι μέλος ο χρήστης
- Το κουμπί για την δημιουργία νέου δωματίου
- Το κουμπί για αναζήτηση δωματίου

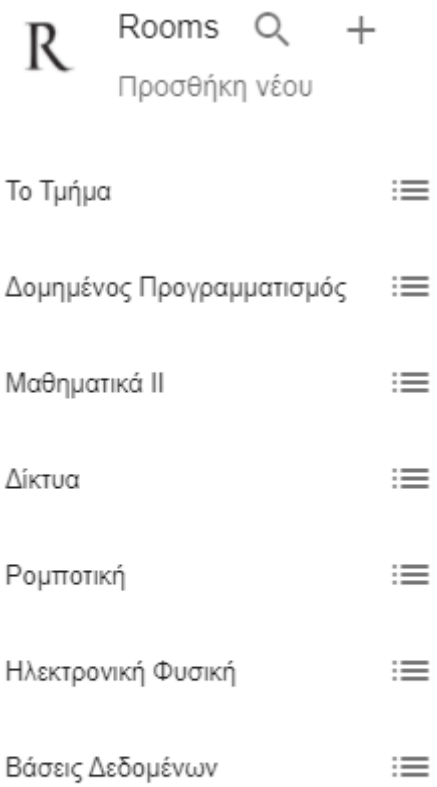
‘Όπως φαίνεται στο Σχήμα 3.14 η λίστα περιέχει τα δωμάτια του χρήστη και σε αυτά που είναι μέλος με σειρά ταξινόμησης σύμφωνα με τη δημιουργία τους.



Σχήμα 3.12: Λίστα με τους Χρήστες



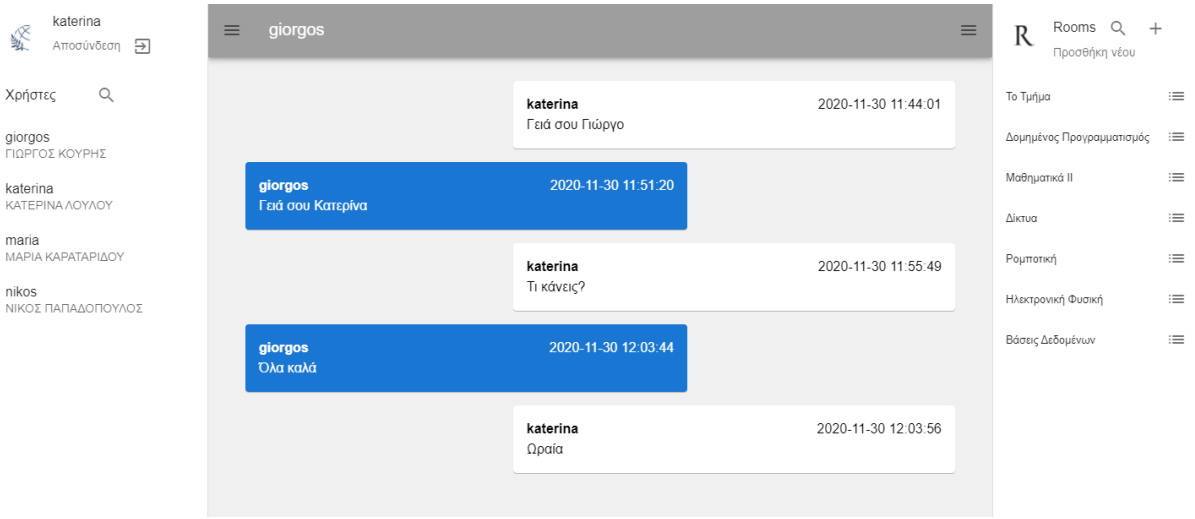
Σχήμα 3.13: Περιοχή με τα Δωμάτια



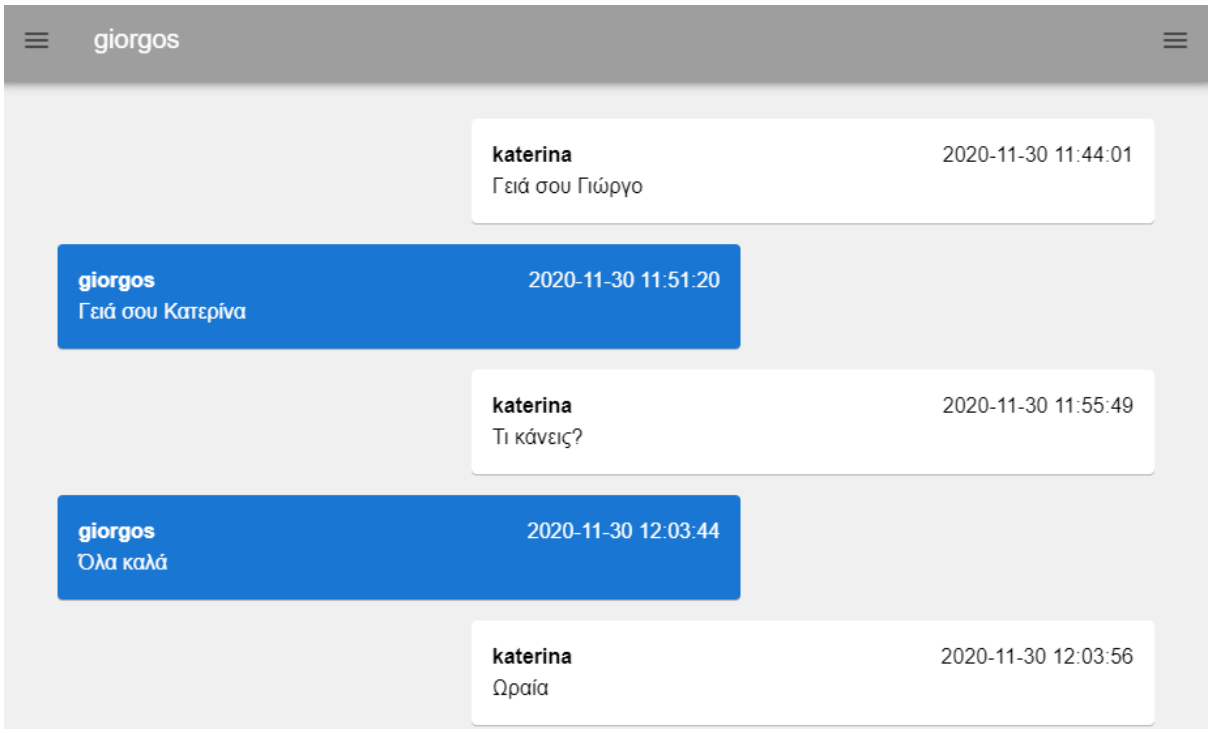
Σχήμα 3.14: Λίστα διαθέσιμων δωματίων για τον συνδεδεμένο χρήστη

Στα Σχήμα 3.15 και 3.16 παρουσιάζονται ενδεικτικά μηνύματα μεταξύ του χρήστη katerina και του giorgios.

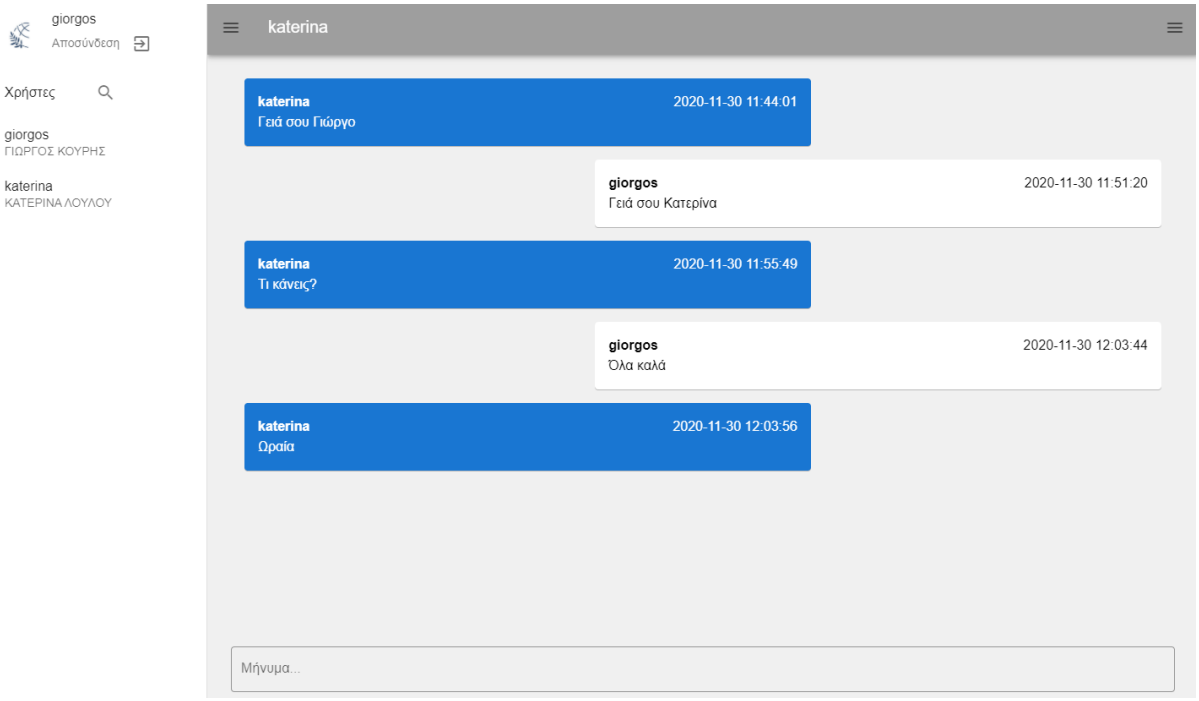
Το μπλε χρώμα δηλώνει ότι προέρχεται από άλλον χρήστη.



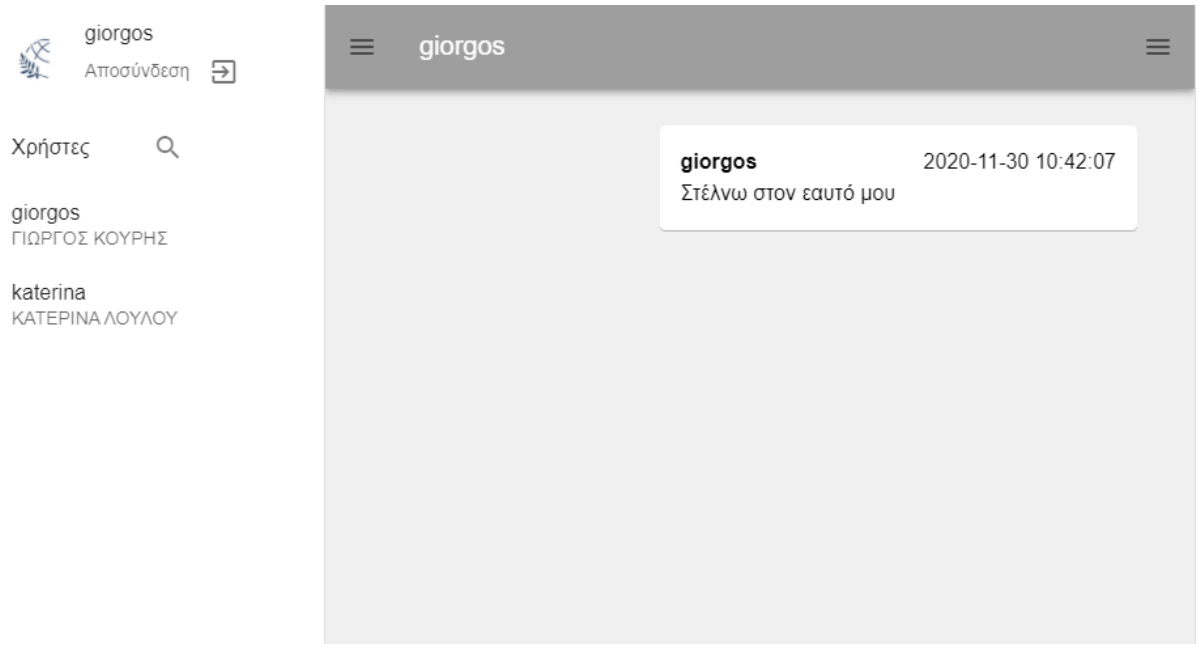
Σχήμα 3.15: Ενδεικτικά μηνύματα μεταξύ του χρήστη katerina



