Τμήμα Πληροφορικής & Τηλεπικοινωνιών – Πανεπιστήμιο Θεσσαλίας

Πτυχιακή εργασία με τίτλο:

Διαδικτυακή εφαρμογή διαχείρισης παραγγελιών επιχειρήσεων μαζικής εστίασης

Ονοματεπώνυμο Φοιτητή: Τζίμης Αναστάσιος-Διονύσιος

ΑΜ: 2113161

Επιβλέπων Καθηγητής: Δαδαλιάρης Αντώνιος

ΝΟΜΙΚΑ

«Με ατομική μου ευθύνη και γνωρίζοντας τις κυρώσεις (1), που προβλέπονται από της διατάξεις της παρ. 6 του άρθρου 22 του Ν. 1599/1986, δηλώνω ότι:

1. Δεν παραθέτω κομμάτια βιβλίων ή άρθρων ή εργασιών άλλων αυτολεξεί χωρίς να τα περικλείω σε εισαγωγικά και χωρίς να αναφέρω το συγγραφέα, τη χρονολογία, τη σελίδα. Η αυτολεξεί παράθεση χωρίς εισαγωγικά χωρίς αναφορά στην πηγή, είναι λογοκλοπή. Πέραν της αυτολεξεί παράθεσης, λογοκλοπή θεωρείται και η παράφραση εδαφίων από έργα άλλων, συμπεριλαμβανομένων και έργων συμφοιτητών μου, καθώς και η παράθεση στοιχείων που άλλοι συνέλεξαν ή επεξεργάσθηκαν, χωρίς αναφορά στην πηγή. Αναφέρω πάντοτε με πληρότητα την πηγή κάτω από τον πίνακα ή σχέδιο, όπως στα παραθέματα.

2. Δέχομαι ότι η αυτολεξεί παράθεση χωρίς εισαγωγικά, ακόμα κι αν συνοδεύεται από αναφορά στην πηγή σε κάποιο άλλο σημείο του κειμένου ή στο τέλος του, είναι αντιγραφή. Η αναφορά στην πηγή στο τέλος π.χ. μιας παραγράφου ή μιας σελίδας, δεν δικαιολογεί συρραφή εδαφίων έργου άλλου συγγραφέα, έστω και παραφρασμένων, και παρουσίασή τους ως δική μου εργασία.

3. Δέχομαι ότι υπάρχει επίσης περιορισμός στο μέγεθος και στη συχνότητα των παραθεμάτων που μπορώ να εντάξω στην εργασία μου εντός εισαγωγικών. Κάθε μεγάλο παράθεμα (π.χ. σε πίνακα ή πλαίσιο, κλπ.), προϋποθέτει ειδικές ρυθμίσεις, και όταν δημοσιεύεται προϋποθέτει την άδεια του συγγραφέα ή του εκδότη. Το ίδιο και οι πίνακες και τα σχέδια

4. Δέχομαι όλες τις συνέπειες σε περίπτωση λογοκλοπής ή αντιγραφής.

(1) «Όποιος εν γνώσει του δηλώνει ψευδή γεγονότα ή αρνείται ή αποκρύπτει τα αληθινά με έγγραφη υπεύθυνη δήλωση του άρθρου 8 παρ. 4 Ν. 1599/1986 τιμωρείται με φυλάκιση τουλάχιστον τριών μηνών. Εάν ο υπαίτιος αυτών των πράξεων σκόπευε να προσπορίσει στον εαυτόν του ή σε άλλον περιουσιακό όφελος βλάπτοντας τρίτον ή σκόπευε να βλάψει άλλον, τιμωρείται με κάθειρξη μέχρι 10 ετών.»

ΕΥΧΑΡΙΣΤΙΕΣ

Θα ήθελα αρχικά να ευχαριστήσω τον επιβλέποντα καθηγητή της παρούσας εργασίας Δρ. Αντώνιο Δαδαλιάρη για την υποστήριξη και την καθοδήγηση που μου παρείχε στο διάστημα συγγραφής της εργασίας. Έπειτα θα ήθελα να ευχαριστήσω την γραμματεία του τμήματος, η οποία ήταν πάντα βοηθητική και φιλική απέναντι μου. Ακόμη, τους καλούς μου φίλους και συμφοιτητές Χαράλαμπο Μυλωθρό και Αλέξανδρο Αργυρίου για την συμβολή τους σε σχεδιαστικές επιλογές και την προθυμία τους για τη δοκιμή της εφαρμογής στα διάφορα στάδια ανάπτυξης της. Τέλος, ευχαριστώ την οικογένεια μου για την καθολική στήριξη σε όλη μου την πορεία ως μαθητής-φοιτητής.

ΠΕΡΙΛΗΨΗ

Η παρούσα εργασία περιγράφει την διαδικασία ανάπτυξης και σχεδιασμού και την γενική λειτουργία μιας διαδικτυακής εφαρμογής διαχείρισης παραγγελιών, η οποία προορίζεται για χρήση σε χώρους μαζικής εστίασης όπως εστιατόρια, καφετέριες, ξενοδοχεία κ.α. Ακόμη, θα γίνει περιγραφή και των τεχνολογιών που χρησιμοποιήθηκαν. Η εφαρμογή έχει κατασκευαστεί χρησιμοποιώντας σύγχρονες αναλογικά με την εποχή συγγραφής του κειμένου διαδικτυακές τεχνολογίες, όπως HTML5, CSS3 (responsive) , javascript, Vue js, node js, express js, axios και MongoDb. Η επιλογή για κατασκευή πάνω σε πλατφόρμες web είναι σκόπιμη και έγινε για λόγους συμβατότητας με μια πληθώρα συσκευών όπως κινητά, συσκευές tablet , σταθερούς και φορητούς υπολογιστές. Η γενική φιλοσοφία σχεδιασμού δίνει προτεραιότητα στην φιλικότητα προς το χρήστη και την όσο δυνατόν περισσότερη ανεξαρτητοποίηση της εφαρμογής από συμπληρωματικές συσκευές και επιπλέον εξοπλισμό.

ABSTRACT

The present essay describes the development and design procedure as well as the general function of a web application used for taking and editing orders in places such as hotels, restaurants, coffee shops ect. Furthermore, there will be an extended description regarding all the technologies used. The application is build using modern web development tools and platforms like HTML5, CSS3 (responsive) , javascript, Vue js, node js, express js, axios and MongoDb. The choice of building a web application as opposed to an android, iOS or Windows program is to maximize compatibility between devices such as mobile phones, tablets, laptop and Desktop Computers. The design philosophy prioritizes user friendliness and making the app as self-dependent as possible, thus reducing the need of complimentary/extra equipment.

ΠΕΡΙΕΧΟΜΕΝΑ

ΚΕΦΑΛΑΙΟ 1ο – ΕΙΣΑΓΩΓΗ

1.1 ΣΥΝΤΟΜΗ ΠΕΡΙΓΡΑΦΗ

Αρχική σκέψη ήταν η δημιουργία μιας εφαρμογής λήψης και διαχείρισης παραγγελιών, η οποία θα είναι όσο περισσότερο αυτόνομη γίνεται, θα είναι συμβατή με όλες τις συσκευές και δεν θα απαιτεί ειδικό εξοπλισμό, όπως PDAs, επιπλέον οθόνες ή δικτυακό εξοπλισμό πέρα από τον σύνηθες που διαθέτει η πλειοψηφία του κόσμου. Παράλληλα η εφαρμογή θα πρέπει να δίνει αρκετές επιλογές παραμετροποίησης των παραγγελιών και πολλές χρήσιμες πληροφορίες, ώστε να υπάρξει ουσιαστική διευκόλυνση στην διαδικασία λήψης παραγγελιών, με τρόπο όμως τέτοιο, ώστε να μην δημιουργεί σύγχυση στους εργαζόμενους. Δηλαδή κάνοντας χρήση μιας απλής διεπαφής χρήστη που θα πρέπει να είναι συγκρίσιμη όσον αφορά την ταχύτητα με την απλή γραφή.

Αρχικά ο χρήστης κάνει σύνδεση με τα στοιχεία εισόδου του, η εφαρμογή αναγνωρίζει τον ρόλο του και τον παραπέμπει στην κεντρική σελίδα η οποία έχει για κάθε ρόλο διαφορετικές επιλογές. Έπειτα ορισμένοι ρόλοι όπως του σερβιτόρου έχουν την επιλογή να εκτελέσουν μια νέα παραγγελία συμπληρώνοντας τον αριθμό του τραπεζιού υποχρεωτικά και τον αριθμό ατόμων, αν θέλουν. Η παραγγελία ενσωματώνεται με άλλες πληροφορίες, όπως ώρα, κατάσταση και παρατηρήσεις (per object ή γενικές) του τραπεζιού στη βάση δεδομένων. Η παραγγελία τότε γίνεται ορατή σε όλες τις συνδεδεμένες συσκευές, έτσι μπορεί να αρχίσει να εκτελείται χωρίς την ανάγκη της προβληματικής πολλές φορές λεκτικής συνεννόησης μεταξύ των διαφόρων ρόλων, εξοικονομώντας με αυτό τον τρόπο χρόνο. Ένα σύστημα ειδοποιήσεων που ενημερώνει για την κατάσταση του εκάστοτε τραπεζιού διασφαλίζει την συνέχιση όσο είναι δυνατόν της παραπάνω διευκόλυνσης. Ένα σύστημα δυναμικής προβολής των παραγγελιών με τρόπο τέτοιον ώστε να επιταχύνεται η διαδικασία ανάγνωσης τους ανάλογα με τις ανάγκες του κάθε ρόλου έχει επίσης εφαρμοστεί. Συμπερασματικά, η εφαρμογή θα πρέπει να συνδράμει στην έγκαιρη και αποτελεσματική διεκπεραίωση των παραγγελιών καθ’ όλη τη διάρκεια παραμονής των πελατών από όλα τα σημεία της επιχείρησης.

1.2 ΣΚΟΠΟΣ

Σκοπός αυτής της εφαρμογής είναι ο αυτοματισμός κάποιων διαδικασιών και η διευκόλυνση και όχι αντικατάσταση των εργαζομένων. Πιο συγκεκριμένα η αυτοματοποίηση της διαδικασίας παραγγελίας μπορεί να αποφέρει τα εξής πλεονεκτήματα:

- Αποφυγή λαθών εξαιτίας τυπογραφικών ή ασυνεννοησίας.

- Αποφυγή κάλυψης οποιασδήποτε απόστασης για να μεταδοθεί πληροφορία δηλαδή, άμεση παράδοση της παραγγελίας στα διάφορα σημεία υλοποίησης της.
- Μείωση κόστους καθώς εξαλείφεται η ανάγκη για στυλό, δελτία παραγγελίας κ.α.
- Παροχή χρήσιμων στατιστικών για την λειτουργία της επιχείρησης.

1.3 ΣΥΝΟΠΤΙΚΑ ΤΟ ΠΕΡΙΕΧΟΜΕΝΟ ΠΟΥ ΑΝΑΛΥΕΤΑΙ ΣΤΑ ΕΠΟΜΕΝΑ ΚΕΦΑΛΑΙΑ

Τεχνολογικό υπόβαθρο Στο δεύτερο κεφάλαιο θα δοθεί μια σύντομη αλλά περιεκτική γενική εικόνα για όλες τις τεχνολογίες που χρησιμοποιήθηκαν στην εφαρμογή, για τους λιγότερο καταρτισμένους αναγνώστες θα συμπεριληφθούν και παραδείγματα που δεν θα έχουν άμεση σχέση με την εφαρμογή αλλά που θα βοηθήσουν στην κατανόηση του κώδικα της εφαρμογής που θα παρουσιαστεί σε βάθος στο τρίτο κεφάλαιο.

Ανάπτυξη και σχεδιασμός. Στο τρίτο κεφάλαιο θα γίνει αναλυτική παρουσίαση του κώδικα HTML, CSS και Javascript για την υλοποίηση όλων των παραπάνω λειτουργιών καθώς και του σχεδιασμού που αλλάζει ανάλογα τη συσκευή και το ρόλο του χρήστη. Πιο συγκεκριμένα θα αναλυθούν:

- Οθόνη εισόδου
  Η οθόνη οπού οι χρήστες θα εισάγουν τα στοιχεία τους που στη συνέχεια θα ελέγχονται και θα συγκρίνονται με αυτά της βάσης δεδομένων.

- Κεντρική οθόνη
  Εδώ θα γίνεται εμφάνιση όλων των παραγγελιών σε μορφή τραπεζιών με νούμερο και ένα χρώμα που θα δηλώνει την κατάσταση στην οποία βρίσκονται. Αυτή η οθόνη θα διαφοροποιείται λίγο έως πολύ ανάλογα με το ρόλο του εισαχθέντος στο σύστημα χρήστη
- Σύστημα δυναμικών ειδοποιήσεων με τη χρήση χρωμάτων

- Σύστημα επιλεκτικής εμφάνισης πληροφοριών ανάλογα με το ρόλο χρήστη

- Σύστημα σημειώσεων

- Σύστημα επιπρόσθετων ειδών

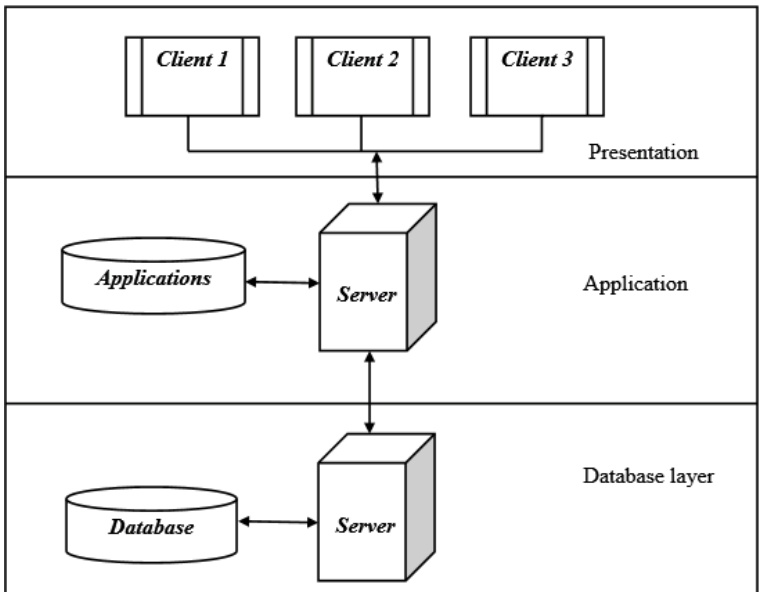
- Οθόνη παραγγελίας
  Σε αυτή την οθόνη εμφανίζεται ο κατάλογος του καταστήματος κατηγοριοποιημένος, τα προϊόντα κάθε κατηγορίας, καθώς και η παραγγελία κάθε στιγμή με επιλογές προθήκης και αφαίρεσης.