Σχήμα 3.16: Ενδεικτικά μηνύματα μεταξύ του χρήστη Katerina - μεγέθυνση

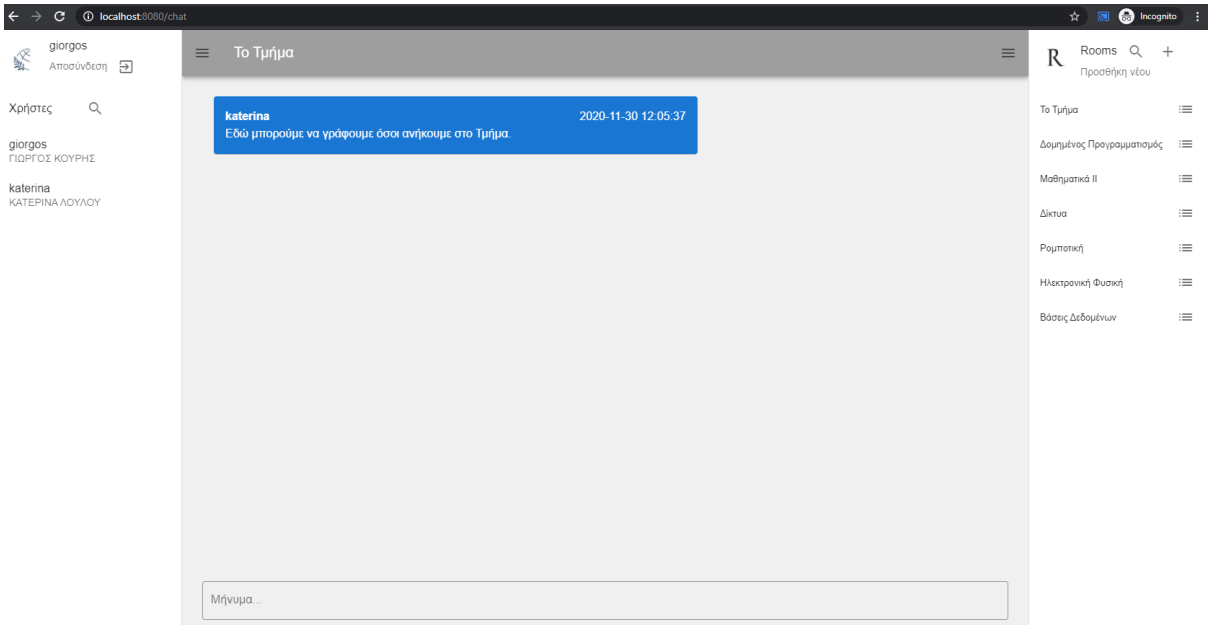


Σχήμα 3.17: Ενδεικτικά μηνύματα μεταξύ του χρήστη giorgios



Σχήμα 3.18: Ενδεικτικά μηνύματα μεταξύ του χρήστη giorgios – αποστολή στον εαυτό του

Στα Σχήμα 3.17 και 3.18 παρουσιάζονται ενδεικτικά μηνύματα μεταξύ του χρήστη katerina και του giorgios όταν ο κεντρικός συνδεδεμένος χρήστης είναι ο giorgios. Φαίνεται ότι ο giorgios έχει μόνο την katerina για φίλη.



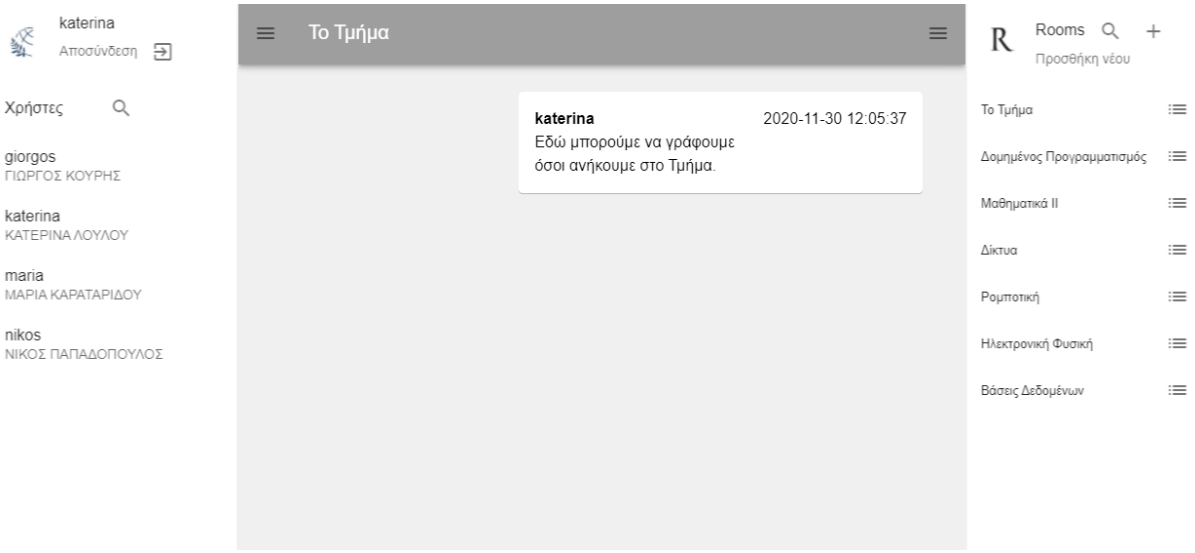
Σχήμα 3.19: Ενδεικτικά μηνύματα του Δωματίου: Το Τμήμα όπου γράφουν όλοι με συνδεδεμένο χρήστη τον giorgios - 1

Στα Σχήματα 3.19 και 3.20 παρουσιάζονται ενδεικτικά μηνύματα του Δωματίου: Το Τμήμα όπου γράφουν όλοι με συνδεδεμένο χρήστη τον giorgios.

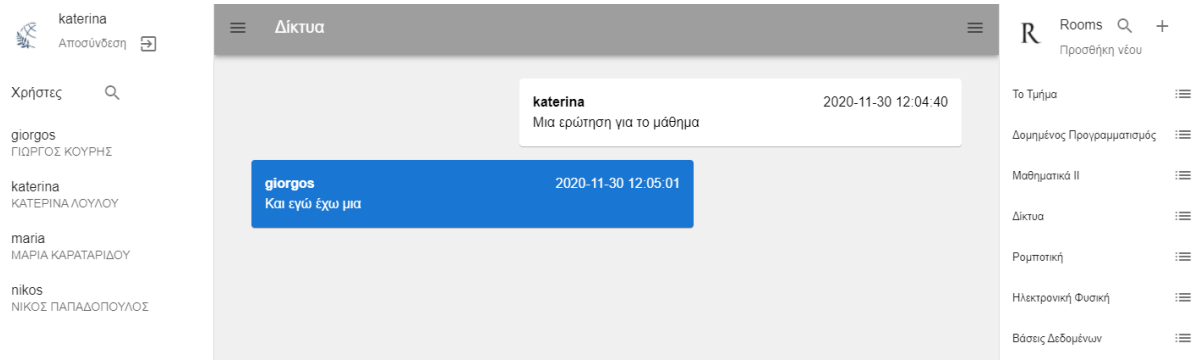
Το μπλε χρώμα δηλώνει ότι προέρχεται από άλλον χρήστη.

Στα Σχήματα 3.21 και 3.22 παρουσιάζονται ενδεικτικά μηνύματα του Δωματίου: Το Τμήμα όπου γράφουν όλοι με συνδεδεμένο χρήστη την katerina.

Σχήμα 3.20: Ενδεικτικά μηνύματα του Δωματίου: Το Τμήμα όπου γράφουν όλοι με συνδεδεμένο χρήστη την giorgios - 2

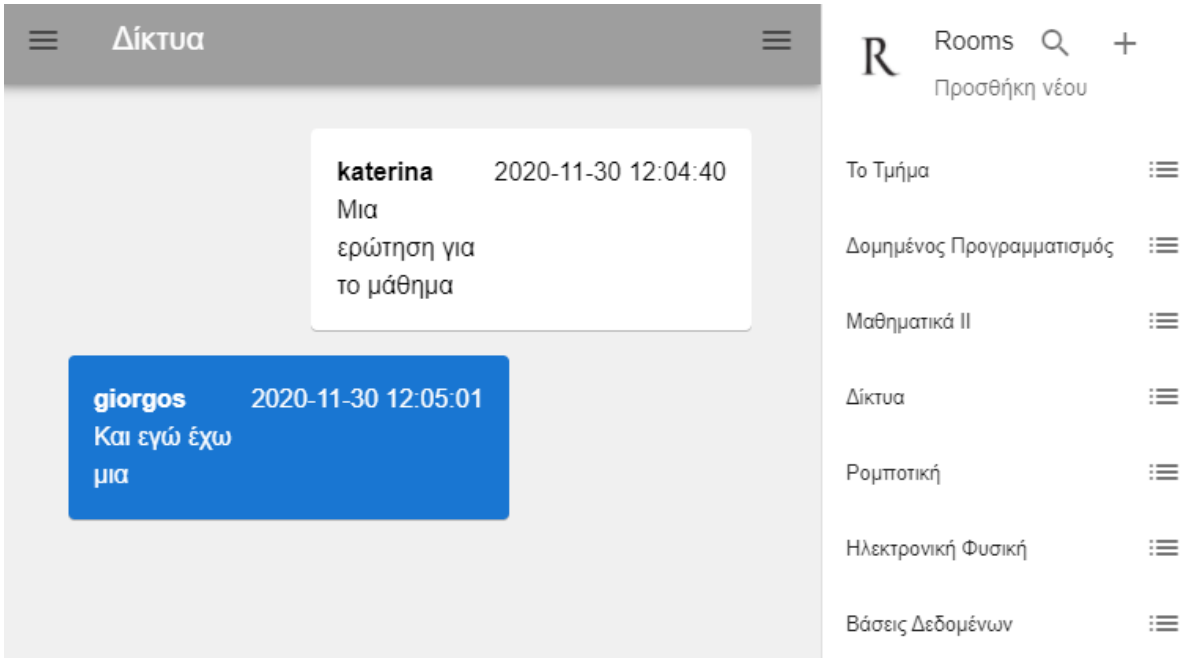


Σχήμα 3.21: Ενδεικτικά μηνύματα του Δωματίου: Το Τμήμα όπου γράφουν όλοι με συνδεδεμένο χρήστη την katerina

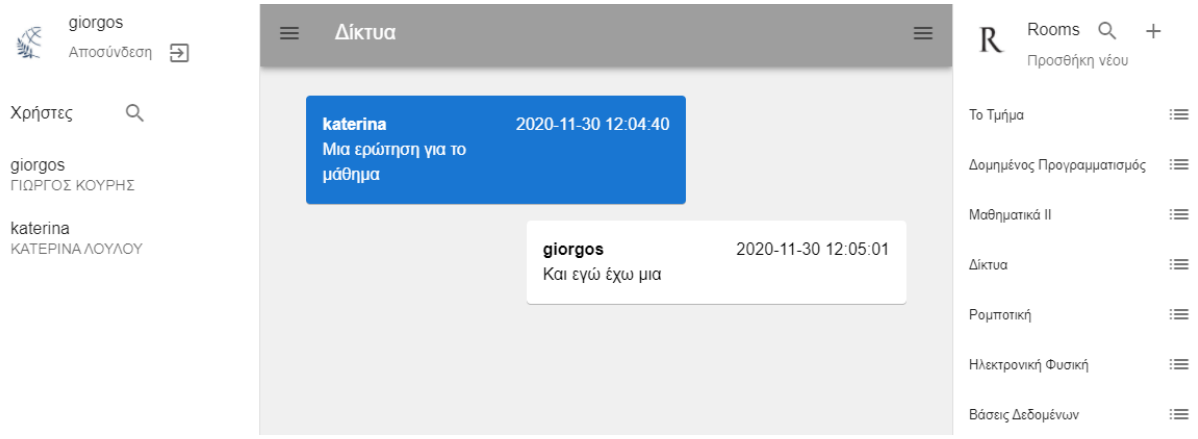


Σχήμα 3.22: Ενδεικτικά μηνύματα του Δωματίου Δίκτυα όπου γράφουν όλοι με συνδεδεμένο χρήστη την Katerina - 1

Στα Σχήματα 3.22 και 3.23 παρουσιάζονται ενδεικτικά μηνύματα του Δωματίου Δίκτυα όπου γράφουν όλοι με συνδεδεμένο χρήστη την katerina.

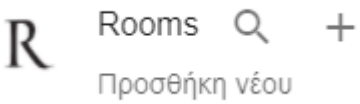


Σχήμα 3.23: Ενδεικτικά μηνύματα του Δωματίου Δίκτυα όπου γράφουν όλοι με συνδεδεμένο χρήστη την Katerina - 2

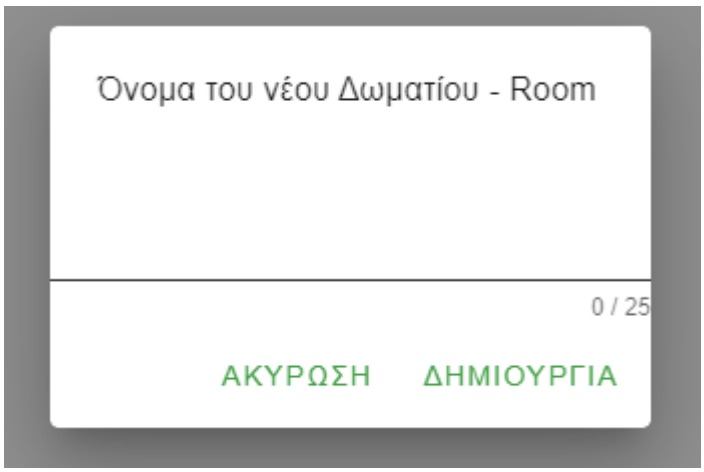


Σχήμα 3.24: ενδεικτικά μηνύματα του Δωματίου Δίκτυα όπου γράφουν όλοι με συνδεδεμένο χρήστη την giorgios

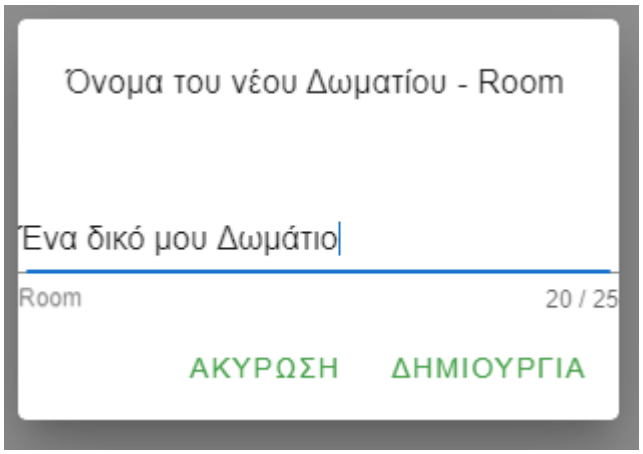
Στα Σχήματα 3.25, 3.26, 3.27 παρουσιάζονται το κουμπί και ο διάλογος για τη Δημιουργία νέου Δωματίου.



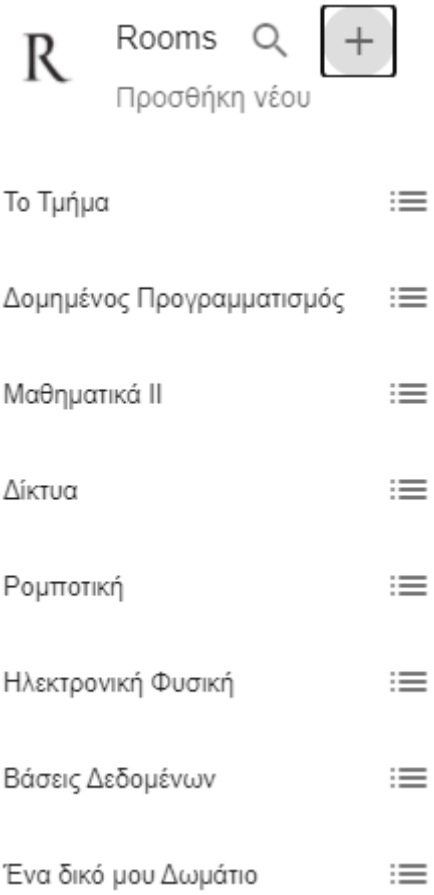
Σχήμα 3.25: Κουμπί για προσθήκη του νέου δωματίου



Σχήμα 3.26: Διάλογος για προσθήκη του νέου δωματίου - 1

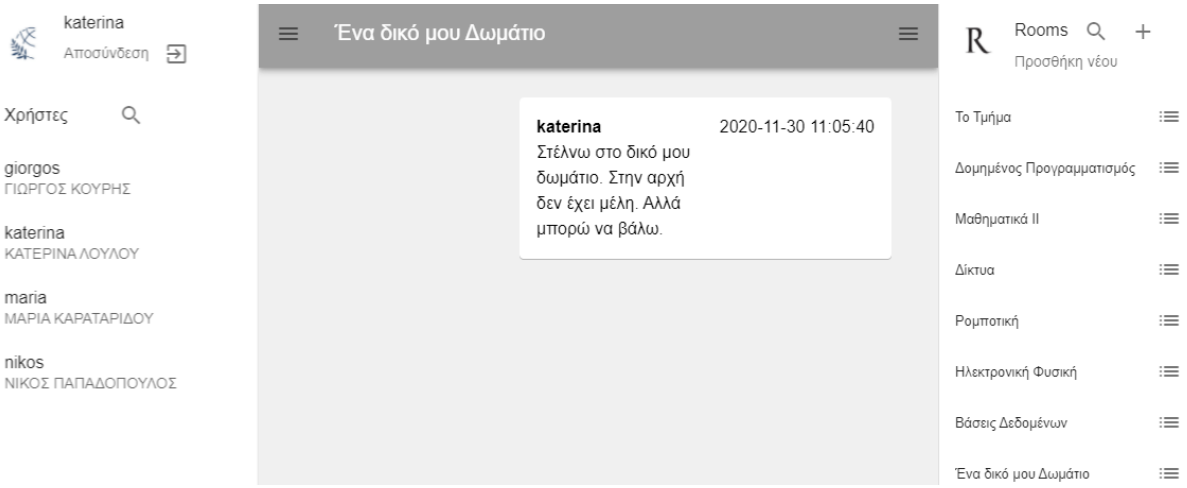


Σχήμα 3.27: Διάλογος για προσθήκη του νέου δωματίου - 2



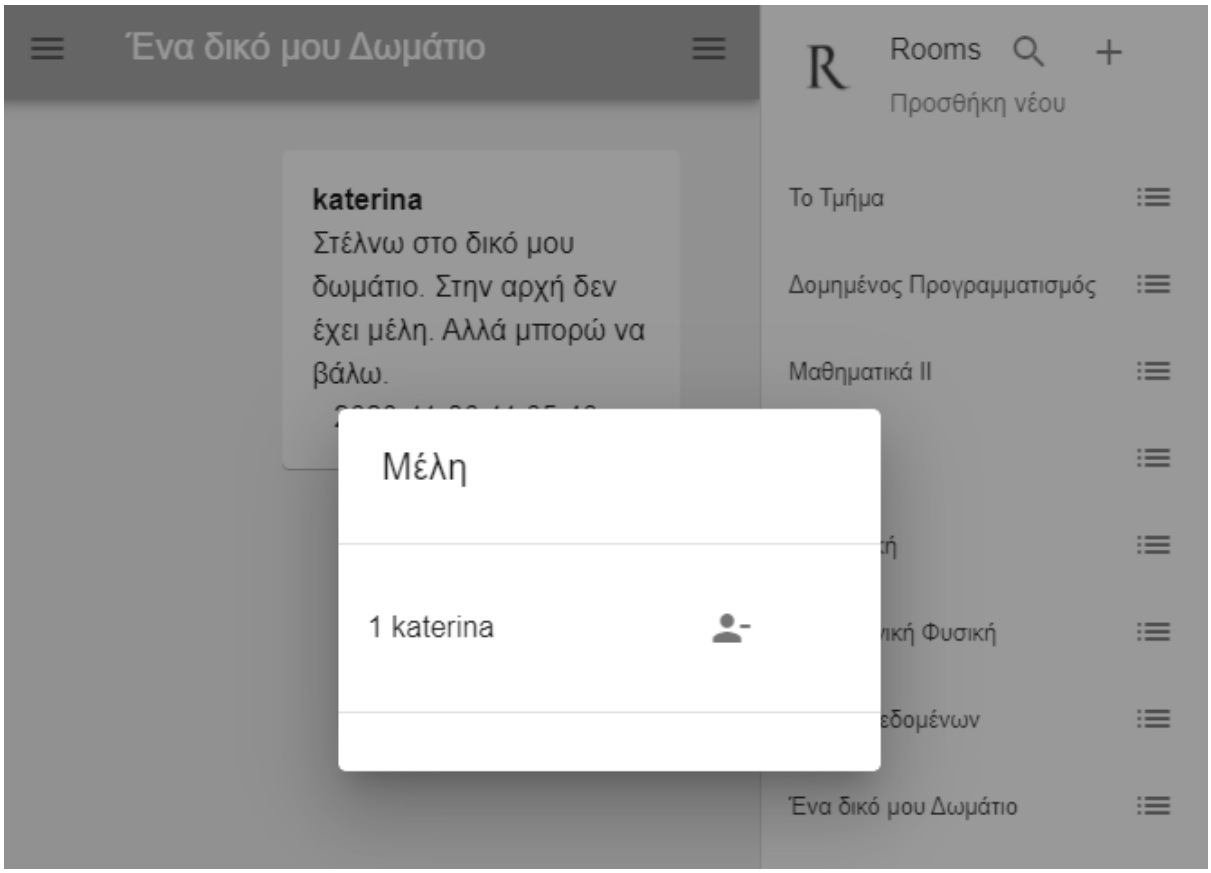
Σχήμα 3.28: Το νέο Δωμάτιο που δημιουργήθηκε

Στο Σχήμα 3.28 παρουσιάζεται το νέο Δωμάτιο που δημιουργήθηκε.

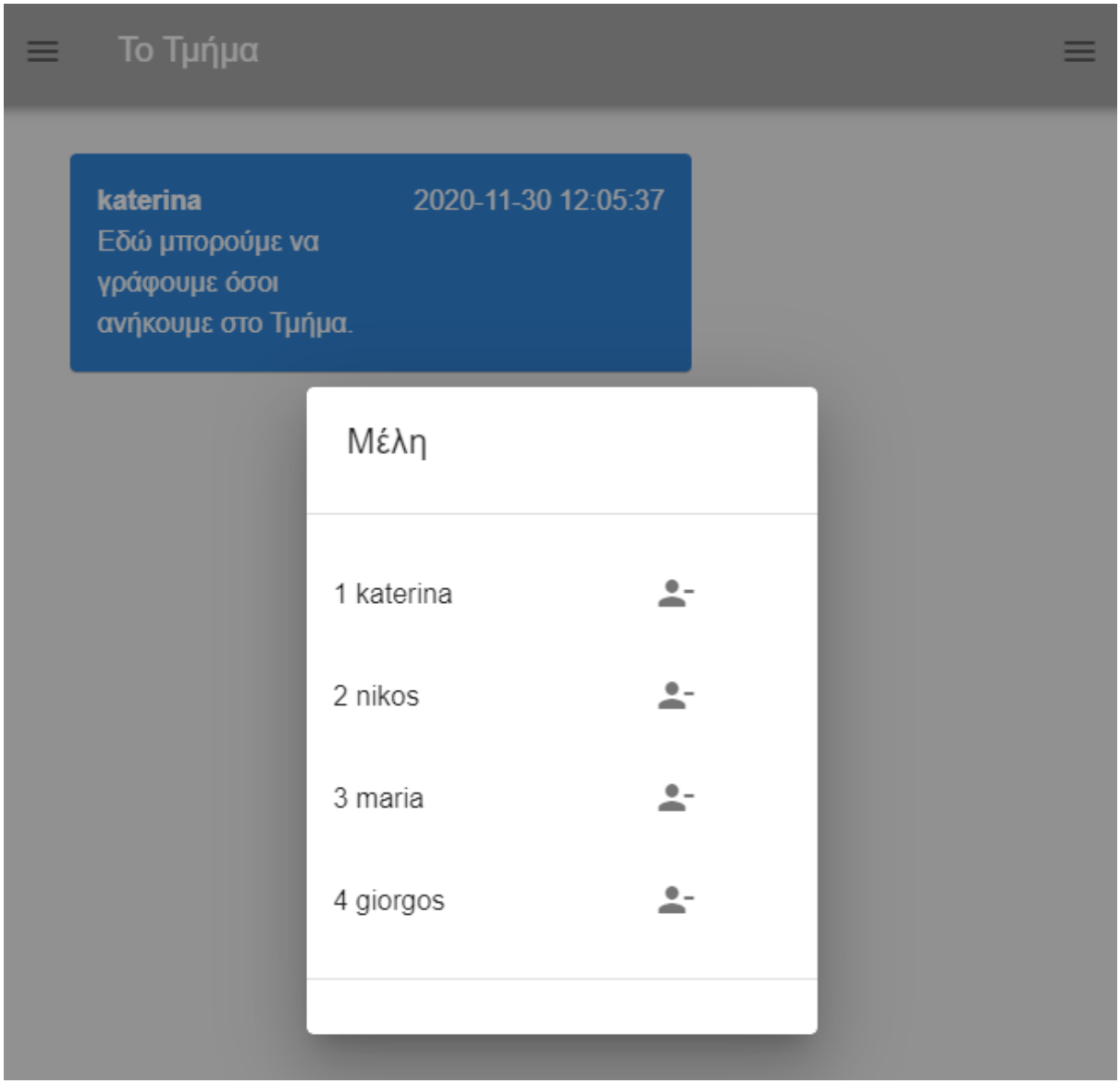


Σχήμα 3.29: Ο διάλογος για τα Μέλη του νέου Δωμάτιου που δημιουργήθηκε

Στο Σχήμα 29 παρουσιάζεται ο διάλογος για τα Μέλη του νέου Δωμάτιου που δημιουργήθηκε.