ΚΕΦΑΛΑΙΟ 2ο – ΤΕΧΝΟΛΟΓΙΚΟ ΥΠΟΒΑΘΡΟ

2.1 ΜΟΝΤΕΛΟ ΠΕΛΑΤΗ-ΔΙΑΚΟΜΙΣΤΗ (CLIENT-SERVER)

Η παρούσα εφαρμογή όπως και η πλειοψηφία των διαδικτυακών εφαρμογών, είναι χτισμένη επάνω στο μοντέλο πελάτη-διακομιστή. Στην επιστήμη των υπολογιστών το μοντέλο αυτό περιγράφεται ως η συνηθέστερη μέθοδος ανάπτυξης λογισμικού στο διαδίκτυο, στην οποία μια συσκευή που ονομάζουμε πελάτης πραγματοποιεί μια αίτηση σε μια άλλη, την οποία ονομάζουμε διακομιστή ή εξυπηρετητή. Ο εξυπηρετητής με τη σειρά του απαντά παρέχοντας τις πληροφορίες ή απορρίπτοντας την αίτηση. Ένας εξυπηρετητής μπορεί να εξυπηρετεί πολλαπλούς πελάτες ταυτόχρονα ή σειριακά, με σειρά προτεραιότητας. Τόσο οι ενέργειες που εκτελούνται κατά την αίτηση όσο και κατά την απάντηση πραγματοποιούνται σε διαφορετικές διεργασίες του ίδιου ή διαφορετικού υπολογιστικού συστήματος. Θεωρούμε λοιπόν ότι η εκάστοτε διαδικτυακή εφαρμογή εκτελείται κατά μήκος μιας σειράς κατανεμημένων συστημάτων όπου ο πελάτης και ο διακομιστής είναι διαφορετικά τμήματα της ίδιας κατανεμημένης εφαρμογής. Τα κατανεμημένα συστήματα μάλιστα είναι απαραίτητο και να επικοινωνούν για λόγους χρόνο-δρομολόγησης και συγχρονισμού για τη σωστή σειρά εκτέλεσης της εφαρμογής.

Η πιο σύγχρονη μορφή του παραπάνω μοντέλου ονομάζεται three tier client-server model δηλαδή μοντέλο τριών επιπέδων. Συγκεκριμένα το πρώτο επίπεδο είναι υπεύθυνο για την παρουσίαση της εφαρμογής δηλαδή την διεπαφή χρήστη. Το δεύτερο επίπεδο αναλαμβάνει την επιχειρησιακή λογική της εφαρμογής ενώ στο τρίτο στεγάζονται τα δεδομένα.



Εικόνα 2.1

Τα πλεονεκτήματα της χρήσης το μοντέλου αυτού είναι:

- Ασφάλεια, καθώς δεν έχουμε άμεση προσπέλαση του χρήστη στη βάση δεδομένων.
- Ευελιξία / Καλύτερος έλεγχος ροής δεδομένων, καθώς έχουμε τρία επίπεδα μπορούμε να διαχωρίσουμε την λειτουργικότητα της εφαρμογής και έτσι να βελτιώσουμε και την απόδοση της. Επίσης γίνεται ευκολότερη η παραμετροποίηση κάποιου τμήματος της εφαρμογής δίχως να επηρεαστούν τα υπόλοιπα. (Για παράδειγμα μια αλλαγή στο επίπεδο παρουσίασης δεν θα επηρεάσει το επίπεδο λογικής).
- Ευκολότερη διαχείριση.

Στην εφαρμογή μας, η λογική διαμοιράζεται στο πρώτο και στο δεύτερο επίπεδο για λόγους που θα περιγράφουν σε επόμενο κεφάλαιο, όπου και θα φανεί η διαφορά στις έννοιες επιχειρησιακή λογική και λογική παρουσίασης (Business logic – Presentation logic).

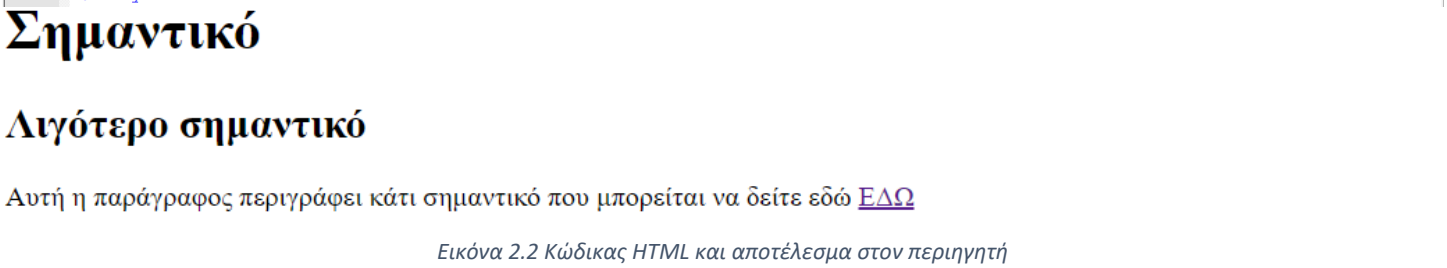
2.2 Η ΓΛΩΣΣΕΣ HTML ΚΑΙ CSS

Οι γλώσσες HTML (hyper text markup language) και CSS (cascading style sheets), δεν είναι γλώσσες προγραμματισμού αλλά γλώσσες σήμανσης και μορφοποίησης, αυτό σημαίνει ότι σε μια ιστοσελίδα ή διαδικτυακή εφαρμογή αυτές είναι υπεύθυνες για τη δομή την τοποθέτηση και μορφή των γραφικών στοιχείων, όπως κείμενο, εικόνες, υπέρ-σύνδεσμοι και άλλα στην οθόνη φτιάχνοντας έτσι το γραφικό περιβάλλον χρήστη.

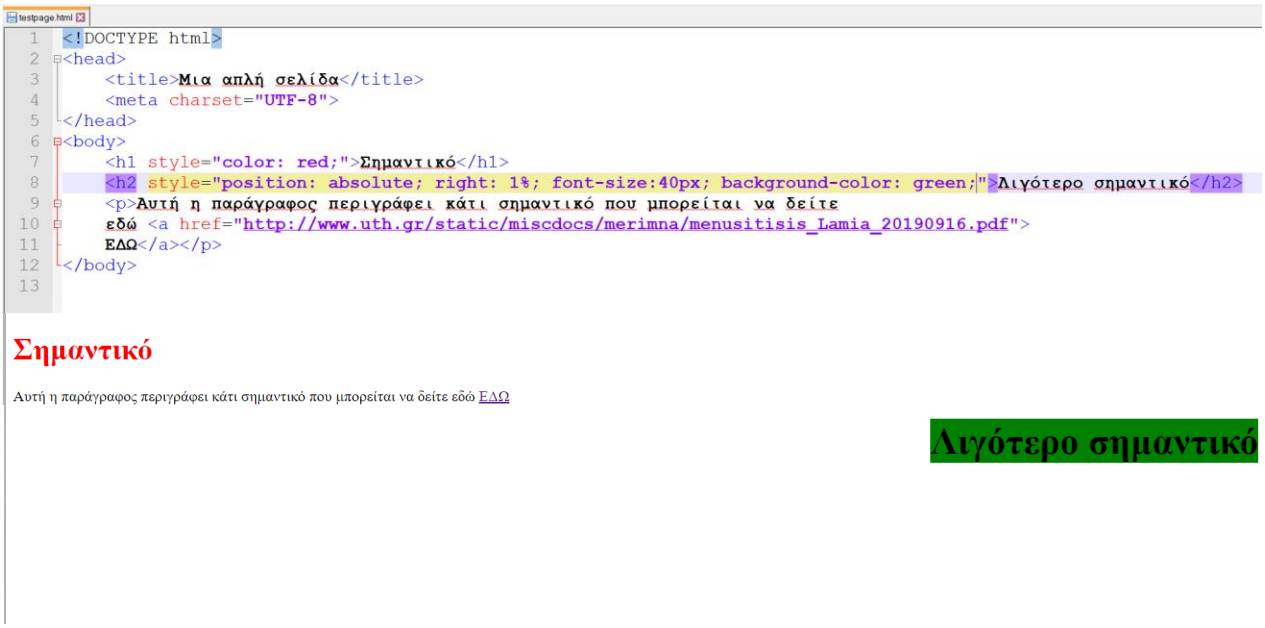
Γενικό παράδειγμα χρήσης:

testpage.html

```
<!DOCTYPE html>
<head>
    <title>Μια απλή σελίδα</title>
    <meta charset="UTF-8">
</head>
<body>
    <h1>Σημαντικό</h1>
    <h2>Λιγότερο σημαντικό</h2>
    <p>Αυτή η παράγραφος περιγράφει κάτι σημαντικό που μπορείται να δείτε
    εδώ <a href="http://www.uth.gr/static/miscdocs/merimna/menusitisis_Lamia_20190916.pdf">
    ΕΔΩ</a></p>
</body>
```



Εικόνα 2.2 Κώδικας HTML και αποτέλεσμα στον περιηγητή



Εικόνα 2.3 Κώδικας HTML+CSS και αποτέλεσμα στον περιηγητή

Όπως βλέπουμε η HTML καθορίζει το περιεχόμενο και τη σειρά εμφάνισης αυτού, ενώ η CSS καθορίζει τη μορφολογία του και διάφορα χαρακτηριστικά όπως η θέση το χρώμα και άλλα.

2.3 Η Γλώσσα Javascript

Η Javascript είναι μια γλώσσα προγραμματισμού και αποτελεί βασική τεχνολογία του παγκόσμιου ιστού (World Wide Web), είναι γλώσσα υψηλού επιπέδου και δίνει δυνατότητα προσθήκης δυναμικών στοιχείων σε διαδικτυακές εφαρμογές και σελίδες αλληλοεπιδρώντας παράλληλα και με τον κώδικα HTML και CSS. Χρησιμοποιεί διερμηνέα ο οποίος συνήθως είναι μέρος ενός περιβάλλοντος εκτέλεσης ενός περιηγητή, παραδείγματα αποτελούν οι: V8 της Google και SpiderMonkey της Monzilla (χρήση σε Chrome και Firefox αντίστοιχα). Η γλώσσα υποστηρίζει αντικειμενοστραφές, προστακτικό καθώς και συναρτησιακό «στυλ» προγραμματισμού και ξεκίνησε αρχικά ως “client-side γλώσσα”, αυτό σημαίνει ότι ο κώδικας εκτελείται στη συσκευή του πελάτη και όχι στον διακομιστή.

Παράδειγμα:

test.html

```html
<!DOCTYPE html>
<html>
<head>
    <title>Μια απλή σελίδα</title>
    <meta charset="UTF-8">
</head>
<body>
    <input id="num1" type="number">
    <select id="choice">
        <option value="add">+</option>
        <option value="sub">-</option>
        <option value="mul">*</option>
        <option value="div">/</option>
    </select>
    <input id="num2" type="number"> <button onclick="calculate();">Υπολόγισε</button>
    <br><h3 id="result"></h3>
</body>
</html>

<!--JAVASCRIPT CODE -->
<script>
    function calculate(){
        let x,y,res;
        //Gets first input
        x = document.getElementById("num1").value;
        //Gets second input
        y = document.getElementById("num2").value;
        //Gets operation selection from select element
        let dummy = document.getElementById("choice");
        let opeation = dummy.options[dummy.selectedIndex].value;
        //Calculation
        switch (opeation){
            case "add":
                res = +x + +y;
                break;
            case "sub":
                res = +x - +y;
                break;
            case "mul":
                res = +x * +y;
                break;
            case "div":
                res = +x / +y;
                break;
            default:
                console.log("something went wrong");
        }
        //Write to screen
        document.getElementById("result").innerHTML = res;
    }
</script>
```

Εικόνα 2.4 Κώδικας για την κατασκευή βασικής αριθμομηχανής σε Javascript

- Κώδικας HTML και Javascript που παράγει μια αριθμομηχανή, ικανή να εκτελέσει τέσσερις βασικές πράξεις. Ο κώδικας εκτελείται στον περιηγητή με τον τρόπο που φαίνεται παρακάτω.

Εικόνα 2.5 Αποτέλεσμα εκτέλεσης κώδικα (εικόνα 2.4) στον περιηγητή

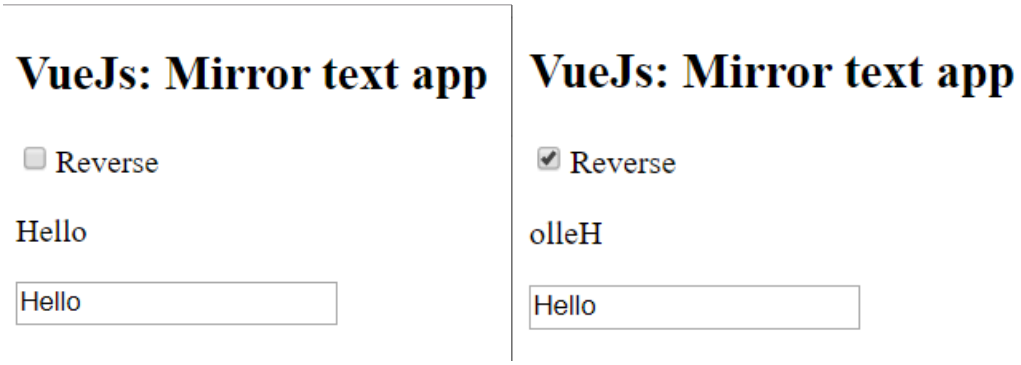
2.4 Javascript Frameworks και Vue JS

Τα “frameworks” της Javascript δεν είναι τίποτα παραπάνω από μια συλλογή από βιβλιοθήκες που σε κάποιες περιπτώσεις εισάγουν και νέα σύνταξη που λειτουργεί παράλληλα με αυτή της «απλής» Javascript. Ανάμεσα στα πιο δημοφιλή frameworks ανήκουν τα Angular JS, React, Vue JS και άλλα. Τα πλεονεκτήματα της χρήσης των προαναφερθέντων είναι μεταξύ άλλων η απλούστευση και καλύτερη διαχείριση του κώδικα και η επιλεκτική εμφάνιση (selective rendering) η οποία ανανεώνει μόνο τα στοιχεία της σελίδας που αλλάζουν κάθε φορά. Ένα απλό παράδειγμα που συμπεριλαμβάνει κώδικα Vue JS ακολουθεί παρακάτω.

```
<body>

<h2>VueJs: Mirror text app</h2>

<div id="app">
  <input type="checkbox" name="reversebox" v-on:click="togglecheck">Reverse
  <p v-if="!checked">{{message}}</p>
  <p v-if="checked">{{reversed}}</p>
  <input v-model="message" v-on:keyup="mirror">
</div>

<script>
App = new Vue({
  el: '#app',
  data:
  {
    message: '',
    reversed: '',
    checked: false
  },
  methods:
  {
    togglecheck : function(){
        if(this.checked){
            this.checked = false;
        }
        else
        {
            this.checked = true;
            this.mirror();
        }
    },
    mirror : function(){
        if(!this.checked)
            return;
        this.reversed = this.message.split('').reverse().join('');
    }
  }
})
</script>
```

Εικόνα 2.6 Εφαρμογή αντιστροφής κειμένου σε Vue

Ο παραπάνω κώδικας εκτελείται αποκλειστικά στη συσκευή πελάτη και δίνει το ακόλουθο αποτέλεσμα στον περιηγητή.