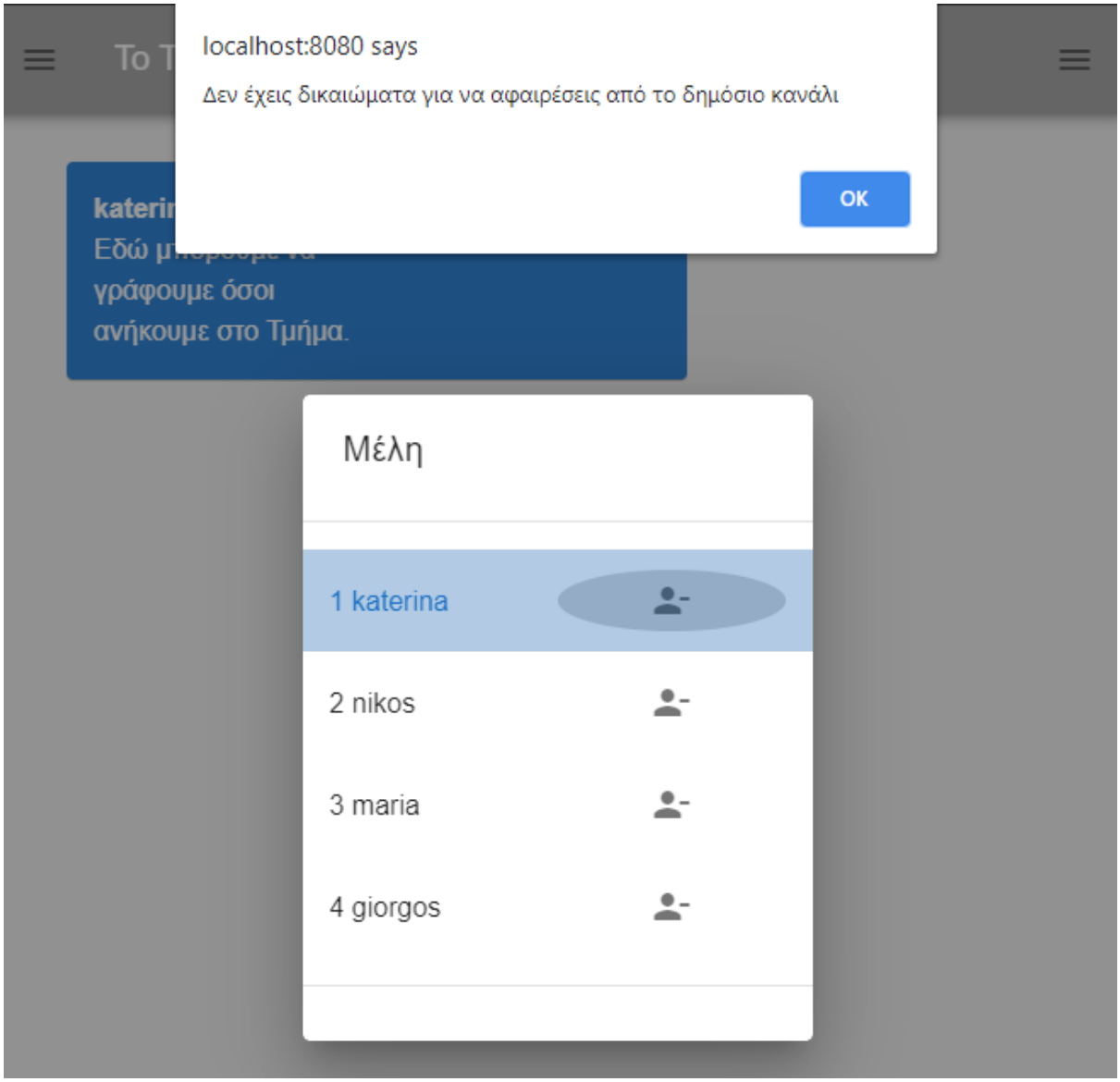


Σχήμα 3.30: Ο διάλογος για τα Μέλη του δωματίου



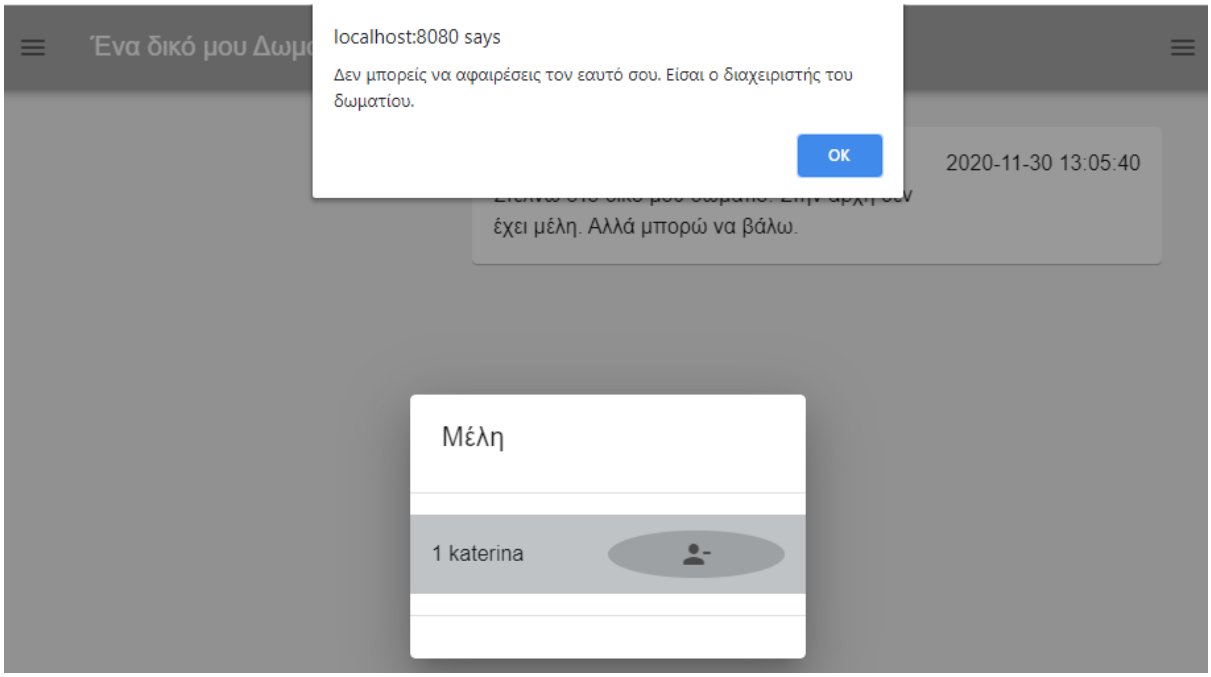
Σχήμα 3.31: Ο διάλογος για τα Μέλη του κοινού δωματίου

Στα Σχήματα 3.30 και 3.31 παρουσιάζεται ο διάλογος για τα Μέλη του προσωπικού και του κοινού δωματίου που περιέχει όλους τους χρήστες.



Σχήμα 3.32: Διάλογος της περίπτωσης που κάποιος θέλει να αφαιρέσει Μέλη του κοινού δωματίου Το Τμήμα μας

Στο Σχήμα 3.32 παρουσιάζεται ο διάλογος της περίπτωσης που κάποιος θέλει να αφαιρέσει Μέλη του κοινού δωματίου Το Τμήμα μας.



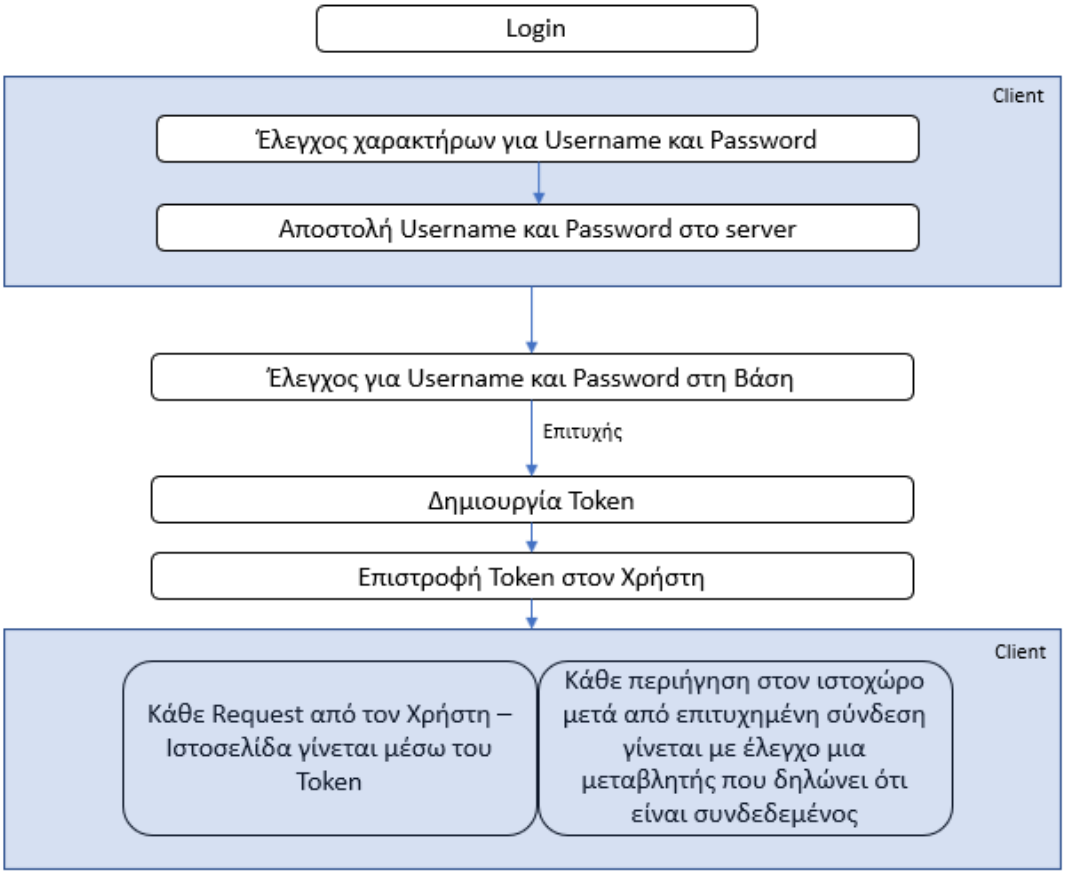
Σχήμα 3.33: Ο διάλογος της περίπτωσης που κάποιος θέλει να αφαιρέσει τον εαυτό του από κάποιο δωμάτιο στο οποίο είναι διαχειριστής

Στο Σχήμα 3.33 παρουσιάζεται ο διάλογος της περίπτωσης που κάποιος θέλει να αφαιρέσει τον εαυτό του από κάποιο δωμάτιο στο οποίο είναι διαχειριστής.

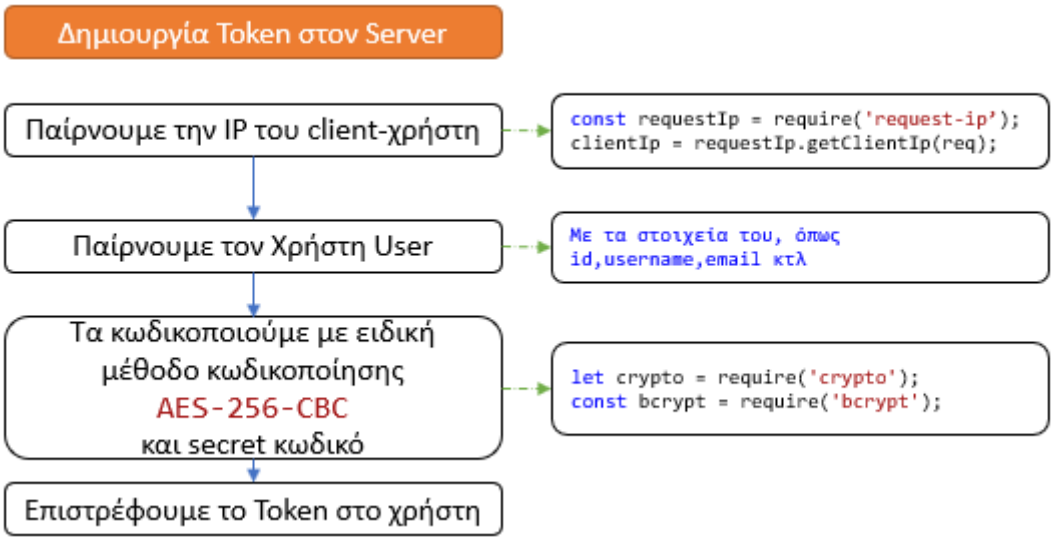
3.3 Είσοδος στο σύστημα – Ταυτοποίηση και Εξουσιοδότηση

Για την είσοδο του χρήστη στο σύστημα ο χρήστης πρέπει να δώσει τα σωστά στοιχεία (email-password). Τότε καλείται στο API του server μια ειδική συνάρτηση η οποία σε περίπτωση επιτυχίας κατασκευάζει ένα token και το επιστρέφει στο front website. Ο χρήστης στη συνέχεια μπορεί να καλεί οποιαδήποτε συνάρτηση του API server με το token και δεν χρειάζεται συνέχεια να δίνει email και password συνέχεια.

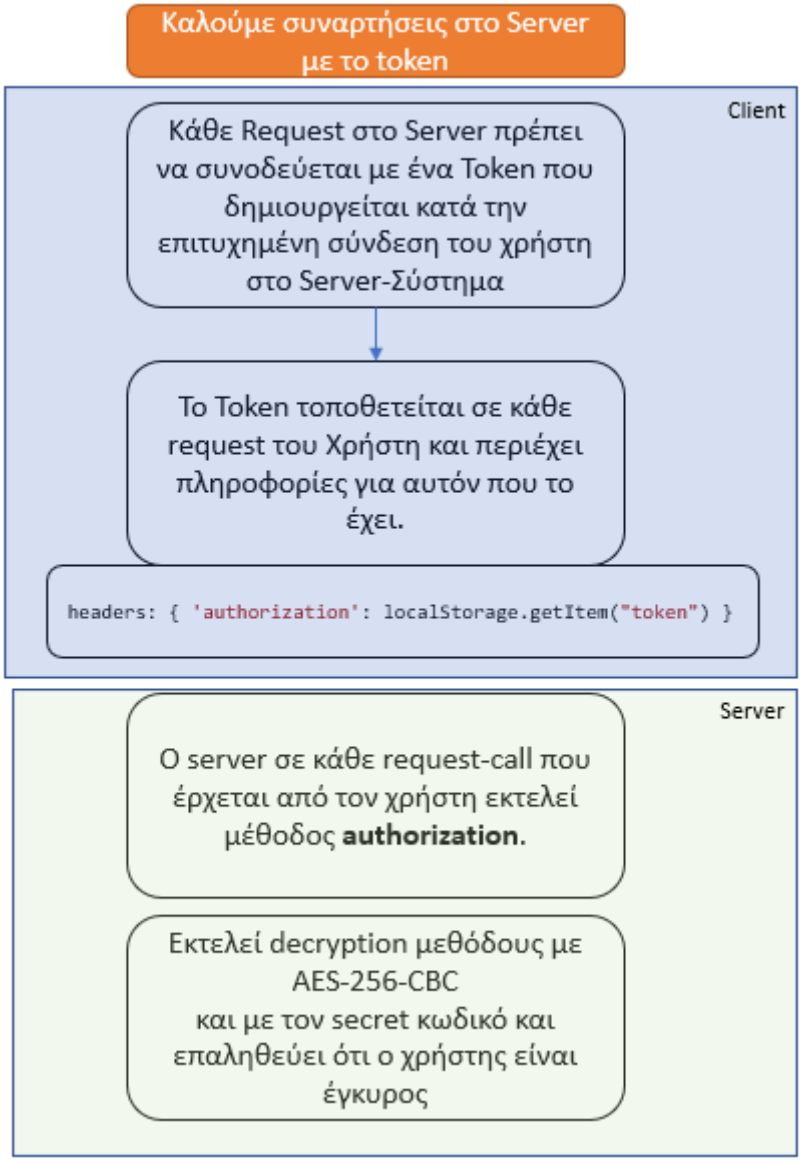
Η συνολική διαδικασία περιγράφεται στα παρακάτω Σχήματα 3.34, 3.35 και 3.36



Σχήμα 3.34: Διαδικασία σύνδεσης του χρήστη και δημιουργίας token



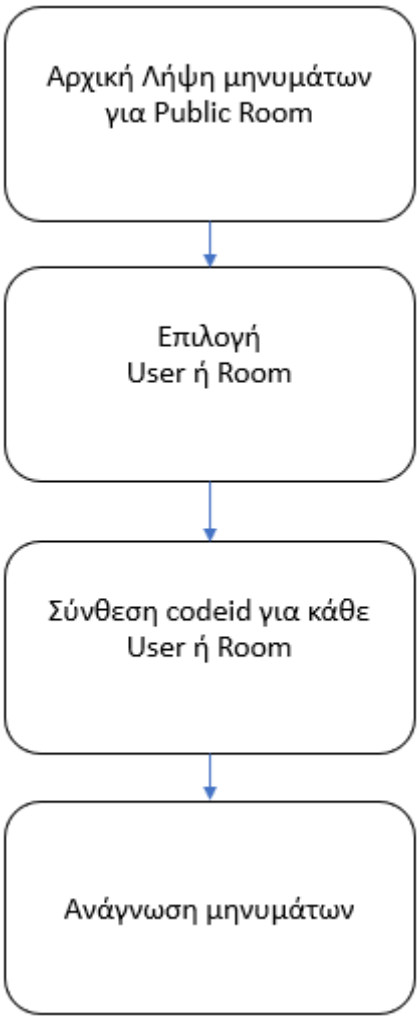
Σχήμα 3.35: Διαδικασία δημιουργίας token



Σχήμα 3.36: Διαδικασία τρόπου με τον οποίο καλείται μια συνάρτηση του API-server από την σελίδα website-front-end με χρήση token

3.4 Διαδικασίες

Στο Σχήμα 3.37 παρουσιάζεται η διαδικασία που εκτελείται όταν μπαίνει στη ιστοσελίδα ο χρήστης για πρώτη φορά.



Σχήμα 3.37: Διαδικασία που εκτελείται όταν μπαίνει στη ιστοσελίδα ο χρήστης για πρώτη φορά

Για την επικοινωνία του χρήστη με Sockets με το API server θα πρέπει γνωρίζει το όνομα του καναλιού-sockets που θα μιλήσει ή θα συμμετάσχει, όπως περιγράφεται στα σχήματα 3.38 και 3.39 και το ονομάζουμε **codeid**. Δηλαδή, όταν ένας χρήστης επιλέγει έναν άλλο χρήστη τότε δημιουργείται κανάλι με το ίδιο όνομα ανάμεσα στους δύο τους. Το ίδιο ισχύει για τα Rooms όπου κάθε room έχει μοναδικό όνομα και μπορούμε να μιλήσουμε ή να δεχτούμε μηνύματα από αυτό.

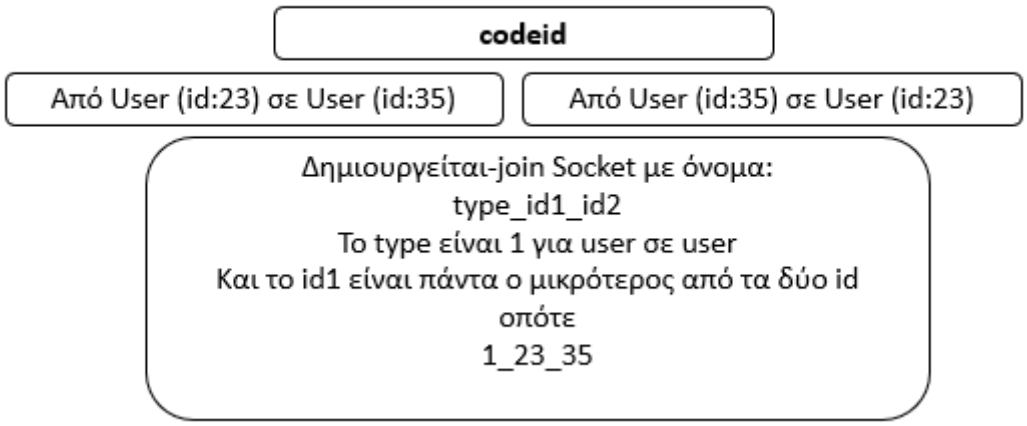
```
let request_query = "?message_type=" + request_data.message_type + "&sender_id=" + request_data.send
er_id + "&receiver_id=" + request_data.receiver_id;


Vue.http.get(state.settings.url.api + '/message/' + request_query, { headers: { 'authorization': localStorage.g
etItem("token") } }).then(resp => { …
```

Στο παραπάνω κώδικα φαίνεται ο τρόπος με τον οποίο χρησιμοποιείται η βιβλιοθήκη Vue Resource για να καλέσουμε μια συνάρτηση με GET request για να λάβουμε τα μηνύματα που αφορούν ένα συγκεκριμένο χρήστη ή room. Να επισημανθεί ότι με κάθε request βάζουμε και το token που μας έχει επιστρέψει ο server όταν συνδεθήκαμε.

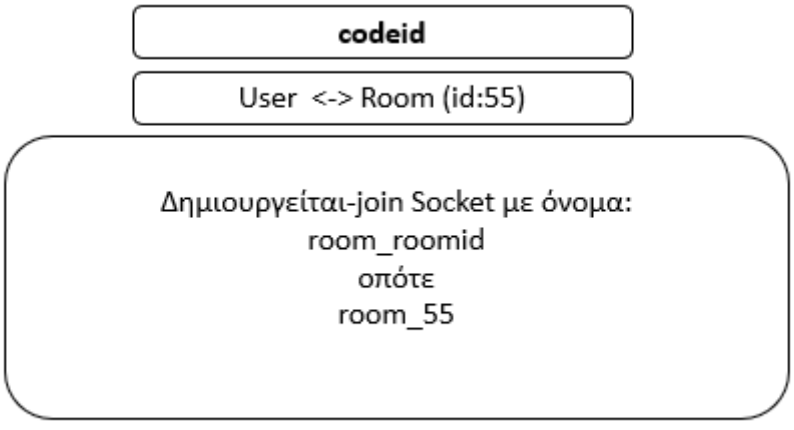
```
  selectUser(alias, id) {
    this.toMessage = alias;
    console.log("id= ", id);

    localStorage.setItem("toMessage", alias);
    localStorage.setItem("idtosend", id);
    localStorage.setItem("mtype", 1);
    let request_data = {
      message_type: parseInt(1),
      sender_id: parseInt(this.user.id),
      receiver_id: parseInt(id),
    };
    this.getMessages(request_data);
  },



  selectRoom(room_name, id) {
    this.toMessage = room_name;
    console.log("id= ", id);

    localStorage.setItem("toMessage", room_name);
    localStorage.setItem("idtosend", id);
    localStorage.setItem("mtype", 2);
    let request_data = {
```

```
    message_type: parseInt(2),
    sender_id: parseInt(this.user.id),
    receiver_id: parseInt(id),
   };
   this.getMessages(request_data);
 },
```

Στον παραπάνω κώδικα φαίνεται ότι όταν επιλέγουμε έναν χρήστη ή δωμάτιο αποθηκεύουμε τα χαρακτηριστικά του για να κατασκευάσουμε το codeid και αν θέλουμε να στείλουμε μήνυμα το χρησιμοποιούμε.