Εικόνα 2.7 Αποτέλεσμα εκτέλεσης κώδικα (εικόνα 2.6), στον περιηγητή

Παρατηρούμε ότι το αντικείμενο Vue έχει δυο στοιχεία (components), το “data” στο οποίο μπορούμε να ορίζουμε μεταβλητές και το “methods” στο όποιο ορίζουμε νέες μεθόδους. Η Vue περιλαμβάνει και αλλά τέτοια στοιχεία που θα αναλυθούν στο επόμενο κεφάλαιο. Τις μεταβλητές που ορίζουμε στο data μπορούμε να τις προσπελάσουμε μέσα από το αντικείμενο Vue χρησιμοποιώντας το αναγνωριστικό “this” και μέσα στον κώδικα HTML περικλείοντας τα στο αναγνωριστικό “{{ }}”. Αξίζει να γίνει και μια αναφορά στον τρόπο με τον οποίο μας δίνεται η δυνατότητα να διαχειριστούμε συμβάντα στη Vue προσθέτοντας κώδικα απευθείας μέσα σε στοιχεία html, παραδείγματα αποτελούν τα αναγνωριστικά “v-on:click” και “v-on:keyup” που εκτελούν ένα μπλοκ κώδικα ή καλούν μια μέθοδο, όταν πραγματοποιείται ένα συμβάν όπως η πίεση ενός πλήκτρου, το “click” ενός κουμπιού και αλλά. Ακόμη, τα αναγνωριστικά “v-bind” που συσχετίζουν σε πραγματικό χρόνο τις τιμές ενός στοιχείου εισόδου (input) της html, με μια μεταβλητή εισχωρώντας τις τιμές του σε αυτήν, αλλά και το αναγνωριστικό “v-if”, το οποίο καθιστά ένα στοιχείο html ορατό, δεδομένης της αληθείας μιας συνθήκης ελέγχου (conditional rendering).

2.5 Ενδιάμεσο λογισμικό και Express Js

Η επικοινωνία ανάμεσα στους πελάτες και τους εξυπηρετητές δεν είναι πάντα άμεση, πολλές φορές το αίτημα πριν φτάσει στον εξυπηρετητή περνά και από άλλα ενδιάμεσα στρώματα λογισμικού τα οποία ονομάζουμε ενδιάμεσο λογισμικό.

«Το ενδιάμεσο λογισμικό είναι ένας ευρύς όρος που καλύπτει όλα τα κατανεμημένα λογισμικά που χρειάζονται για την υποστήριξη της συνεργασίας πελατών και διακομιστών. Πού αρχίζει το ενδιάμεσο λογισμικό και πού τελειώνει; Αρχίζει με την ομάδα API στην πλευρά του πελάτη που χρησιμοποιείται για την κλήση μιας υπηρεσίας, και καλύπτει την μεταβίβαση του αιτήματος στο δίκτυο και την αντίστοιχη απάντηση.» Ορισμός Orfali, Harkey και Edwards

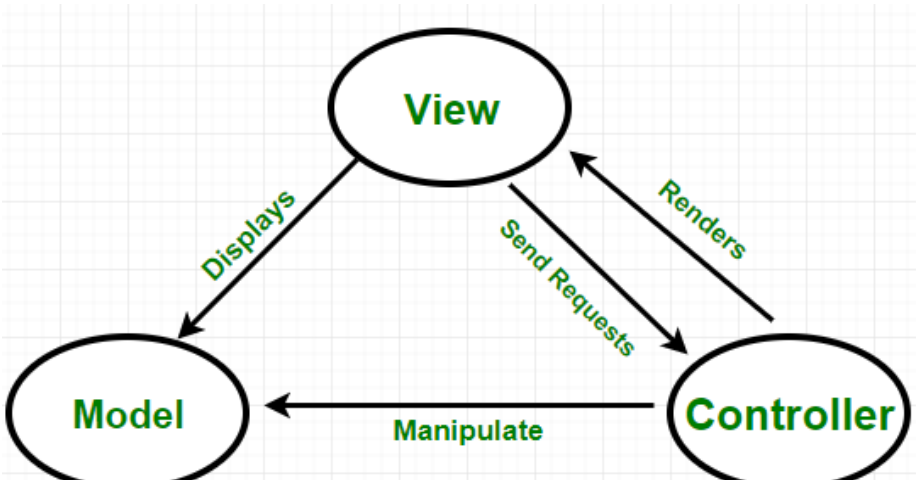
Το Express Js είναι ακόμη ένα framework της Javascript και συγκεκριμένα ένα πρόσθετο (module) της node Js (βλέπε κεφάλαιο 2.6)

```
const express = require('express')
const app = express()
const port = 3000

app.get('/', (req, res) => res.send('Hello World!'))

app.listen(port, () => console.log(`Example app listening on port ${port}!`))
```

Εικόνα 2.8 Κώδικας δημιουργίας μιας βασικής εφαρμογής διακομιστή σε express js

Στο παραπάνω παράδειγμα δημιουργούμε έναν εξυπηρετητή ο οποίος ακούει για αιτήματα στη θύρα 3000, μόλις ο πελάτης εισέλθει στο κεντρικό φάκελο (root directory – '/') τότε θα εκτελεστεί η συνάρτηση που θα εκτυπώσει στην οθόνη το μήνυμα "Hello World".

Το Express Js βασίζεται στο μοντέλο αρχιτεκτονικής MVC (Model View Controller). Αναφερόμενοι στα τρία δομικά στοιχεία του μοντέλου μπορούμε να πούμε ότι: Το "model" διαχειρίζεται τα δεδομένα που μετακινούνται στο δίκτυο, το "view" αποτελεί την διεπαφή χρήστη (User interface), ενώ το controller είναι υπεύθυνο για τη δρομολόγηση των αιτημάτων. Κατά το αίτημα ενός πελάτη οι δρομολογητές που "ακούνε" για συμβάντα, λαμβάνουν το μονοπάτι αιτήματος (request url) και εκτελούν τον αντίστοιχο κώδικα ή αρχείο που βρίσκεται στο μονοπάτι αυτό. Να σημειωθεί ότι, το μονοπάτι δεν είναι απαραίτητα πραγματικό αλλά μπορεί να είναι και εικονικό, αρκεί να έχουμε δημιουργήσει τον αντίστοιχο δρομολογητή που "ακούει" για το συγκεκριμένο http "path based" αίτημα. Το παρακάτω διάγραμμα απεικονίζει τη λειτουργία του μοντέλου MPV και την πορεία των δεδομένων κατά ένα αίτημα πελάτη.



Εικόνα 2.9

2.6 Node Js

Το node js είναι ένα περιβάλλον εκτέλεσης (runtime environment) και framework της javascript που επιτρέπει την εκτέλεση κώδικα javascript στον διακομιστή. Κάνοντας χρήση της node ανοίγεται η δυνατότητα της σχεδόν αποκλειστικής χρήσης της γλώσσας javascript σε όλο το «φάσμα» της εφαρμογής, δηλαδή από τη συσκευή του πελάτη μέχρι και τον διακομιστή. Η node είναι σε μεγάλο βαθμό επεκτάσιμη αφού υποστηρίζει διάφορα πρόσθετα τα οποία επεκτείνουν τη λειτουργικότητα της εφαρμογής.

Αφού έχουμε εγκαταστήσει στο σύστημα τη node και το node package manager (npm) , μπορούμε έπειτα από το τερματικό του λειτουργικού μας συστήματος να πληκτρολογήσουμε την εντολή npm install <module> για να κάνουμε εγκατάσταση του επιθυμητού πρόσθετου. Ορισμένα από τα "modules" που χρησιμοποιήθηκαν στην παρούσα εφαρμογή είναι τα: Crypto, Express, pug, Cookie-Parser, MongoDb. Δημιουργούμε μια νέα εφαρμογή express πληκτρολογώντας αρχικά την εντολή "npm install -g express-generator" και στη συνέχεια την εντολή "express –view=pug myapp". Οι παραπάνω εντολές θα δημιουργήσουν ένα φάκελο με όλα τα αρχεία που χρειάζονται για την βασική λειτουργία μιας διαδικτυακής εφαρμογής. Συγκεκριμένα στο αρχείο package.json μπορούμε να ορίσουμε το πρώτο αρχείο το οποίο θα εκτελεστεί κατά την εκκίνηση της εφαρμογής μας (εντολή "npm start"), στο πεδίο "start" έχει οριστεί ως προεπιλεγμένο αρχείο εκτέλεσης το www.

package.json | app.js | www

```
{
  "name": "myapp",
  "version": "0.0.0",
  "private": true,
  "scripts": {
    "start": "node ./bin/www"
  },
  "dependencies": {
    "axios": "^0.18.1",
    "client-sessions": "^0.8.0",
    "cookie-parser": "~1.4.3",
    "debug": "~2.6.9",
    "express": "~4.16.0",
    "express-session": "file:express-session",
    "http-errors": "~1.6.2",
    "mongodb": "^3.2.7",
```

Εικόνα 2.10 Αρχείο package.json

Στο αρχείο www θα δούμε ότι υπάρχει ο απαραίτητος κώδικας για την δημιουργία ενός διακομιστή στον υπολογιστή μας, αναλυτικότερα εδώ υπάρχει ο κώδικας για την δημιουργία του διακομιστή και ο ορισμός της θύρας στην οποία αυτός θα "ακούει" για αιτήματα, ενώ

πραγματοποιείται επίσης η διαχείριση των σφαλμάτων που μπορεί να συμβούν κατά ένα αίτημα.

```
var port = normalizePort(process.env.PORT || '3000');
app.set('port', port);

/**
 * Create HTTP server.
 */

var server = http.createServer(app);

/**
 * Listen on provided port, on all network interfaces.
 */

server.listen(port);
server.on('error', onError);
server.on('listening', onListening);
```

Εικόνα 2.11 Κώδικας δημιουργίας διακομιστή, αρχείο www

Στη συνέχεια καλείται το αρχείο app.js, σε αυτό το αρχείο άξιο αναφοράς είναι πως ορίζεται το αρχείο δρομολόγησης “routing” (βλέπε κεφ. 2.5), για τα HTTP αιτήματα Get ή Post, εδώ γίνεται επίσης η αρχική διαμόρφωση του “view engine” (βλέπε κεφ. 2.5).

```
// view engine setup
app.set('views', path.join(__dirname, 'views'));
app.set('view engine', 'pug');

app.use(logger('dev'));
app.use(express.json());
app.use(express.urlencoded({ extended: false }));
app.use(cookieParser());
app.use(express.static(path.join(__dirname, 'public')));

app.use('/', indexRouter);
app.use('/users', usersRouter);
app.use(express.static('public'));

// catch 404 and forward to error handler
app.use(function(req, res, next) {
  next(createError(404));
```

Εικόνα 2.12 Κώδικας δημιουργίας διακομιστή, αρχείο app.js

Τέλος πηγαίνοντας στο αρχείο ”/routes/index.js”, θα βρούμε τον κώδικα με τον οποίο δρομολογείται το αίτημα Get στο “root directory” / του διακομιστή προς το αρχείο html που επιθυμούμε να βλέπει πρώτα ο εκάστοτε χρήστης.

index.js app.js www

```
1  var express = require('express');
2  var bend = require('../public/javascripts/backend');
3
4  var router = express.Router();
5  var app = express();
6  const crypto = require('crypto');
7
8
9  //Root directory redirect
10 router.get('/', function(req, res, next) {
11     res.sendFile('login.html', { root: "public" } );
12 });
```

Εικόνα 2.13 Κώδικας δημιουργίας διακομιστή, αρχείο index js

Να τονίσουμε ότι η οργάνωση και διαμοιρασμός του κώδικα σε αυτά τα αρχεία δεν είναι απαραίτητος, αλλά αποτελεί παρόλα αυτά, τη συνήθη δομή οργάνωσης σε μια εφαρμογή express/node js.

2.7 Βάσεις Δεδομένων

Βάσεις δεδομένων ονομάζουμε τις συλλογές δεδομένων που είναι οργανωμένες με τρόπο τέτοιο ώστε να μπορούν να εκτελεστούν εύκολα οι πράξεις της αναζήτησης, προβολής, προσπέλασης και ενημέρωσης τους. Για το λόγο αυτό, χρησιμοποιούν πολλές διαφορετικές δομές δεδομένων όπως δέντρα, λίστες και πίνακες κατακερματισμού. Οι βάσεις δεδομένων μπορούν να περιέχουν δεδομένα πολλών ειδών όπως ονόματα, ηλικίες, ημερομηνίες και άλλα, καθώς υποστηρίζουν την αποθήκευση των πιο γνωστών τύπων δεδομένων (int, float, Boolean, char, sting, array, binary και αλλά). Οι δυο κυριότερες κατηγορίες βάσεων είναι οι σχεσιακές και οι μη σχεσιακές βάσεις δεδομένων (SQL ή NoSQL).

- **Σχεσιακές βάσεις δεδομένων (SQL)**

Οι σχεσιακές βάσεις δεδομένων είναι οργανωμένες σε πίνακες με ορισμένα κάθε φορά πεδία που ονομάζουμε στήλες, κάθε σειρά ή καταχώρηση σε ένα πίνακα έχει τον ίδιο αριθμό πεδίων που συσχετίζονται με αυτήν. Για παράδειγμα σε μια έναν πίνακα όπου θα αποθηκεύονταν οι πελάτες ενός καταστήματος θα μπορούσε να υπάρχει μια καταχώρηση με τα εξής πεδία:

- Idπελάτη (Πρωτεύον κλειδί): 33 (int)
- Όνομα: ‘Δημήτριος Παπαγιαννάκης’ (string)
- Ηλικία: 27 (int)

Μια δεύτερη καταχώρηση θα έπρεπε να είχε ακριβώς τα ίδια πεδία, να σημειωθεί όμως γενικότερα πως υπάρχει δυνατότητα κενών τιμών σε πεδία που το επιτρέπουν.

Έστω ότι έχουμε άλλον ένα πίνακα στον οποίο κρατάμε τα προϊόντα τις παραγγελίες του καταστήματος, με δομή:

- Idπαραγγελίας (Πρωτεύον κλειδί): 221 (int)
- Idπελάτη(Ξένο κλειδί): 33 (int)
- Περιεχόμενα: 'Γάλα άπαχο 1L' (string)
- Τιμή: 1,79 (float)
- Ημερομηνία αγοράς: 22/06/2019 (date)

Το πεδίο 'id πελάτη' παραπέμπει στον πίνακα πελατών, στον πίνακα αυτό είναι μοναδικό και το ονομάζουμε πρωτεύον κλειδί(primary key), στον δεύτερο πίνακα το ίδιο πεδίο υποδεικνύει τη σχέση μεταξύ των δύο πινάκων και το ονομάζουμε ξένο κλειδί.