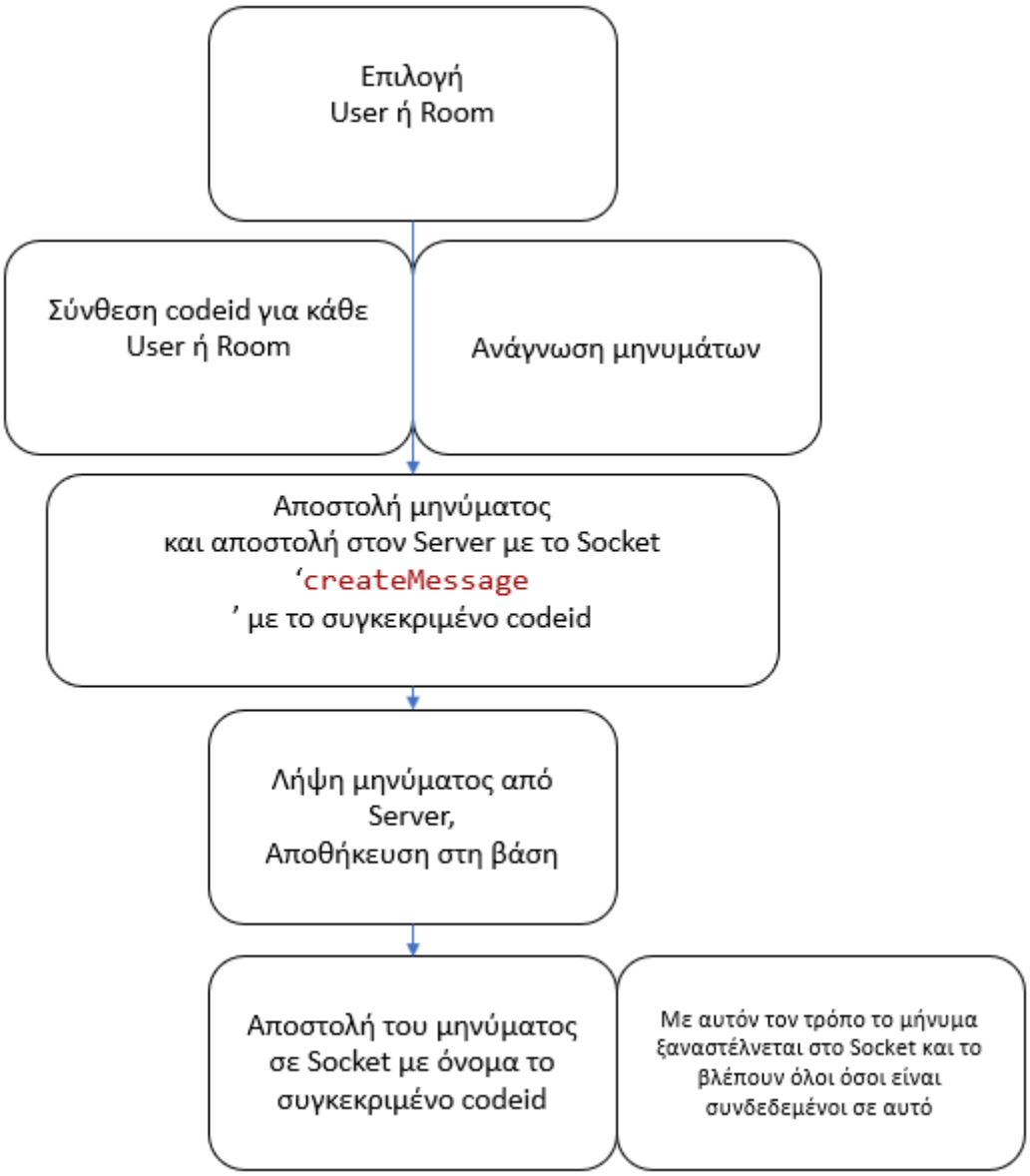


Σχήμα 3.38: Μέθοδος απόδοσης codeid σε socket για επικοινωνία μεταξύ δύο χρηστών-users

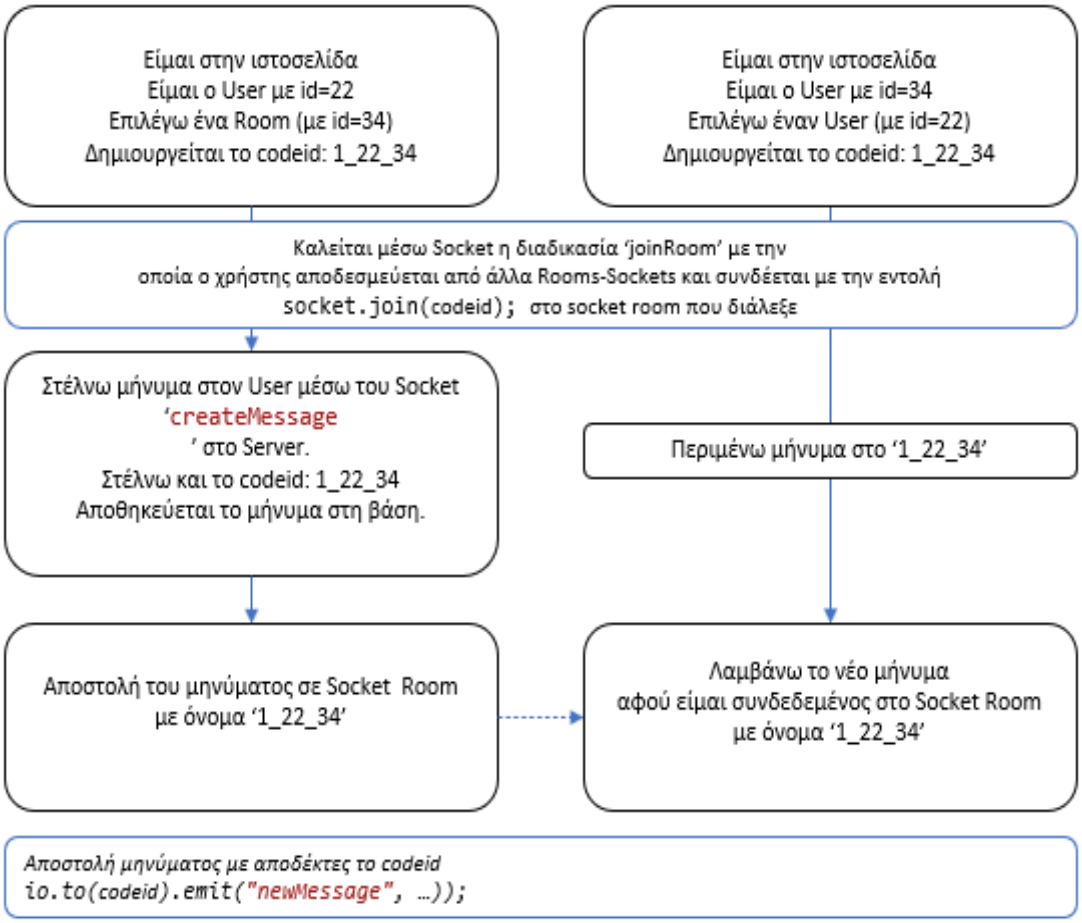


Σχήμα 3.39: Μέθοδος απόδοσης codeid σε socket για ένα δωμάτιο-room

Στο Σχήμα 3.40 παρουσιάζεται ο τρόπος με τον οποίο στέλνει ο χρήστης ένα μήνυμα σε έναν άλλο χρήστη ή room και πως ενημερώνονται τα υπόλοιπα μέλη μέσω των sockets.

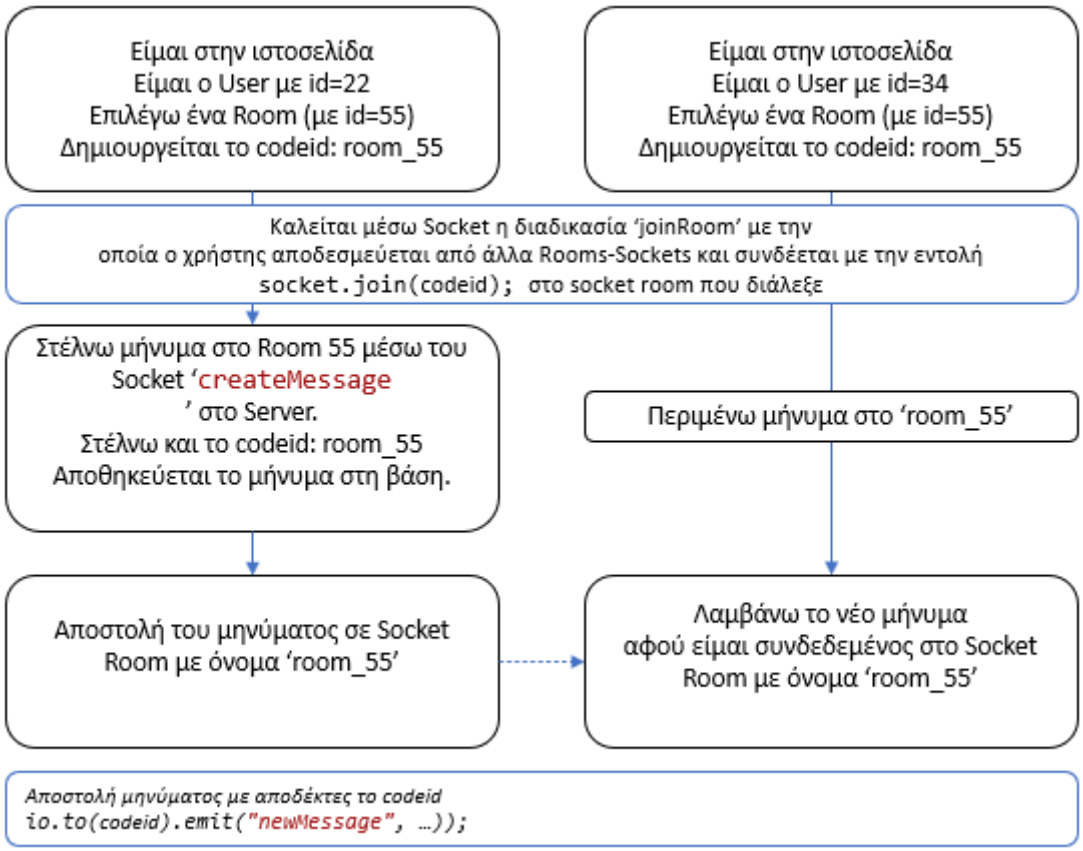


Σχήμα 3.40: Τρόπος με τον οποίο στέλνει ο χρήστης ένα μήνυμα σε έναν άλλο χρήστη ή room και πως ενημερώνονται τα υπόλοιπα μέλη μέσω των sockets



Σχήμα 3.41: Ανάλυση τρόπου επικοινωνίας χρήστη με χρήστη

Στα Σχήματα 3.41 και 3.42 αναλύεται περισσότερο ο τρόπος επικοινωνίας χρήστη με χρήστη και χρήστη με room.



Σχήμα 3.42: Ανάλυση τρόπου επικοινωνίας χρήστη με room

```
let codeid = null;
if(request_data.message_type == 1)
{
codeid  = request_data.message_type + "_";
if (request_data.sender_id < request_data.receiver_id) {
codeid  += request_data.sender_id + "_" + request_data.receiver_id;
}
else {
codeid  += request_data.receiver_id + "_" + request_data.sender_id;
}
}
if(request_data.message_type == 2)
{
codeid  = "room"+request_data.receiver_id;
}
```

Στον παραπάνω κώδικα φαίνεται ο τρόπος απόδοσης του codeid.

```
send() {
  if (this.$refs.form.validate()) {
    let idtosend = localStorage.getItem("idtosend");
    let mtype = localStorage.getItem("mtype");

    idtosend = typeof idtosend !== "undefined" ? parseInt(idtosend) : 0;
    mtype = typeof mtype !== "undefined" ? parseInt(mtype) : 0;

    let room = localStorage.getItem("room");

    const payload = {
      alias: this.user.alias,
      message_txt: this.text,
      sender_id: this.user.id,
      receiver_id: idtosend,
      message_type: mtype,
      room: room,
    };
    this.createMessage(payload);
```

Στον παραπάνω κώδικα φαίνεται ο τρόπος αποστολής ενός μηνύματος σε έναν χρήστη ή room σύμφωνα με αυτά που αναφέρθηκαν στα παραπάνω διαγράμματα.

Στο Σχήμα 3.43 παρουσιάζονται και οι υπόλοιπες λειτουργίες που έχει στη διάθεση του ο χρήστης. Στο τελευταίο κεφάλαιο παρουσιάζονται και οι βελτιώσεις και οι προσθήκες που θα μπορούσαν να εισαχθούν.

Επιπρόσθετες Λειτουργίες	Δημιουργία Room από κάποιον Χρήστη
	Προσθήκη Μελών σε Room που ανήκει στο Χρήστη
	Αφαίρεση Μελών σε Room που ανήκει στο Χρήστη
	Αφαίρεση του εαυτού μου από κάποιο Room
	Αναζήτηση χρηστών και rooms

Σχήμα 3.43: Οι υπόλοιπες λειτουργίες που έχει στη διάθεση του ο χρήστης

3.5 Διαδικασίες στην πλευρά του Server

1. Πρέπει να συμπεριληφθούν οι σημαντικές βιβλιοθήκες με πρώτη την express που εξυπηρετεί σε δημιουργία server με ευκολίες στη χρήση.

```
const express = require('express');
const app = express();
```

2. Συμπερίληψη βιβλιοθηκών που χρειάζονται για διευθέτηση των cookies, των παραμέτρων του url αλλά και των παραμέτρων από τις φόρμες μέσω των post requests.

```
const cookieParser = require('cookie-parser');
const bodyParser = require('body-parser');
var cors = require('cors');
app.use(cookieParser());
app.use(bodyParser.json());
app.use(bodyParser.urlencoded({
  extended: true,
}));
app.use(cors())
```

3.

Δημιουργία API web server στην port 3001 για χρήση σε τοπικό δίκτυο.

```
const server = app.listen(3001, function () {
  console.log('server running on port 3001');
});
```

Δημιουργία socket.io server.

```
const io = require('socket.io')(server);
```

4. Όλη η διαχείριση των Sockets πρέπει να γίνεται μέσα στη μέθοδο που ορίζεται από το io.on

```
io.on('connection', function (socket) {
```

```
…
```

});

5. Με το παρακάτω κομμάτι κώδικα ο server ’ακούει’ σε κάποιον client-χρήστη που του στέλνει μήνυμα στο "createMessage" socket.

Στη συνέχεια καλείται συνάρτηση που θα πάρει τα δεδομένα και θα χειριστεί το μήνυμα.

Στο τέλος το socket αυτό θα στείλει σε όλους που ‘ακούνε’ στο socket με ίδιο codeid και συγκεκριμένα στη μέθοδο-socket ‘newMessage’ του κάθε χρήστη.

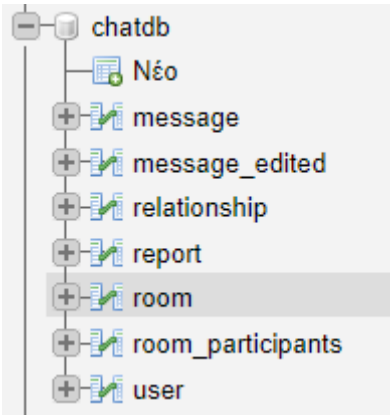
```
socket.on("createMessage", (payload) => {
userController.writeMessage(payload);
   io.to(payload.room).emit("newMessage", new Message(payload.alias, payload.message_txt, payload.sender_id));
  });
```

6. Στο επόμενο κομμάτι κώδικα παρουσιάζεται η συνάρτηση που χειρίζεται το νέο μήνυμα, όπως αναφέρθηκε στο προηγούμενο κομμάτι. Το πρώτο που εκτελείται πάντα είναι η αυθεντικοποίηση του χρήστη που καλεί το API και το συγκεκριμένο url. Στη συνέχεια καλείται η συνάρτηση με την οποία αποθηκεύεται το κείμενο στη βάση δεδομένων με επιπρόσθετες πληροφορίες που αφορούν τον αποστολέα, τον παραλήπτη και τον τύπο του μηνύματος.

```
   async writeMessage(payload) {
      //message_txt, sender_id,receiver_id,message_type
      try {
         // Authorize
         let user = auth.authtoken(req, res);
         if (typeof user === 'undefined' || user === null) {
        return helpers.error(res, 'AUTH_USER_AUTHORIZATION_INVALID', 400);
         }
         let [result] = await db.execute("INSERT INTO message (active, sender_id, receiver_id, message_type, message_text) VALUES (1," + payload.sender_id + " ," + payload.receiver_id + " ," + payload.message_type + ", '" + payload.message_txt + "')");
         return 1;
      } catch (error) {
```

```
        console.log("userController error=", error);
        return 0;
    }
}
```

3.6 Η Βάση



Σχήμα 3.44: Οι πίνακες της βάσης

Η βάση κατασκευάστηκε με MySQL 10.4.11-MariaDB [31]. Στο σχήμα 3.44 φαίνεται η σχηματικό με του πίνακες που χρησιμοποιήθηκαν.

Στη συνέχεια ακολουθούν Σχήματα με τους πίνακες και τα πεδία που περιλαμβάνει ο καθένας.

Οι πίνακες μεταξύ τους δεν είναι συνδεδεμένοι με foreign keys και όλοι διαθέτουν τα πεδία

Id που είναι το κεντρικό κλειδί και αύξων αριθμός,

Το active για να προσδιοριστεί αν είναι ενεργή η εγγραφή

Και τα πεδία created_at και updated_at που είναι για την αποθήκευση πότε δημιουργήθηκε και πότε ανανεώθηκε η εγγραφή στη βάση.

#	Όνομα	Τύπος	Σύνθεση	Χαρακτηριστικά	Κενό	Προεπιλογή
1	id	int(10)		UNSIGNED	Όχι	*Καμία*
2	active	tinyint(1)		UNSIGNED	Όχι	1
3	username	varchar(30)	utf8_general_ci		Όχι	*Καμία*
4	email	varchar(255)	utf8_general_ci		Όχι	*Καμία*
5	password	varchar(100)	utf8_general_ci		Όχι	*Καμία*
6	alias	varchar(30)	utf8_general_ci		Όχι	*Καμία*
7	name	varchar(255)	utf8_general_ci		Όχι	*Καμία*
8	firstname	varchar(100)	utf8_general_ci		Όχι	*Καμία*
9	lastname	varchar(100)	utf8_general_ci		Όχι	*Καμία*
10	parentname	varchar(100)	utf8_general_ci		Όχι	*Καμία*
11	kind	tinyint(4)		UNSIGNED	Όχι	*Καμία*
12	lang	enum('en', 'el')	utf8_general_ci		Όχι	el
13	theme	enum('light', 'dark')	utf8_general_ci		Όχι	light
14	created_at	timestamp			Ναι	current_timestamp()
15	updated_at	timestamp			Ναι	current_timestamp()

Σχήμα 3.45: Ο πίνακας user

#	Όνομα	Τύπος	Σύνθεση	Χαρακτηριστικά	Κενό	Προεπιλογή
1	id	int(11)			Όχι	*Καμία*
2	active	tinyint(1)		UNSIGNED	Όχι	1
3	sender_id	int(10)		UNSIGNED	Όχι	*Καμία*
4	receiver_id	int(10)		UNSIGNED	Όχι	*Καμία*
5	message_type	tinyint(1)		UNSIGNED	Όχι	*Καμία*
6	message_text	varchar(500)	utf8mb4_general_ci		Όχι	*Καμία*
7	is_edited	tinyint(1)		UNSIGNED	Όχι	0
8	is_read	tinyint(1)		UNSIGNED	Όχι	0
9	created_at	timestamp			Ναι	current_timestamp()
10	updated_at	timestamp			Ναι	current_timestamp()

Σχήμα 3.46: Ο πίνακας message

id	active	sender_id	receiver_id	message_type	message_text	is_edited	is_read
1	1	1	4	1	Γειά σου Γιώργο	0	0
2	1	4	1	1	Γειά σου Κατερίνα	0	0
3	1	1	4	1	Τι κάνεις?	0	0
4	1	4	1	1	Όλα καλά	0	0
5	1	1	4	1	Ωραία	0	0
6	1	1	18	2	Μια ερώτηση για το μάθημα	0	0
7	1	4	18	2	Και εγώ έχω μια	0	0
8	1	1	1	2	Εδώ μπορούμε να γράφουμε όσοι ανήκουμε στο Τμήμα.	0	0
9	1	4	4	1	Στέλνω στον εαυτό μου	0	0
10	1	1	22	2	Στέλνω στο δικό μου δωμάτιο. Στην αρχή δεν έχει μέ...	0	0

Σχήμα 3.47: Ο πίνακας message συμπληρωμένος

#	Όνομα	Τύπος	Σύνθεση	Χαρακτηριστικά	Κενό	Προεπιλογή
1	**id**	int(10)		UNSIGNED	Όχι	*Καμία*
2	**active**	tinyint(1)		UNSIGNED	Όχι	1
3	**user_id1**	int(10)		UNSIGNED	Όχι	*Καμία*
4	**user_id2**	int(10)		UNSIGNED	Όχι	*Καμία*
5	**rel_type**	tinyint(3)		UNSIGNED	Όχι	1
6	**created_at**	timestamp			Ναι	*NULL*
7	**updated_at**	timestamp			Ναι	*NULL*

Σχήμα 3.48: Ο πίνακας relationship

id	active	user_id1	user_id2	rel_type	created_at
1	1	1	2	1	2020-11-16
2	1	1	3	1	2020-11-16
3	1	1	4	1	2020-11-16
4	1	2	1	2	2020-11-16

Σχήμα 3.49: Ο πίνακας relationship συμπληρωμένος

#	Όνομα	Τύπος	Σύνθεση	Χαρακτηριστικά	Κενό	Προεπιλογή
1	id	int(10)		UNSIGNED	Όχι	*Καμία*
2	active	tinyint(1)		UNSIGNED	Όχι	1
3	room_admin_id	int(10)		UNSIGNED	Όχι	*Καμία*
4	room_name	varchar(50)	utf8mb4_general_ci		Όχι	*Καμία*
5	room_type	tinyint(1)		UNSIGNED	Όχι	1
6	room_sessionid	varchar(100)	utf8mb4_general_ci		Ναι	*NULL*
7	allow_others_toaddusers	tinyint(1)		UNSIGNED	Όχι	2
8	created_at	timestamp			Ναι	current_timestamp()
9	updated_at	timestamp			Ναι	current_timestamp()

Σχήμα 3.50: Ο πίνακας room

id	active	room_admin_id	room_name	room_type	room_sessionid	allow_others_toaddusers
1	1	1	Το Τμήμα	112	0	0
2	1	1	Δομημένος Προγραμματισμός	1	0	0
3	1	1	Μαθηματικά ΙΙ	1	0	0
4	1	2	Ψηφιακά Συστήματα	1	0	0
5	1	3	Ηλεκτρονικά Κυκλώματα	1	0	0
18	1	1	Δίκτυα	1	1	2
19	1	1	Ρομποτική	1	1	2
20	1	1	Ηλεκτρονική Φυσική	1	1	2
21	1	1	Βάσεις Δεδομένων	1	1	2
22	1	1	Ένα δικό μου Δωμάτιο	1	1	2