Παρατηρούμε στο συγκεκριμένο παράδειγμα, πως ένας πελάτης μπορεί να έχει εκτελέσει πολλαπλές παραγγελίες, αλλά κάθε παραγγελία σχετίζεται με μόνο έναν πελάτη, για τον λόγο αυτό η παραπάνω σχέση ονομάζεται «ένα προς πολλά». Αντίστοιχα υπάρχουν και σχέσεις «ένα προς ένα» και «πολλά προς πολλά», παράδειγμα μιας σχέσης «ένα προς ένα» θα μπορούσε να αποτελεί ένας πίνακας διευθύνσεων όπου κάθε πελάτης θα σχετιζόταν αυστηρά με μια και μόνο διεύθυνση, ενώ για αναπαράσταση μιας σχέσης «πολλά προς πολλά» συνήθως χρησιμοποιείται ένας τρίτος πίνακας που περιέχει τις τιμές των δύο πρωτευόντων κλειδιών των πινάκων που συσχετίζει.

Χαρακτηριστική γλώσσα για τη δημιουργία και επεξεργασία σχεσιακών βάσεων δεδομένων αποτελεί η MySQL. Ένα αίτημα στη βάση δεδομένων που ταιριάζει και με το παραπάνω παράδειγμα θα μπορούσε να μοιάζει κάπως έτσι.

"SELECT clients.Όνομα, orders.Περιεχόμενα FROM clients INNER JOIN orders ON clients.Idπελάτη = orders.Idπελάτη" WHERE clients.idΠελάτη = 33"

Ο παραπάνω κώδικας θα μας έδινε όλες τις παραγγελίες του πελάτη με id 33 εμφανίζοντας μας το όνομα του και ακριβώς δίπλα τα περιεχόμενα της παραγγελίας.

- **Μη σχεσιακές βάσεις δεδομένων (Not only SQL - NoSQL)**

Οι μη σχεσιακές βάσεις δεδομένων διαδίδονται ολοένα και περισσότερο, διατηρούν σε μεγάλο βαθμό την οργάνωση που είδαμε παραπάνω αφού περιέχουν δομές που μοιάζουν με πίνακες, η κάθε μια με τα αντίστοιχα πεδία. Η κύρια διαφορά εντοπίζεται στο ότι κάθε καταχώρηση ενός «πίνακα» μπορεί να έχει τα δικά της ξεχωριστά πεδία, η διαφορά αυτή προσφέρει ευλυγισία στον τρόπο αποθήκευσης των δεδομένων, καθώς δεν χρειάζεται να συμπεριλαμβάνουμε όλα τα πεδία σε κάθε καταχώρηση, ενώ μας δίνεται η ελευθερία να έχουμε πιθανώς και παραπάνω

πεδία μόνο στις καταχωρήσεις στις οποίες αυτό απαιτείται. Ας δώσουμε ένα παράδειγμα για να γίνει το παραπάνω πιο ξεκάθαρο.

Έστω ότι έχουμε δημιουργήσει σε μια μη σχεσιακή βάση δεδομένων τον πίνακα ή συλλογή, όπως θα την ονομάζουμε από εδώ και πέρα, στην οποία κρατάμε τα στοιχεία των επισκεπτών σε ένα νοσοκομείο. Μια καταχώρηση ή έγγραφο (document) όπως ονομάζεται σε αυτή τη συλλογή θα μπορούσε να είχε την εξής μορφή:

- Id: 45
- Τύπος: Ασθενής
- Όνομα: Παναγιώτης Παπαδόπουλος
- Ηλικία: 55
- Πάθηση: Πνευμονία
- Αλλεργίες: Πενικιλίνη
- Φαρμακευτική αγωγή: Aerolin
- Συχνότητα: 4 ώρες
- Κατάσταση: Σταθερή με θετική πρόοδο

Ένα δεύτερο έγγραφο στην ίδια συλλογή θα μπορούσε να είναι ως εξής:

- Id: 10
- Τύπος: Πρώην ασθενής
- Όνομα: Μαρία Χιώνη
- Ηλικία: 29
- Κατάσταση: Εξιτήριο (22/04/2017)

Και ένα τρίτο έγγραφο:

- Id: 47
- Τύπος: Επίσκεψη
- Παραλήπτης επίσκεψης: 45
- Όνομα: Αθανασία Μήλου
- Ημερομηνία: 15/01/2020

Όπως μπορούμε να δούμε τα τρία έγγραφα αν και αντιπροσωπεύουν ευρύτερα το ίδιο πράγμα και ανήκουν στην ίδια συλλογή, διαθέτουν παρόλα αυτά διαφορετικά σε ποσότητα και περιεχόμενο πεδία ανάλογα με τις ανάγκες κάθε φορά, εξοικονομώντας έτσι χώρο στη μνήμη και μειώνοντας την πολυπλοκότητα της βάσης σε σχέση με μια αντίστοιχη σχεσιακή. Μπορούμε να παρομοιάσουμε τον συγκεκριμένο τρόπο οργάνωσης σαν ένα ψηφιακό συρτάρι από έγγραφα, ο οποίος αποτελεί τελικώς έναν πιο φυσικό τρόπο φύλαξης των δεδομένων.

Παράδειγμα αιτήματος σε βάση MongoDb: hospital.visitors.find({ ‘τύπος’ : ‘Επίσκεψη’ });

Τέλος, σημαντικό πλεονέκτημα των “NoSQL” βάσεων είναι ότι αποθηκεύουν τα δεδομένα τους ως αντικείμενα JavaScript (Αρχεία JSON ή -BSON) βοηθώντας έτσι στην συνέχεια όσον αφορά

τον κώδικα των εφαρμογών που χρησιμοποιούν την JavaScript και τα frameworks της και στα υπόλοιπα τμήματα τους, πέραν της βάσης δεδομένων (client side , server side).

Αξίζει να σημειωθεί πως οι βάσεις μη σχεσιακού τύπου δεν είναι εγγυημένο ότι τηρούν πάντοτε το μοντέλο ACID (Ατομικότητα-Συνέπεια-Απομόνωση-Μονιμότητα) που ισχύει στις SQL βάσεις. Θεωρούνται όμως «εντέλει συνεπής» σύμφωνα και με το θεώρημα CAP

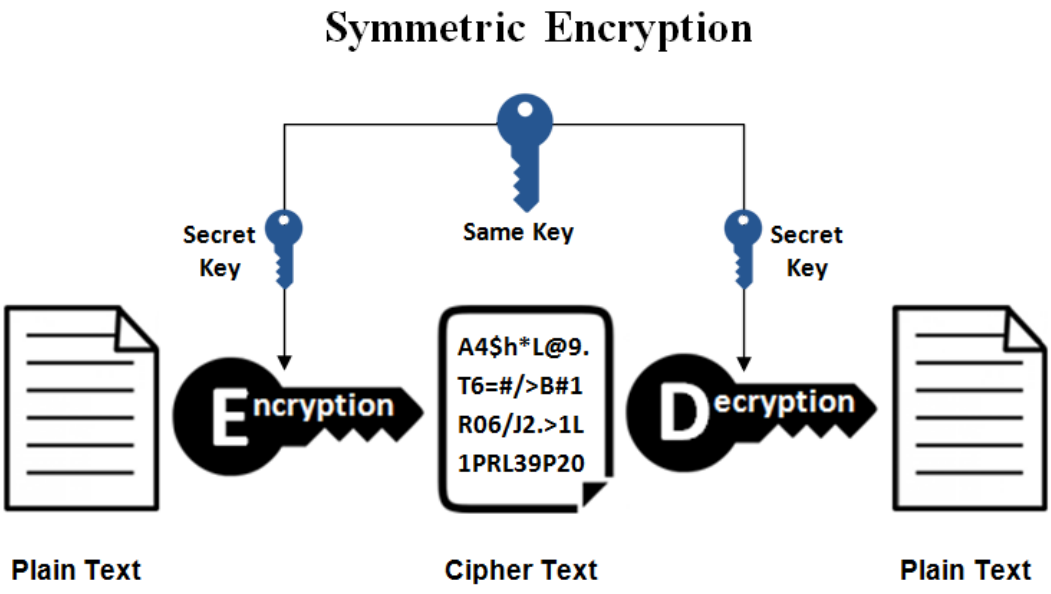
Πηγή - https://www.pliroforiki-edu.gr/unit/ch0202-xaraktiristika-diaxeirisis-basis-dedomenon/

2.8 Ασφάλεια

2.8.1 Ασύμμετρη κρυπτογραφία

Η ασύμμετρη κρυπτογραφία ή κρυπτογραφία δημοσίου κλειδιού, δημιουργήθηκε για να λύσει το πρόβλημα που υπάρχει , όταν χρησιμοποιείται η μέθοδος συμμετρικής κρυπτογραφίας. Κατά την μέθοδο αυτή, ένα μήνυμα κρυπτογραφείται χρησιμοποιώντας μια λέξη κλειδί “passphrase”. Στη συνέχεια, το κρυπτογραφημένο μήνυμα αποστέλλεται στον παραλήπτη ο οποίος μπορεί να το αποκρυπτογραφήσει εφόσον έχει στην κατοχή του τη λέξη κλειδί. Στο σημείο αυτό παρουσιάζεται και το πρόβλημα, το οποίο υπόκειται στο ότι δεν μπορούμε να εγγυηθούμε ότι η λέξη κλειδί θα μεταδοθεί με ασφάλεια προς τον παραλήπτη του μηνύματος, καθώς υπήρχε η πιθανότητα της υποκλοπής του συμμετρικού κλειδιού από τρίτους.

Το διαδίκτυο όμως σήμερα, δεν χρησιμοποιείται πλέον μόνο για την ασφαλή ανταλλαγή

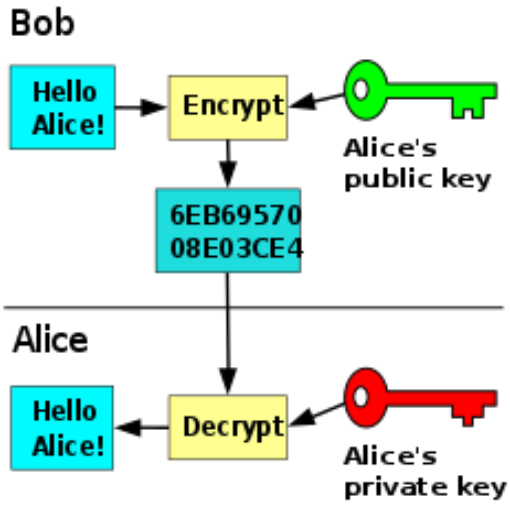


Εικόνα 2.14

μηνυμάτων απόρρητων και μη, χρησιμοποιείται επίσης για την πραγμάτωση πληρωμών και τη μεταφορά χρημάτων κατά μήκος πολλών υπηρεσιών. Ένας νέος τρόπος κρυπτογράφησης

έπρεπε να εφαρμοστεί έτσι ώστε να εγγυηθεί την ανταλλαγή του συμμετρικού κλειδιού με πραγματική ασφάλεια. Έτσι λοιπόν εισάχθηκε ή μέθοδος κρυπτογραφίας ασύμμετρου κλειδιού ως βάση της ασφαλούς επικοινωνίας στο διαδίκτυο. Αυτή, καθώς και άλλα στρώματα ασφάλειας όπως ο κατακερματισμός των δεδομένων, οι ψηφιακές υπογραφές-πιστοποιητικά και η φύλαξη των διαφόρων τμημάτων των εφαρμογών σε διαφορετικούς ανεξάρτητους σε λειτουργία «χώρους», χρησιμοποιούνται από την συντριπτική πλειοψηφία των εφαρμογών στο σύγχρονο παγκόσμιο ιστό.

Η κρυπτογραφία ασύμμετρου κλειδιού περιγράφεται ως εξής, έστω ότι επιθυμούμε να μπορούμε να στείλουμε και να μας στείλουν πληροφορίες με ασφάλεια. Τόσο εμείς όσο και ο κάθε χρήστης της τεχνολογίας αυτής, έχουμε στην κατοχή μας ένα μαθηματικά συνδεδεμένο ζεύγος κλειδιών. Επιλέγουμε το ένα κλειδί να είναι το ιδιωτικό μας κλειδί, το οποίο θα παραμένει στην κατοχή μας και μόνο, ενώ το δεύτερο λέμε ότι είναι το δημόσιο κλειδί μας το οποίο καθιστούμε ορατό σε όλους. Πλέον, όποιος επιθυμεί να μας στείλει κάποιο μήνυμα μπορεί να χρησιμοποιήσει το δημόσιο κλειδί μας για να το κρυπτογραφήσει, με αυτόν τον τρόπο, μόνο εμείς θα έχουμε τη δυνατότητα αποκρυπτογράφησης του μηνύματος χρησιμοποιώντας το ιδιωτικό μας κλειδί.



Εικόνα 2.15

Το ιδιωτικό και το δημόσιο κλειδί, έχουν παραχθεί μέσα από ένα πρόγραμμα που κάνει χρήση του μαθηματικού αλγόριθμου(RSA), θα μπορούσαμε να πούμε ότι σαν γενική ιδέα ο RSA εκμεταλλεύεται τη δυσκολία σε χρονική πολυπλοκότητα της μαθηματικής πράξης υπολογισμού της συνάρτησης Euler(Συνάρτηση Φ).

Ας δούμε συνοπτικά τα βήματα του αλγορίθμου παράλληλα με ένα παράδειγμα.

1) Επιλογή 2 πρώτων αριθμών p,q (για παράδειγμα p=2, q=7)

2) Υπολογισμός γινομένου N=p*q (N=14)
3) Υπολογισμός της συνάρτησης Φ(N) (Πλήθος αριθμών μικρότερων του 14 που δεν μοιράζονται κανένα κοινό διαιρέτη πέραν της μονάδας με το 14. Έχοντας ως δεδομένο ότι τα p και q είναι πρώτοι αριθμοί αυτό μπορεί να υπολογιστεί από τον τύπο Φ(N) = (p-1)*(q-1) , που στη δική μας περίπτωση ισούται με Φ(N)=6)

4) Επιλογή ενός αριθμού, έστω e από τη λέξη encryption, με τις εξής προϋποθέσεις

- 1 < e < N
- e δεν έχει κοινούς διαιρέτες πέρα της μονάδας με τα N και Φ(N)

Το ζεύγος αριθμών (e, N) είναι και αυτό που είναι δημόσιο σε οποιονδήποτε ενδιαφέρεται να μας στείλει κάποιο μήνυμα αποτελεί δηλαδή το δημόσιο μας κλειδί (στην προκειμένη περίπτωση είναι το ζεύγος (5,14).

5) Επιλογή ενός αριθμού, έστω d από τη λέξη decryption τέτοιου ώστε να ισχύει: e*d(modΦ(N)) = 1 (στη δική μας περίπτωση το 5*d(mod6)= 1, . Μια ορθή επιλογή θα ήταν ο αριθμός d=11, καθώς (5*11)mod6 = 1)

Το ζεύγος αριθμών (d,N), είναι αυτό που θα χρησιμοποιηθεί για την αποκρυπτογράφηση του μηνύματος και συγκεκριμένα ο αριθμός d αποτελεί το ιδιωτικό μας κλειδί. Αν κρυπτογραφήσουμε ένα μήνυμα με το δημόσιο κλειδί ενός παραλήπτη, τότε αυτό μπορεί να αποκρυπτογραφηθεί μόνο από το ιδιωτικό κλειδί του παραλήπτη το οποίο δεν διαμοιράζεται και άρα η επικοινωνία είναι ασφαλής.

Αν θέλαμε λοιπόν να κρυπτογραφήσουμε ένα απλό μήνυμα, για παράδειγμα το γράμμα 'B', θα έπρεπε αρχικά να το αντιστοιχίσουμε με μια αριθμητική τιμή, έστω m, συνήθως η μετατροπή γίνεται σε ASCII κώδικα. Για λόγους ευκολίας των πράξεων θα θεωρήσουμε ότι στη προκειμένη περίπτωση 'B'=2, καθώς πρέπει m < N (μεγάλα μηνύματα στέλνονται σε πολλαπλά τμήματα). Στη συνέχεια κρυπτογραφούμε χρησιμοποιώντας το δημόσιο κλειδί εκτελώντας την πράξη c=m^e(modN) , όπου c το κρυπτογραφημένο μήνυμα. Για εμάς θα ισχύει: c=2^5(mod14) δηλαδή c=4, ο παραλήπτης του μηνύματος μπορεί έπειτα να αποκρυπτογραφήσει το c, αποκαλύπτοντας έτσι το περιεχόμενο του αρχικού μηνύματος m, εκτελώντας την πράξη m = c^d (modN) = 4^11(mod14) = 4.194.304mod14 ≡ 2. Πράγματι η αρχική τιμή του μηνύματος μας ήταν το 2.

<u>Παρατηρήσεις</u>:

- Σε πραγματικές συνθήκες οι αριθμοί p και q απαιτείται πολύ μεγάλοι, δηλαδή εκατοντάδες ψηφία σε μήκος. Θυμηθείτε ότι το ιδιωτικό κλειδί d, είναι ο αριθμός που απαιτείται για την αποκρυπτογράφηση οποιουδήποτε μηνύματος είχε κρυπτογραφηθεί χρησιμοποιώντας το αντίστοιχο δημόσιο κλειδί, ο αριθμός αυτός όπως είδαμε προηγουμένως προκείμενου να υπολογιστεί απαιτεί την γνώση της συνάρτησης Euler (Φ(N)), δηλαδή του πλήθους όλων των αριθμών από το 1 έως το N (γινόμενο p*q), που δεν έχουν κοινό διαιρέτη με το N, πέραν της μονάδας. Ο υπολογισμός αυτός, για

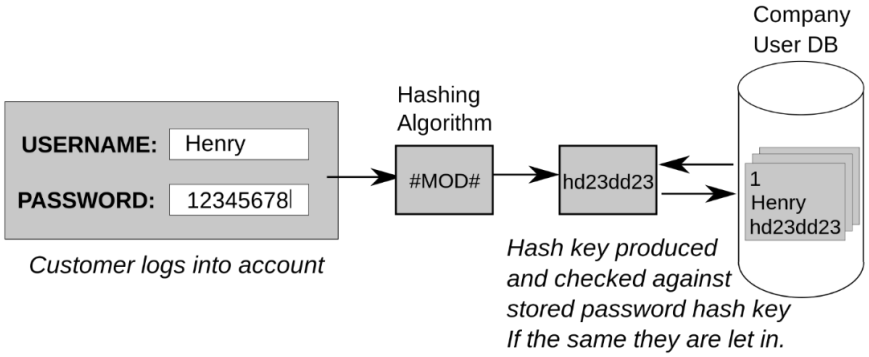
αριθμούς τέτοιου μήκους είναι προς το παρόν τόσο χρονοβόρος ώστε να θεωρείται απλά ανέφικτος, ακόμα και από τους ισχυρότερους υπολογιστές και κατανεμημένα συστήματα της εποχής μας. Σε περίπτωση λοιπόν υποκλοπής, έχοντας στη κατοχή του μόνο το κρυπτογραφημένο κείμενο και τις τιμές του δημοσίου κλειδιού e και N, ο υποκλοπέας είναι πρακτικά αδύνατον, να υπολογίσει το Φ(N) και συμπερασματικά το ιδιωτικό κλειδί του πραγματικού παραλήπτη ώστε να δει τις πληροφορίες σε ευανάγνωστη μορφή.

- Η συμμετρική κρυπτογραφία χρησιμοποιείται ακόμα στο πρωτόκολλο https. Η ασύμμετρη κρυπτογραφία χρησιμοποιείται κατά κύριο λόγο για την ασφαλή ανταλλαγή του συμμετρικού κλειδιού ανάμεσα στα συστήματα που επικοινωνούν.

2.8.2 Αλγόριθμοι κατακερματισμού (Hashing)

Σε περιπτώσεις ταυτοποίησης ή σύγκρισης δεδομένων, όπως για παράδειγμα σε ένα σύστημα 'login', θα ήταν εξαιρετικά χρήσιμο να αποφεύγαμε εντελώς την αποστολή αυτών καθαυτών των δεδομένων και να αποστέλλαμε αντί αυτού, μια κατακερματισμένη τιμή τους (hashed value), ή οποία, σε περίπτωση υποκλοπής θα είναι πρακτικά αδύνατον να οδηγήσει στα αρχικά δεδομένα. Παράλληλα, καλύπτεται και ένα ζήτημα ιδιωτικότητας, καθώς πλέον δεν απαιτείται ούτε από τους δημιουργούς των εφαρμογών η αποθήκευση των κωδικών των χρηστών τους. Για να πραγματοποιήσουμε τα παραπάνω χρησιμοποιούμε διάφορους αλγόριθμους κατακερματισμού οι οποίοι, εκτελούν πρακτικά μη αντιστρέψιμους από οποιοδήποτε σύστημα της εποχής μας μετασχηματισμούς στα δεδομένα που παίρνουν ως είσοδο. Αξίζει να σημειωθεί ότι με την πάροδο του χρόνου και όσο τα υπολογιστικά συστήματα γίνονται ολοένα και πιο ισχυρά, πολλοί από τους αλγορίθμους αυτούς που στο παρελθόν αποτελούσαν προτιμώμενες λύσεις, καταστάθηκαν παρωχημένοι, καθώς η έξοδός τους (digest), μπορούσε πλέον να οδηγήσει στα αρχικά δεδομένα.

Παραδείγματα τέτοιων αλγορίθμων είναι οι: MD5, SHA0, SHA1, ενώ άλλοι που χρησιμοποιούνται όμως ακόμη και σήμερα είναι οι αλγόριθμοι: SHA2, SHA3



Εικόνα 2.16 Παράδειγμα χρήσης αλγόριθμων κατακερματισμού σε σύστημα εισόδου login

2.8.3 Ψηφιακή υπογραφή και πιστοποιητικά

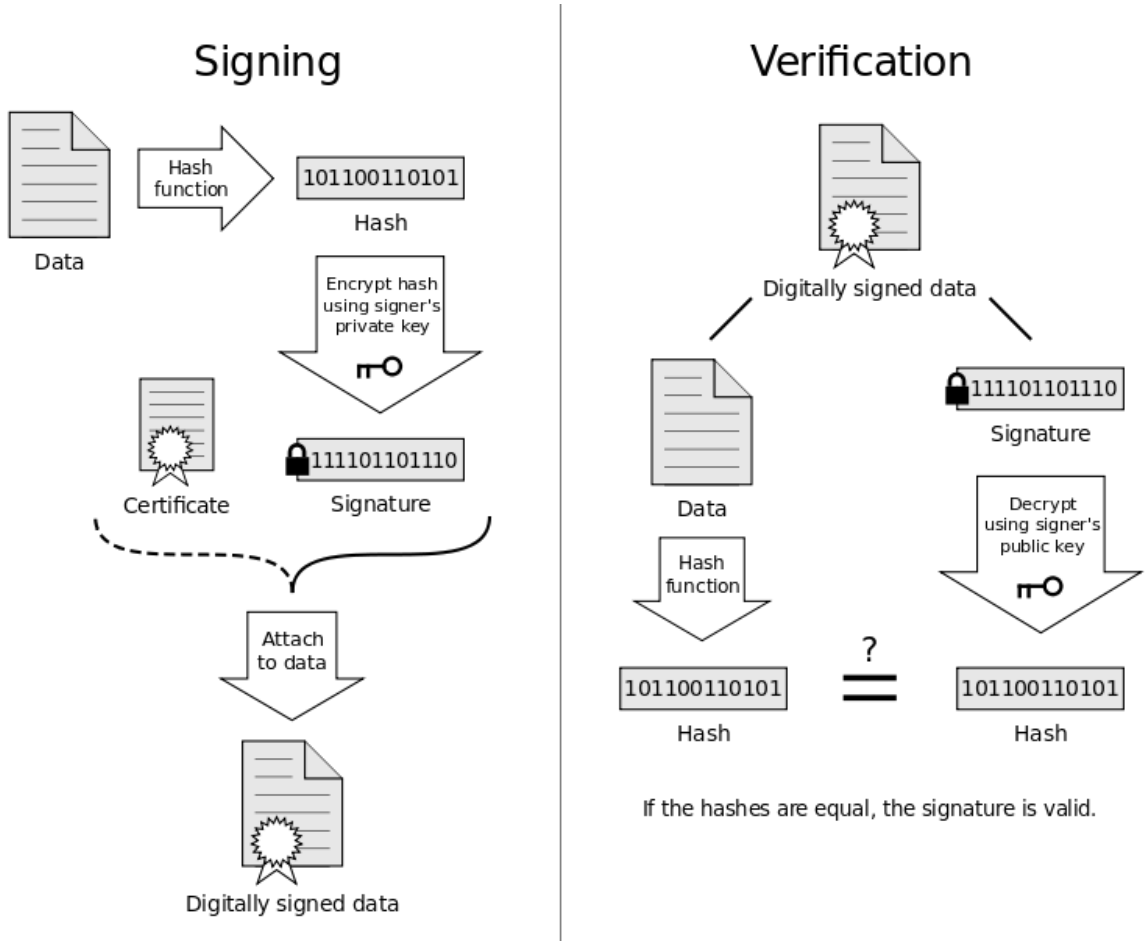
Η ασύμμετρη κρυπτογραφία μας εξασφαλίζει ότι τα δεδομένα μας θα μπορούν να χρησιμοποιηθούν μόνο από τον προοριζόμενο παραλήπτη. Αυτό που ακόμη δεν εξασφαλίζεται, είναι η ακεραιότητα των δεδομένων κατά τη μεταφορά τους καθώς και το αν ο αποστολέας του μηνύματος είναι πραγματικά αυτός που ισχυρίζεται.

Τα παραπάνω μπορούν να εξασφαλιστούν με τη χρήση των πιστοποιητικών, δεδομένου ότι έχουμε στην κατοχή μας το πιστοποιητικό του πραγματικού αποστολέα κάθε φορά. Αρχικά, ο αποστολέας στέλνει το έγγραφο το οποίο ονομάζουμε πιστοποιητικό όπου εμπεριέχονται κάποιες βασικές πληροφορίες για αυτόν, όπως το όνομα, η διεύθυνση του συστήματος του στο διαδίκτυο (domain name), το δημόσιο κλειδί του, καθώς και η ψηφιακή υπογραφή του. Η ψηφιακή υπογραφή είναι ένα μήνυμα μοναδικό για τον κάθε χρήστη το οποίο έχει αρχικά κατακερματιστεί με κάποιον αλγόριθμο κατακερματισμού (Συνήθως SHA256) και στη συνέχεια κρυπτογραφηθεί, χρησιμοποιώντας το ιδιωτικό κλειδί του υπογράφοντος.

Έστω ότι την πρώτη φορά η αποστολή του πιστοποιητικού γίνεται με τρόπο τέτοιο ώστε να μας εξασφαλίζει ότι ο κάτοχος του πιστοποιητικού είναι πραγματικά αυτός που ισχυρίζεται. Για παράδειγμα, θα μπορούσαμε να φανταστούμε το σενάριο ότι το πιστοποιητικό εγκαθίσταται αρχικά στο σύστημα μας αυτοπροσώπως από τον πραγματικό αποστολέα με τη χρήση ενός usb δίσκου. Από αυτό το σημείο και έπειτα, οποιοσδήποτε άλλος επιθυμεί να επικοινωνήσει μαζί μας θα απαιτούμε πρώτα να μας αποστέλλει το πιστοποιητικό που αποδεικνύει την ταυτότητα του. Αν λοιπόν κάποιος ισχυρίζεται ότι είναι ο πραγματικός αποστολέας, αυτό μπορεί να εξακριβωθεί αποκρυπτογραφώντας την ψηφιακή υπογραφή που εμπεριέχεται στο πιστοποιητικό του κάνοντας χρήση του δημόσιου κλειδιού του πραγματικού αποστολέα και στη συνέχεια συγκρίνοντας τις ψηφιακές υπογραφές ή για την ακρίβεια τις κατακερματισμένες τιμές αυτών (αυτή του πιστοποιητικού που μας εστάλη με εκείνη που αναγράφεται στο γνήσιο πιστοποιητικό που είναι εγκατεστημένο στο σύστημα μας) . Αν οι κατακερματισμένες τιμές είναι ίσες, τότε μπορούμε να είμαστε σίγουροι πως η ταυτότητα του αποστολέα είναι αληθής αφού μόνο το δικό του δημόσιο κλειδί μπορεί να μας δώσει ακριβώς την κατακερματισμένη τιμή ή "digest" της ψηφιακής υπογραφής του. Επιπλέον μπορούμε να είμαστε σίγουροι, πως εντός του καναλιού επικοινωνίας που εδραιώσαμε, τα δεδομένα παραμένουν ακέραια, καθώς η παραμικρή αλλαγή θα άλλαζε σε μεγάλο βαθμό και την κατακερματισμένη τιμή που προηγουμένως ελέγξαμε.