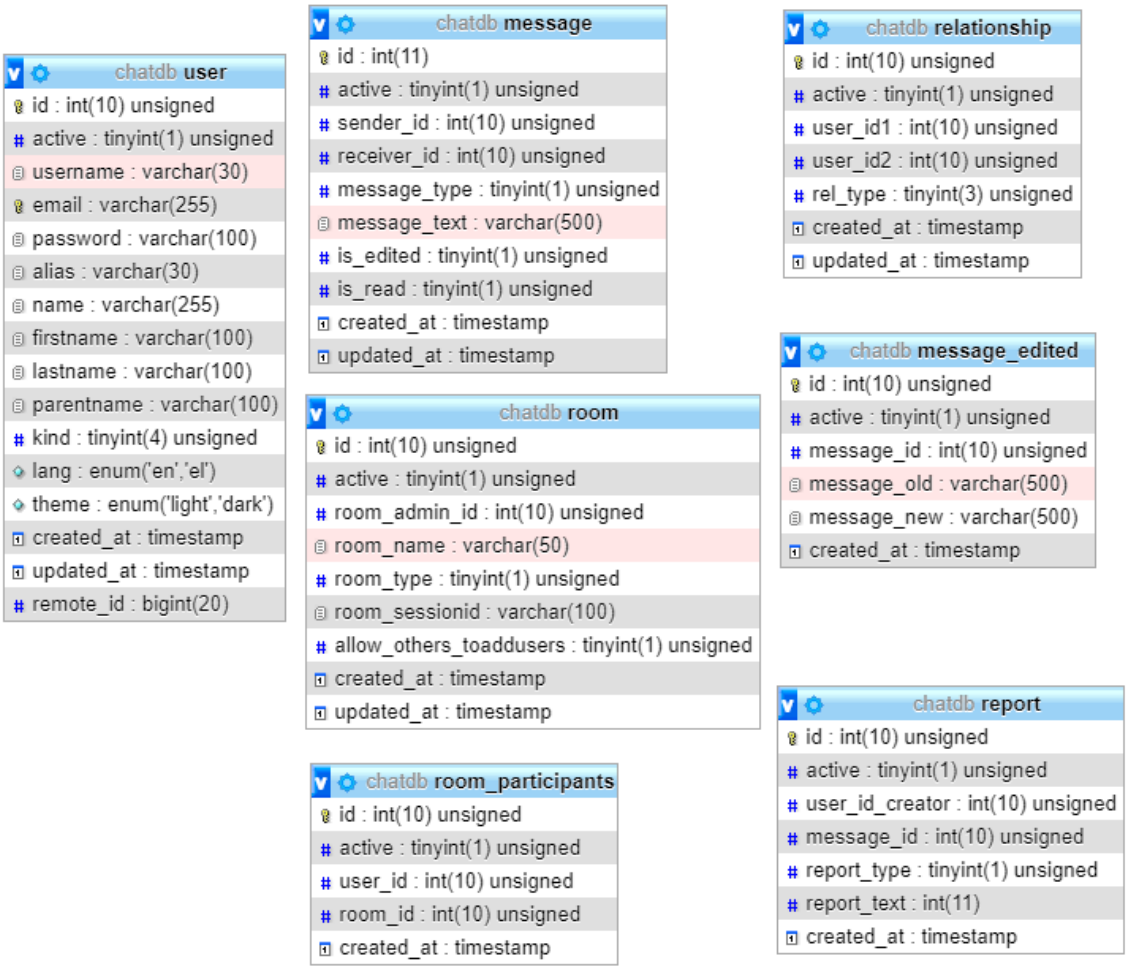
Σχήμα 3.51: Ο πίνακας room συμπληρωμένος

#	Όνομα	Τύπος	Σύνθεση	Χαρακτηριστικά	Κενό	Προεπιλογή
1	id	int(10)		UNSIGNED	Όχι	*Καμία*
2	active	tinyint(1)		UNSIGNED	Όχι	1
3	user_id	int(10)		UNSIGNED	Όχι	*Καμία*
4	room_id	int(10)		UNSIGNED	Όχι	*Καμία*
5	created_at	timestamp			Ναι	current_timestamp()

Σχήμα 3.52: Ο πίνακας room_participants

id	active	user_id	room_id	created_at
1	1	1	1	2020-11-15 12:12:33
2	1	2	1	2020-11-15 12:18:30
3	1	3	1	2020-11-15 12:18:30
4	1	4	1	2020-11-15 12:18:30
5	1	1	2	2020-11-15 12:20:52
6	1	1	3	2020-11-15 12:20:52
7	1	2	4	2020-11-15 12:21:25
8	1	3	5	2020-11-15 12:21:25
15	1	1	18	2020-11-23 12:43:56
16	1	1	19	2020-11-23 12:44:18
17	1	1	20	2020-11-23 12:45:57
18	1	1	21	2020-11-23 12:48:09
23	1	1	22	2020-11-30 13:52:37

Σχήμα 3.53: Ο πίνακας room_participants συμπληρωμένος



Σχήμα 3.54: Η βάση με όλους τους πίνακες της

3.7 Ανάλυση ασφάλειας στο σύστημα και στα δεδομένα

Ξεχωρίζουμε τριών ειδών ασφάλεια στο σύστημα.

Ασφάλεια με κωδικούς πρόσβασης στην ιστοσελίδα.

Ο χρήστης μπορεί να συνδεθεί με την κεντρική σελίδα του front page μόνο αν εισάγει σωστά τα στοιχεία του, email και password.

Ασφάλεια με token για την αυθεντικοποίηση στην επικοινωνία του χρήστη με τον server

Αφού ο χρήστης συνδεθεί με επιτυχία ο server API του επιστρέφει ένα μοναδικό και ατομικό token [32]. Σε αυτό περιέχονται και προσωπικές πληροφορίες.

Με αυτό το token ο χρήστης μπορεί να καλεί συναρτήσεις του API για την εξυπηρέτησης του, όπως get messages, create message, get rooms κτλ. Με αυτό τον τρόπο δεν χρειάζεται ο χρήστης να στέλνει κάθε φορά το email,password του για να πιστοποιήσει ότι έχει δικαίωμα να καλέσει και εκτελέσει κάποιες διεργασίες του API server. Επίσης, μέσω του token ο server ξέρει ποιος κάλεσε το API αλλά και να έχει δικαίωμα σε κάποιες συναρτήσεις ή μεθόδους.

Ασφάλεια αποστολής δεδομένων με SSL

Η επικοινωνία του front end website και του back end API server πραγματοποιείται μέσω SSL ασφαλούς επικοινωνίας για να παρέχεται ασφάλεια κατά την μετάδοση ευαίσθητων δεδομένων στο διαδίκτυο. Το SSL [33] χρησιμοποιεί μεθόδους κρυπτογράφησης των δεδομένων που ανταλλάσσονται μεταξύ δύο συσκευών εγκαθιδρύοντας μία ασφαλή σύνδεση μεταξύ τους μέσω του διαδικτύου.

Κεφάλαιο 4ο: Συμπεράσματα και προτάσεις βελτίωσης

Παρουσιάστηκε και αναλύθηκε μια εφαρμογή γραπτής επικοινωνίας μέσω διαδικτύου που υλοποιήθηκε με node.js, web sockets, vue.js και βάση δεδομένων mySql. Εστίασε στη σύγχρονη γραπτή επικοινωνία εξ' αποστάσεως μεταξύ δύο χρηστών ή περισσότερων όταν επικοινωνούν μέσω ενός δωματίου. Σε ένα τμήμα ενός ιδρύματος ή μιας υπηρεσίας μπορεί να παρέχει υπάρχει ένα κοινό δωμάτιο που θα μπορούν να επικοινωνήσουν όλοι οι χρήστες μεταξύ τους που είναι εγγεγραμμένοι και συνδεδεμένοι στο σύστημα-εφαρμογή. Ο κάθε χρήστης μπορεί να δημιουργήσει δωμάτιο και να συμπεριλάβει μέλη-χρήστες σε αυτό για να επικοινωνούν μεταξύ τους μόνοι όσοι ανήκουν σε αυτό.

Η εφαρμογή που δημιουργήθηκε είναι ένα website και ο χρήστης μπορεί να τη χρησιμοποιήσει μέσω ενός περιηγητή. Χρησιμοποιήθηκε vue.js για το front-end, που αφορά το επίπεδο παρουσίασης και προβολής των γραφικών στοιχείων, για λόγους εκμάθησης της τεχνολογίας αλλά και για πιο γρήγορο άμεση ανανέωση γραφικών στοιχείων και εύκολη προσαρμογή του socket.io. Επιλέχθηκε node.js server για την εξυπηρέτηση του front-end και των request ενέργειων ιδίως όσο αφορά τη διαχείριση της βάσης.

Οι χρήστες που ανήκουν σε ένα εργασιακό χώρο να μπορούν να συνομιλήσουν όλοι μαζί σε ένα κοινό κανάλι-δωμάτιο. Οι χρήστες να μπορούν να δημιουργήσουν κανάλια-δωμάτια και να συμπεριλάβουν άλλους χρήστες σε αυτά και να συνομιλήσουν όλοι μαζί σε πραγματικό χρόνο. Οι χρήστες μπορούν να διαγράψουν τον εαυτό τους από κάποιο δωμάτιο. Ο χρήστης που χρησιμοποιεί την ιστοσελίδα πρόσβασης στο σύστημα είναι εξουσιοδοτημένος.

Η ασφάλεια του συστήματος περιεγράφηκε στο προηγούμενο κεφάλαιο και μπορούμε να συμπεράνουμε ότι η χρήση token σε κάθε απαίτηση-request του διασφαλίζει την αυθεντικοποίηση του χρήστη χωρίς να απαιτείται συνεχώς η εισαγωγή email-password.

Όσον αφορά τις βελτιώσεις η πιο σημαντική είναι η διαχείριση των χρηστών. Η λίστα των χρηστών πρέπει να είναι συγκεκριμένη ή να συνδέεται με τη βάση του εκάστοτε τμήματος ή υπηρεσίας. Πρέπει να μπορούν διευκρινιστούν οι σχέσεις των χρηστών με ένα είδος φιλίας. Επίσης, πρέπει να υπάρχει ένα είδος απαγόρευσης-block και αναφοράς-report για κάποιον χρήστη ή από κάποιο δωμάτιο ώστε να αποφευχθεί η ενοχλητική δραστηριότητα από χρήστες. Επίσης, πρέπει να υλοποιηθεί μια πιο εξεζητημένη αναζήτηση χρηστών ή δωματίων από κάποιον χρήστη με κριτήρια φιλίας ή δημοτικότητας. Επιπρόσθετα, πρέπει να ενεργοποιηθεί η επισήμανση του πλήθους των μηνυμάτων που δεν έχει αναγνώσει ο χρήστης από κάποιο δωμάτιο ή χρήστη. Ακόμη, να ενεργοποιηθεί ειδική σήμανση που δείχνει πότε ο χρήστης είναι ενεργός. Επίσης, πρέπει να τοποθετηθούν ειδικές επιλογές σε κάθε μήνυμα που να μπορεί κάποιος χρήστης να αναφέρει (report) το περιεχόμενο κάποιου μηνύματος και να μπορεί να κάνει edit σε κάποιο μήνυμα. Έχουν δημιουργηθεί πίνακες για αυτές τις δύο λειτουργίες το message_edited και το report. Στον πρώτο εισάγονται τα μηνύματα που έχουν επεξεργαστεί από τον

ιδιοκτήτη-χρήστη και τα παλαιότερα πρέπει να κρατιούνται σε έναν άλλον πίνακα για λόγους ασφαλείας.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] https://nodejs.org/en/

[2] https://www.html5rocks.com/en/tutorials/websockets/basics/

[3] https://vuejs.org/

[4] https://www.mysql.com/

[5] https://github.com/cbh6/vuejs-socketio-chat

[6] https://www.w3schools.com/js/js_htmldom.asp

[7] https://github.com/Kaperskyguru/ChatAppWithVue_Socket.io

[8] https://github.com/psborul/nuxt-chat-app

[9] https://expressjs.com/

[10] https://nuxtjs.org/

[11] https://www.npmjs.com/package/vue-socket.io

[12] https://socket.io/

[13] https://vuetifyjs.com/en/

[14] https://v8.dev/

[15] https://en.wikipedia.org/wiki/BSD_licenses

[16] https://www.npmjs.com/

[17] https://en.wikipedia.org/wiki/API

[18] https://nodejs.org/en/download/]

[19] https://www.tutorialspoint.com/nodejs/nodejs_express_framework.htm

[20] https://angularjs.org/

[21] https://www.w3schools.com/react/react_jsx.asp

[22] https://animate.style/

[23] http://velocityjs.org/

[24] https://en.wikipedia.org/wiki/Single-page_application

[25] https://router.vuejs.org/

[26] https://vuex.vuejs.org/

[27] https://en.wikipedia.org/wiki/Yammer

[28] https://www.zendesk.com/

[29] https://en.wikipedia.org/wiki/WebSocket

[30] https://getbootstrap.com/

[31] https://mariadb.org/

[32] https://en.wikipedia.org/wiki/JSON_Web_Token

[33] https://en.wikipedia.org/wiki/Transport_Layer_Security