Σε ρεαλιστικές συνθήκες βέβαια, δεν υπάρχει η δυνατότητα να ελέγχουμε την προέλευση των πιστοποιητικών που έχουμε εγκαταστήσει εξακριβώνοντας αυτοπροσώπως την ταυτότητα του κάθε αποστολέα. Για το λόγο αυτό όλοι οι σύγχρονοι περιηγητές έχουν προ εγκατεστημένα τα πιστοποιητικά, όλων των «αξιόπιστων» φορέων, τα οποία μάλιστα είναι υπογεγραμμένα (μέθοδος ψηφιακής υπογραφής), από διάφορους εξ' ορισμού αξιόπιστους παρόχους πιστοποιητικών, που ονομάζονται «αρχές πιστοποίησης» ή "CA" (Certification Authority). Μπορούμε να εμπιστευόμαστε τις αρχές πιστοποίησης καθώς προκειμένου να υπογράψουν ένα

πιστοποιητικό, εκτελούν πρώτα έλεγχο προς εξακρίβωση της ταυτότητας που ισχυρίζεται ο εκάστοτέ ενδιαφερόμενος.



Εικόνα 2.17

2.8.4 Πρωτόκολλο HTTPS

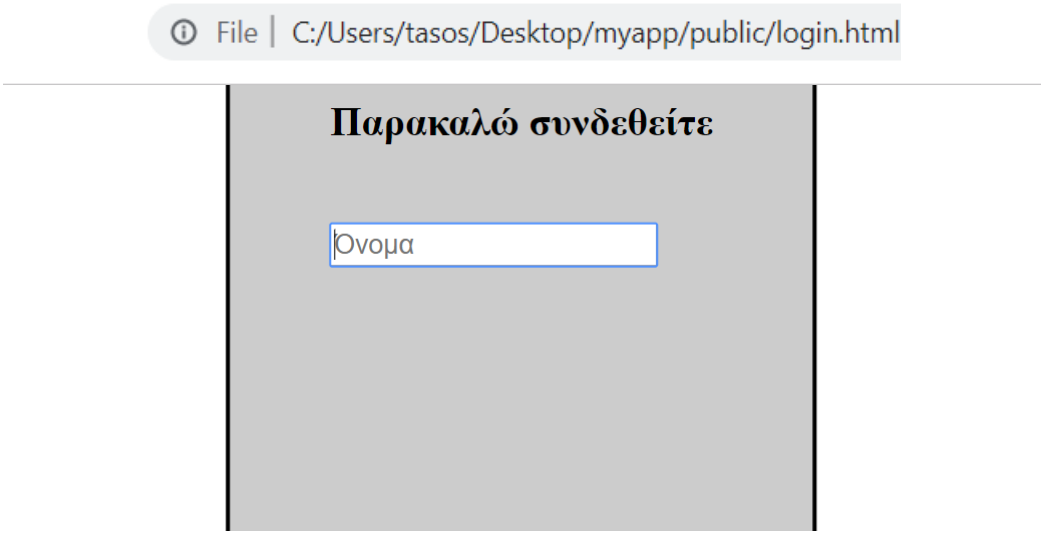
Όλες οι προαναφερθέντες διεργασίες καθώς και άλλες που δεν θα αναλυθούν στην παρούσα εργασία, αποτελούν μέρος μιας διαδικασίας που ονομάζεται “SSL handshake”, η οποία εκτελείται κάθε φορά που το σύστημα μας «θέλει» να επικοινωνήσει με ασφάλεια με άλλα συστήματα στο διαδίκτυο. Το πρωτόκολλο HTTP, εκτεταμένο με τα μέτρα ασφαλείας SSL δομούν το ασφαλές πρωτόκολλο επικοινωνίας HTTPS.

Κεφάλαιο 3ο – Περιγραφή λειτουργικότητας και πηγαίος κώδικας

3.1 Οθόνη εισόδου

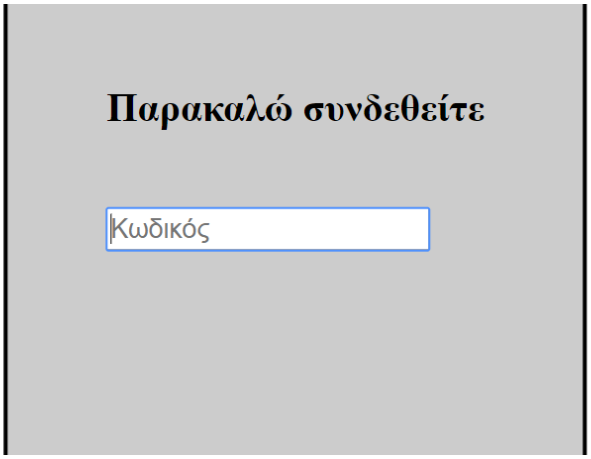
3.1.1 - Σύνοψη λειτουργίας από την οπτική του χρήστη

Η οθόνη εισόδου, είναι το πρώτο πράγμα που βλέπουν οι χρήστες όταν εκτελείται η εφαρμογή. Αποτελείται από ένα μια απλή φόρμα με μία είσοδο και μια επικεφαλίδα, ενώ η καταχώρηση (submit), της φόρμας γίνεται απλά πατώντας το πλήκτρο enter.



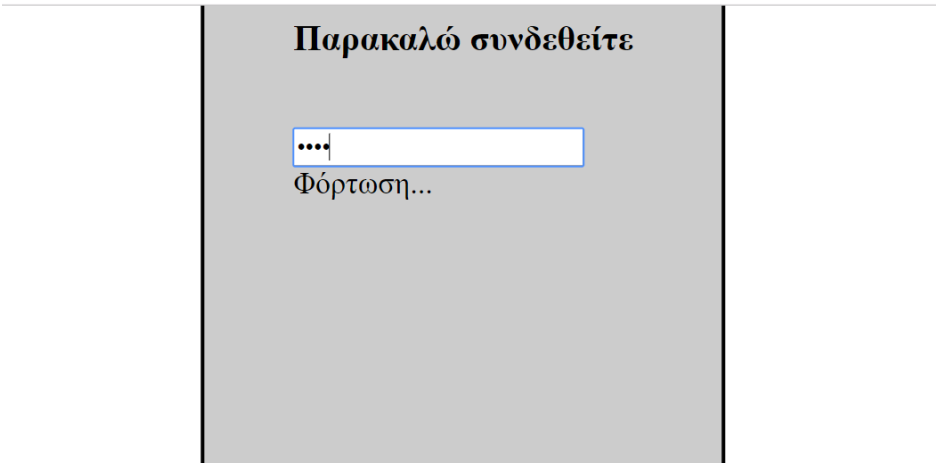
Εικόνα 3.1

Εφόσον πληκτρολογήσουμε το όνομα χρήστη και πατήσουμε το πλήκτρο enter η φόρμα αποθηκεύει προσωρινά την τιμή του ονόματος και αλλάζει μορφή περιμένοντας πλέον το συνθηματικό μας.



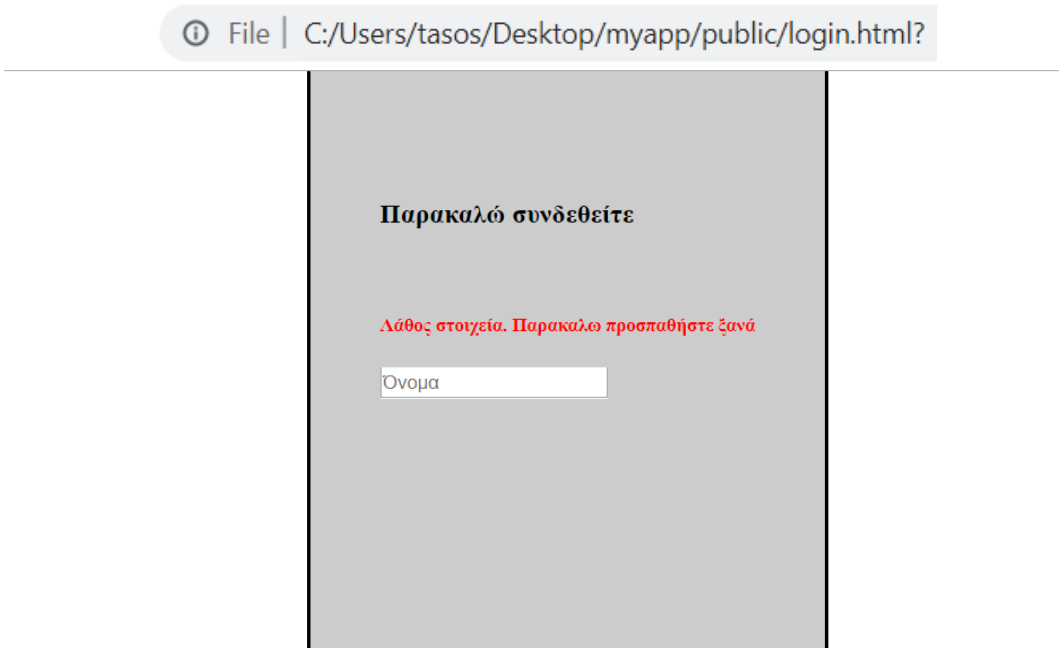
Εικόνα 3.2

Έπειτα από την εισαγωγή των στοιχείων εισόδου μας, πραγματοποιείται ο έλεγχος και η επικοινωνία με το διακομιστή και τη βάση δεδομένων.



Εικόνα 3.3 Καταχώρηση κωδικού και αίτημα ajax

Σε περίπτωση εισαγωγής λανθασμένων στοιχείων, η εφαρμογή επιστρέφει στην αρχική οθόνη, ενημερώνοντας το χρήστη για την αποτυχία σύνδεσης. Η διαδικασία στη συνέχεια μπορεί να επαναληφθεί.



Εικόνα 3.4 Λανθασμένα στοιχεία, μήνυμα λάθους και χαρακτήρας '?' στο url

Σε περίπτωση εισαγωγής, σωστού ζεύγους στοιχείων η εφαρμογή περνάει στο επόμενο βήμα λειτουργίας της δηλαδή την κεντρική οθόνη.

3.1.2 – Κώδικας οθόνης εισόδου

Κώδικας: HTML

Ο κώδικας HTML είναι σχετικά σύντομος.

```
<!DOCTYPE html>
<html>
    <head>
        <script src="https://cdn.jsdelivr.net/npm/vue"></script>
        <script src="https://cdnjs.cloudflare.com/ajax/libs/axios/0.18.0/axios.js"></script>
        <title>login</title>
        <meta charset="utf-8">
        <link rel="stylesheet" type="text/css" href="stylesheets/login.css">
    </head>
    <body>
        <div id="root" class="center">
            <h3>Παρακαλώ συνδεθείτε</h3> <br>
            <h5 v-if="wrongcre" style="color:red">Λάθος στοιχεία. Παρακαλω προσπαθήστε ξανά</h5>
            <input v-model='userinput' v-bind:type="type" v-bind:placeholder="holder" v-on:keyup.enter="getinput">
            <br><span v-if="ajaxRequest">Φόρτωση...</span>
        </div>
    </body>
</html>
```

Εικόνα 3.5

Στο “head” tag, συμπεριλαμβάνονται τα frameworks των Vue js και Axios js. Για τη Vue, έχουμε ήδη μιλήσει στο δεύτερο κεφάλαιο.. Το Axios, είναι ένα framework της javascript που χρησιμοποιήθηκε ώστε να γίνει πιο ευανάγνωστη και εύκολη η σύνταξη των αιτημάτων Ajax (Asynchronous request) , προς το διακομιστή.

Στο “body” tag εντοπίζονται, το κείμενο «Παρακαλώ συνδεθείτε» καθώς, τη φόρμα εισόδου <input> με διάφορα “attributes”, όπως το “v-on:keyup.enter” που καλεί την συνάρτηση “getinput” και τα “v-bind:placeholder και v-bind:type” που συσχετίζουν τα html attributes “placeholder” και “type” με τις αντίστοιχες μεταβλητές “placeholder” και “type” που έχουν οριστεί στο αντικείμενο Vue.

Κώδικας: Vue

```
app = new Vue({
    el: '#root',
    data: {
        userinput: '',
        type: 'text',
        holder: 'Όνομα',
        username: '',
        password: '',
        ajaxRequest: false,
        wrongcre: false,
    postResults: []
    },
    methods: {
        getinput : function(){
            if(this.userinput.length < 2)
                return
            else if(this.type == 'text'){
                this.wrongcre = false;
                this.username = this.userinput
                this.userinput = ''
                this.type = 'password'
                this.holder = 'Κωδικός'
            }
            else if(this.type == 'password'){
                this.password = this.userinput
                this.ajaxRequest = true;
                axios.post('/public/requestlogin/', {usrn : this.username, pass : this.password})
              .then(function (response) {
                if(response.data.answer == "kick"){
                    window.location.href = response.data.page + ".html?";
                }
                else
                    window.location.href = response.data.page + ".html";

              })
              .catch(function (error) {
                // handle error
                console.log(error);
              });
            }

        }
    },
    created : function(){
        if(window.location.href.includes('?'))
            this.wrongcre = true;
    }
```

Εικόνα 3.6

Στον κώδικα Vue, αρχικά συναντάμε τη μεταβλητή el, η οποία πάντα παίρνει την τιμή id του html tag στο οποίο θέλουμε να έχει ισχύ ο κώδικας. Στη συνέχεια συναντάμε το component “data”. Μέσα στο data, ορίζουμε τις μεταβλητές που επιθυμούμε να έχουμε πρόσβαση από οπουδήποτε (global μεταβλητές). Εδώ αρχικοποιούμε τη μεταβλητή “userinput” στην οποία εκχωρούνται οι τιμές που δίνει ως είσοδο ο χρήστης λόγο της συσχέτισης v-model στον κώδικα html.

Άλλες μεταβλητές είναι οι type και placeholder, με αρχικές τιμές, “text ” και “Όνομα” αντίστοιχα. Όπως αναφέραμε προηγουμένως οι τιμές αυτές εισχωρούντα στα αντίστοιχα πεδία type και placeholder στην html κάνοντας το πεδίο input να έχει προεπιλεγμένη τιμή το αλφαριθμητικό «Όνομα» και ο τύπος του να είναι “text”, να αναμένει δηλαδή ως είσοδο κανονικό κείμενο.

Έπειτα δημιουργούμε τις μεταβλητές username και password όπου θα φιλοξενήσουν τα στοιχεία που θα δώσει ως είσοδο ο χρήστης και αρχικά έχουν κενή τιμή.

Τέλος συναντάμε και άλλες δύο μεταβλητές τύπου Boolean, που ονομάσαμε “ajaxRequest” και “wrongcre” με αρχική τιμή “false” τις οποίες, χρησιμοποιούμε για να ελέγξουμε αν εκτελείται ένα ajax αίτημα και αν ο χρήστης έδωσε λάθος στοιχεία αντίστοιχα.

Στο component “methods” έχουμε ορίσει μόνο μία μέθοδο, την getinput, Η getinput καλείται κάθε φορά που πατάμε enter και η λειτουργία της περιγράφεται ως εξής:

Αρχικά ελέγχουμε αν η είσοδος που δόθηκε από το χρήστη είναι μικρότερη από δύο χαρακτήρες σε μήκος, αν η συνθήκη είναι αληθής, τότε η συνάρτηση τερματίζεται και δεν έχουμε κάποιο ορατό αποτέλεσμα. Σε διαφορετική περίπτωση συνεχίζουμε με την εκτέλεση του υπόλοιπου κώδικα.

Στη συνέχεια κάνουμε έλεγχο σχετικά με τον τύπο του πεδίου input της html, ελέγχοντας το περιεχόμενο της μεταβλητής type.

Αν το type είναι ίσο με ‘text’ σημαίνει ότι μόλις λάβαμε το Όνομα χρήστη και άρα θέτουμε την τιμή της μεταβλητής username να είναι ίση με userinput δηλαδή με την είσοδο που λάβαμε από το χρήστη. Ακόμη θέτουμε την μεταβλητή type να είναι ίση με ‘password’, ώστε να αλλάξει ο τύπος εισόδου στο πεδίο input της html και να αναμένει πλέον τον κωδικό του χρήστη. Τέλος θέτουμε και το προεπιλεγμένο κείμενο του input (placeholder), σε “Κωδικός”, ώστε να δώσουμε και οπτική ανατροφοδότηση στο χρήστη.

Αν το type είναι ίσο με “password” σημαίνει ότι η είσοδος που μόλις έχουμε λάβει είναι ο κωδικός χρήστη και άρα εισχωρούμε την τιμή του κωδικού αυτού στη μεταβλητή password. Πλέον έχουμε στη κατοχή μας τα απαραίτητα στοιχεία ώστε να τα συγκρίνουμε με αυτά στη βάση δεδομένων. Εκτελούμε ένα αίτημα ajax, όπως φαίνεται στον κώδικα, στο εικονικό μονοπάτι ‘/public/requestlogin/’ μεταφέροντας τα δεδομένα username και password ως usrn και pass στο ενδιάμεσο λογισμικό που «ακούει» για αιτήματα στο συγκεκριμένο μονοπάτι.

Εφόσον λάβουμε απάντηση από το διακομιστή, αποφασίζουμε αν θα κάνουμε ανακατεύθυνση στην κεντρική σελίδα ή, σε περίπτωση λανθασμένων στοιχείων, ξανά στη στην ίδια σελίδα εισόδου (βλέπε Εικόνα 3.6 στο: “window.location.href = response.data.page + “html”), ενώ στο σενάριο ανεπιτυχούς προσπάθειας εισόδου προσθέτουμε και ένα “?” επιπλέον στη διεύθυνση url.

Τέλος ας δούμε και τη λειτουργία ενός τρίτου component της Vue.. Το “created” component, εκτελεί τον κώδικα που βρίσκεται εντός αυτού πριν το αντικείμενο vue προλάβει να εκτελέσει

οποιαδήποτε άλλη ενέργεια. Το component αυτό καθώς και άλλα που θα δούμε αργότερα ανήκουν στο «χρονοδιάγραμμα ζωής» του αντικειμένου Vue και μας επιτρέπουν να προγραμματίσουμε τη σειρά ενεργειών με την οποία θα εκτελείται ο κώδικας μας, σε διάφορα στάδια της δημιουργίας και επεξεργασίας του DOM. (Περισσότερα για το vue lifecycle στο: https://vuejs.org/v2/guide/instance.html).

Έτσι λοιπόν καταλαβαίνουμε ότι το πρώτο πράγμα που εκτελείται στον παραπάνω κώδικα είναι ο έλεγχος που θα μας υποδείξει αν στο url συμπεριλαμβάνεται το σύμβολο "?"· Έτσι, μπορούμε να γνωρίζουμε αν προηγουμένως είχε συμβεί ανακατεύθυνση εξαιτίας λανθασμένων στοιχείων και στην περίπτωση αυτή, να θέσουμε την τιμή της μεταβλητής "wrongcre" σε αληθής (true) για να έχουμε και το αντίστοιχο μήνυμα λάθους, το οποίο εμφανίζεται όταν η συνθήκη v-if="= "wrongcre" είναι αληθής (Εικόνες 3.4 και 3.5).

Κώδικας: Ενδιάμεσο λογισμικό - Express js

```
var express = require('express');
const crypto = require('crypto');
var bend = require('../public/javascripts/backend');
var router = express.Router();
var app = express();

router.post('/public/requestlogin/', function(req, res){
    //Password hashing to compare to hashed password stored on the db, usage of one way hashing algorithm
    let hashpass = crypto.createHash('sha256').update(req.body.pass).digest('hex');

    bend.login(req.body.usrn, hashpass).then((value) => {
        if(value){
            req.session.user = value.username;
            req.session.role = value.role;
            res.send({page : "waiter", answer : "ok"});
        }
        else{
            res.send({page : "login", answer : "kick"});
        }

    });

});
```

Εικόνα 3.7

Ο παραπάνω δρομολογητής ενεργοποιείται όταν εκτελούμε από ένα ajax post αίτημα στη διεύθυνση '/public/requestlogin/'. Ο κώδικας που εκτελείται είναι ως εξής:

Αρχικά χρησιμοποιούμε μια συνάρτηση της βιβλιοθήκης crypto (node js module), για να εφαρμόσουμε τον αλγόριθμο κατακερματισμού SHA256, στον κωδικό χρήστη που περάσαμε προηγουμένως ως όρισμα με όνομα “pass”. Στη συνέχεια καλούμε τη συνάρτηση login που εμπεριέχεται στο αρχείο backend.js μαζί με την υπόλοιπη λειτουργικότητα του διακομιστή, στην οποία περνάμε ως όρισμα το όνομα χρήστη, καθώς και το hash value του κωδικού. Τέλος ανάλογα με τη επιστρεφόμενη τιμή της συνάρτησης login, στέλνουμε στον πελάτη και το αντίστοιχο όνομα του αρχείου html στο οποίοι θα εκτελέσει ανακατεύθυνση, αλλά και ένα μήνυμα ως απάντηση (“ok” ή “kick”), ενώ σε περίπτωση επιτυχούς σύνδεσης, αναθέτουμε επίσης στον πίνακα της συνόδου που έχουμε δημιουργήσει (session cookie), τις τιμές του “username” και “role” που επεστράφησαν από τη συνάρτηση login.

Προτιμούμε να στείλουμε το όνομα του αρχείου, μέσω μιας μεταβλητής στο front-end και να μην το έχουμε σε μορφή απλού κειμένου, καθώς δεν θέλουμε να είναι εμφανής η δομή του directory του διακομιστή στον κώδικα συσκευής πελάτη.

Κώδικας: Διακομιστής - node js

```
const MongoClient = require('mongodb').MongoClient;

exports.login = function(usrn, password){
    const dbpromise = new Promise((resolve, reject) => {
        var url = "mongodb://localhost:27017/";
        const client = new MongoClient(url, { useNewUrlParser: true });

        client.connect((err => {
            const collection = client.db("verna").collection("team");
            collection.findOne({ $and: [ {name:usrn}, {pass:password} ] }, (err, result) => {
                if(err) throw err;
                let username = null;
                let password = null;
                let role = null;
                if(result != null){
                    username = result.name;
                    password = result.pass;
                    role = result.role;
                }
                else{
                    resolve(null);
                }
                if (username && password && role) resolve({username, password, role});
                else reject({Error: 'something went wrong'});
            });
            client.close();
        }));

    });
    dbpromise.then((err, result) => {
        if (err) console.log(err);
            return result;
    });

    return dbpromise;
};
```

Εικόνα 3.8

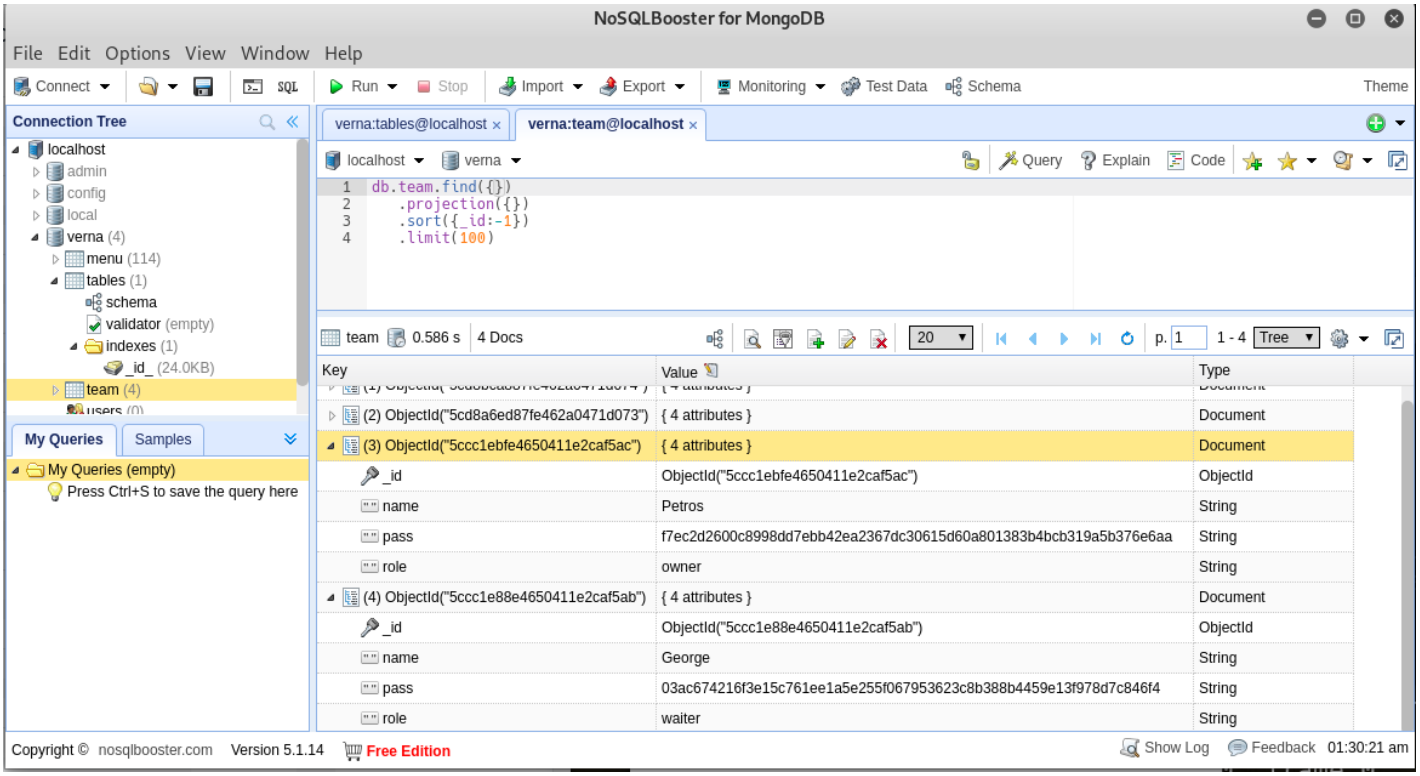
Στην προηγούμενη εικόνα φαίνεται το τμήμα κώδικα που είναι υπεύθυνο για τον έλεγχο στην πλευρά του διακομιστή, των στοιχείων που παρείχε ο χρήστης, με τα αντίστοιχα πεδία στη βάση δεδομένων. Στον κώδικα αρχικά βλέπουμε την σύνδεση στη βάση δεδομένων mongoDb, που φιλοξενείται στο σύστημα μας αφού αποτελεί και τον διακομιστή (εξού και η διεύθυνση localhost). Χρησιμοποιούμε το αντικείμενο client για να συνδεθούμε στη βάση που έχουμε ονομάσει“verna” και να επιλέξουμε τη συλλογή που μας ενδιαφέρει. Στην προκειμένη περίπτωση, οι χρήστες είναι αποθηκευμένοι στη συλλογή “team”. Στη συνέχεια εκτελούμε ένα αίτημα στη βάση το οποίο μας επιστρέφει μια καταχώρηση από τη συλλογή, που θα έχει το όνομα και τον κωδικό που έχουμε ως όρισμα, αν αύτη βέβαια υπάρχει.

Να επισημάνουμε σε αυτό το σημείο ότι ο κώδικας node js εκτελείται ασύγχρονα όταν εκτελεί ορισμένες ενέργειες, όπως επικοινωνία με μια βάση δεδομένων. Εξαιτίας αυτού, υπάρχει η πιθανότητα, η συνάρτηση να επιστρέψει τις μεταβλητές που προορίζονται ως επιστρεφόμενες , δίχως αυτές όμως να έχουν πάρει ακόμα τιμή. Για το λόγο αυτό χρησιμοποιούμε δομές που στη node ονομάζονται “promises”. Οι δομές αυτές σε γενικές γραμμές θα λέγαμε ότι είναι απλά τιμές που επιστρέφονται από ασύγχρονες συναρτήσεις. Τις χρησιμοποιούμε παράλληλα με τα “callbacks” ώστε να ελέγξουμε τη σειρά εκτέλεσης των συναρτήσεων μας. Στον κώδικα ορίζουμε μια ασύγχρονη συνάρτηση const dbpromise = new Promise((resolve, reject) => …

Μετά το πέρας της εκτέλεσης της παραπάνω συνάρτησης, τρία πράγματα δύναται να συμβούν, είτε το αίτημα θα εκτελεστεί και θα λάβουμε μια απάντηση, άρα θα έχουμε μια ανακλημένη τιμή (promise resolve) ,είτε να συμβεί κάποιο σφάλμα διακομιστή (promise reject). Εφόσον έχει βρεθεί χρήστης με τα στοιχεία που μας δόθηκαν τότε αναθέτουμε τις τιμές (username, password και role), σε τοπικές μεταβλητές που δημιουργήσαμε για τον σκοπό αυτό. Αν όλες οι τιμές έχουν ανατεθεί με επιτυχία, τότε κάνουμε resolve τις τιμές αυτές και εξερχόμαστε από την ασύγχρονη συνάρτηση. Σε διαφορετική περίπτωση επιστρέφουμε την τιμή null (promise resolve null). Τέλος αν συνέβη κάποιο σφάλμα κατά την επικοινωνία με τη βάση δεδομένων δεν επιστρέφουμε κάποια τιμή (promise reject).

Με τη λέξη κλειδί “then”, μπορούμε να ορίσουμε τι θα συμβεί αμέσως μετά την έξοδο από τη συνάρτηση dbpromise, όπου και εκτυπώνουμε οποιοδήποτε τυχόν σφάλμα συνέβη κατά την εκτέλεση. Στο τέλος επιστρέφουμε τις τιμές που ανατέθηκαν στη δομή dbpromise, πίσω στο ενδιάμεσο λογισμικό.

Κώδικας: Βάση δεδομένων και τερματικό διακομιστή



Εικόνα 3.9 Συλλογή team στη βάση δεδομένων

Στη βάση δεδομένων που έχουμε ονομάσει verna, συμπεριλαμβάνονται οι συλλογές menu, tables και team, που φιλοξενούνται τα έγγραφα του καταλόγου του καταστήματος, των τραπεζίων που βρίσκονται στο κατάστημα και του προσωπικού του καταστήματος αντίστοιχα.

Συγκεκριμένα στη συλλογή team, βλέπουμε ότι κάθε έγγραφο έχει τα πεδία, name, pass (σε κατακερματισμένη μορφή SHA256) και role. Αυτά αντιπροσωπεύουν αντίστοιχα το, όνομα χρήστη, κωδικό και ρόλο του στο κατάστημα.

Αφού εκτελέσουμε αίτημα εύρεσης εγγράφου, έχουμε έξοδο στο τερματικό του διακομιστή με την εξής μορφή.

```
{
  username: 'George',
  password: '03ac674216f3e15c761ee1a5e255f067953623c8b388b4459e13f978d7c846f4',
  role: 'waiter'
}
POST /public/requestlogin/ 200 64.782 ms - 31
GET /waiter.html 304 1.554 ms - -
GET /stylesheets/waiter.css 304 0.640 ms - -
POST /public/askauth/ 200 1.741 ms - 55
```

Εικόνα 3.10

3.2 Κεντρική οθόνη

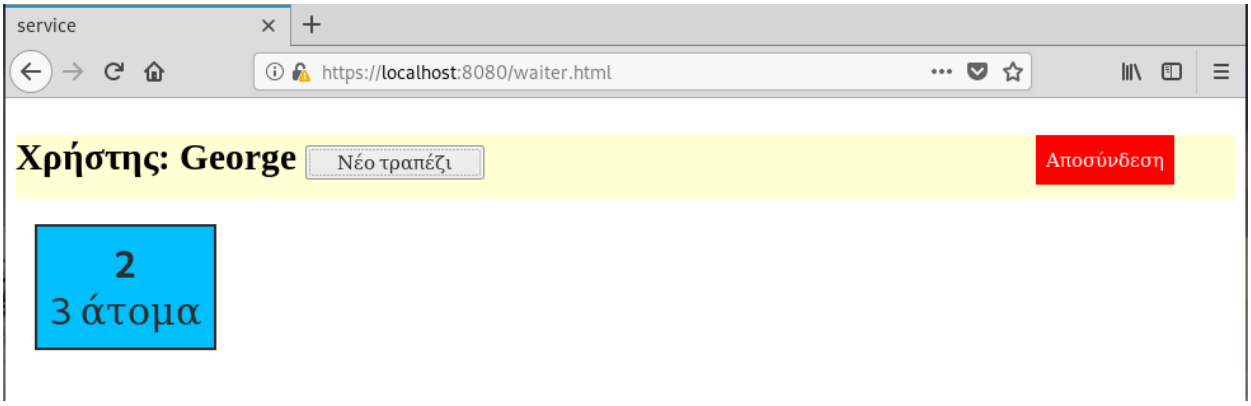
3.2.1 - Σύνοψη λειτουργίας από την οπτική του χρήστη

Η κεντρική οθόνη απαρτίζεται από τρία διακριτά τμήματα Το επάνω μέρος, το μέρος επιλογών και το τμήμα που εμφανίζονται τα τραπέζια. Στο επάνω μέρος φαίνεται το όνομα του συνδεδεμένου χρήστη ενώ στα δεξιά υπάρχει το κουμπί αποσύνδεσης από την εφαρμογή. Στην περίπτωση που ο ρόλος του χρήστη είναι “waiter”, δηλαδή σερβιτόρος, τότε δίνεται και η επιλογή για δημιουργία νέου τραπεζιού.

Στο κάτω μέρος της οθόνης εμφανίζονται τα τραπέζια που φιλοξενούν κόσμο κάθε στιγμή στο κατάστημα. Τα τραπέζια καταλαμβάνουν μικρό χώρο στην οθόνη και αναπαριστώνται ως μικρά ορθογώνια με τις βασικές πληροφορίες όπως το νούμερο τραπεζιού και τα άτομα που φιλοξενεί να αναγράφονται επάνω. Το χρώμα του τραπεζιου δηλώνει και την κατάσταση στην οποία βρίσκεται. Μόλις ο κωδικός χρώματος αλλάξει παραμένει ως έχειν ώσπου να ειδοποιηθεί για την εκάστοται αλλαγή ο κάθε ρόλος, όταν σύμβει αυτό το χρώμα ξαναγίνεται αυτό που ήταν πριν την αλλαγή. Να σημειωθεί εδώ ότι η ειδοποίηση περι της αλλαγής από συσκευή ενός ρόλου δεν θα αλλάξει το χρώμα των συσκευών και των υπόλοιπων ρόλων. Στην παρούσα έκδοση της εφαρμογής οι διαθέσιμοι κωδικοί χρώματος είναι:

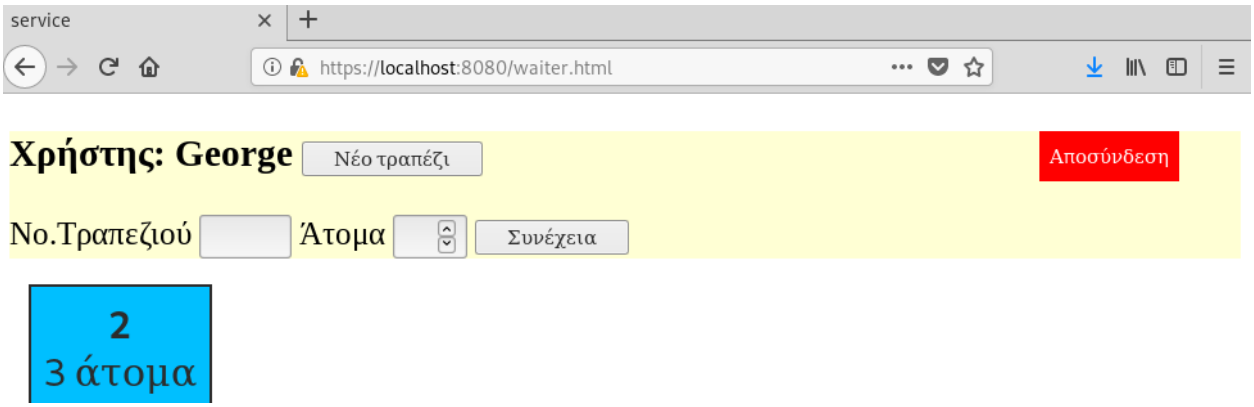
Εικόνα 3.11 Color codes

ΜΠΛΕ: ΚΑΝΟΝΙΚΗ ΚΑΤΑΣΤΑΣΗ **| ΚΟΚΚΙΝΟ**: ΝΕΟ ΤΡΑΠΕΖΙ (ΟΡΑΤΟ ΜΟΝΟ ΣΤΟΥΣ ΡΟΛΟΥΣ «ΚΟΥΖΙΝΑ Ή ΜΠΟΥΦΕ») **| ΚΙΤΡΙΝΟ**: ΤΡΟΠΟΠΟΙΗΜΕΝΗ ΠΑΡΑΓΓΕΛΙΑ **| ΠΡΑΣΙΝΟ**: ΖΗΤΗΘΗΚΕ ΛΟΓΑΡΙΑΣΜΟΣ



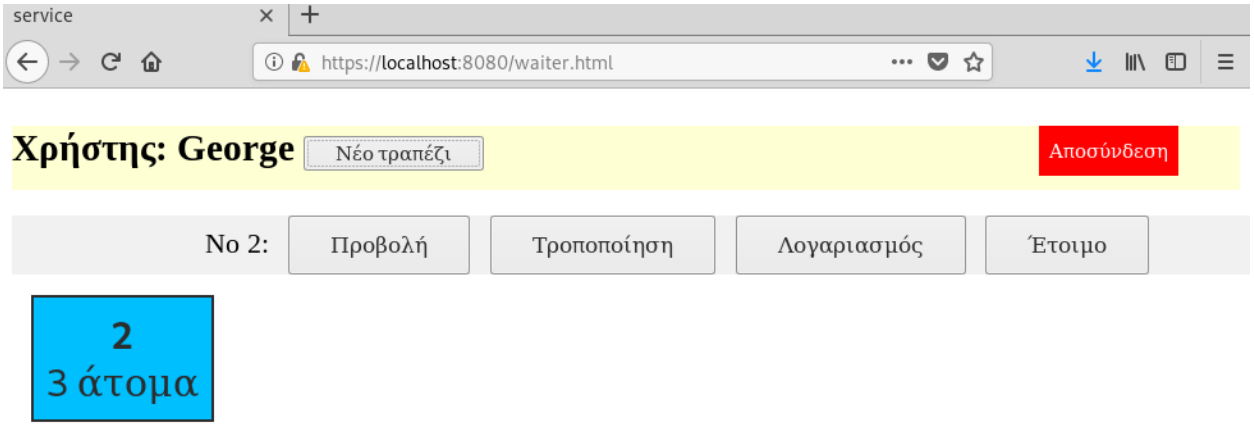
Εικόνα 3.12

Πατώντας στην ένδειξη νέο τραπέζι εμφανίζονται οι εξής επιλογές.



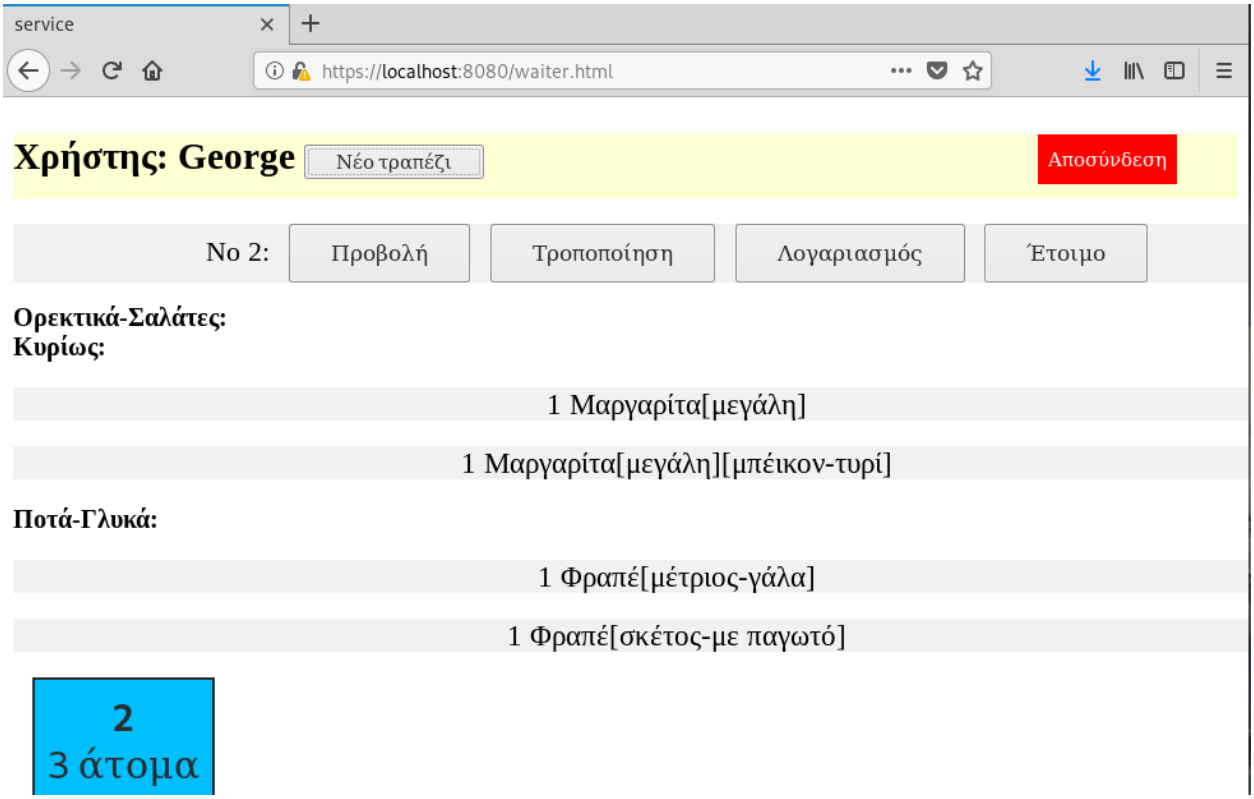
Εικόνα 3.13

Τέλος στο κέντρο της οθόνης παρουσιάζονται συνήθως μετά από μια ενέργεια του χρήστη, δίαφορες επιλογές και πληροφορίες πολλές φορές διαφορετικές ανάλογα με το ρόλο του συνδεδεμένου χρήστη. Για παράδειγμα πατώντας επάνω στο τραπέζι με αριθμό δύο, εμφανίζονται και οι επιλογές που του ρόλου “waiter”.



Εικόνα 3.14

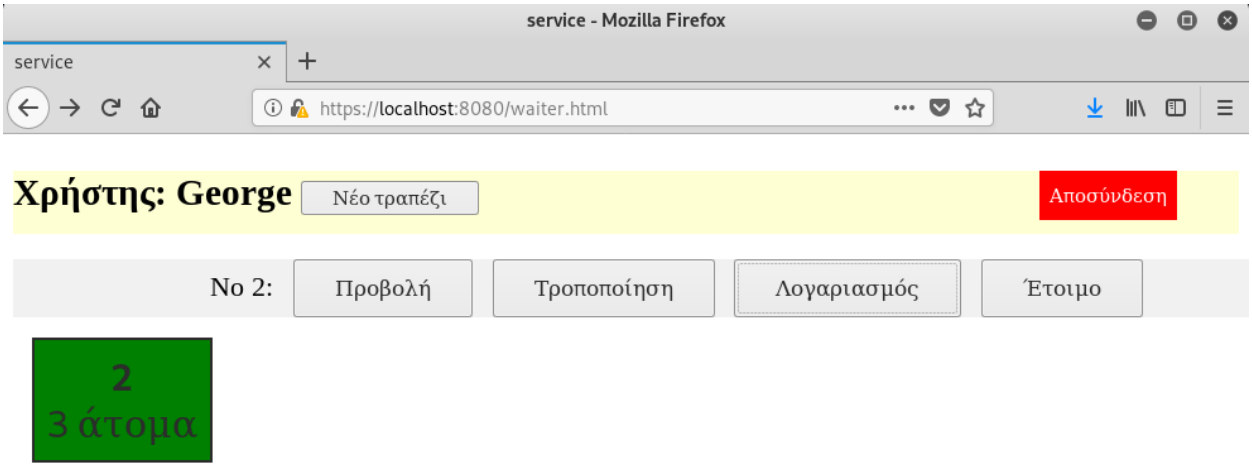
Πατώντας στην επιλογή «**Προβολή**», μπρούμε να δούμε την παραγελία ή οποία μάλιστα είναι ταξινομημένη σε, «ορεκτικά και σαλάτες», «κυρίως» και «ποτά - γλυκά».



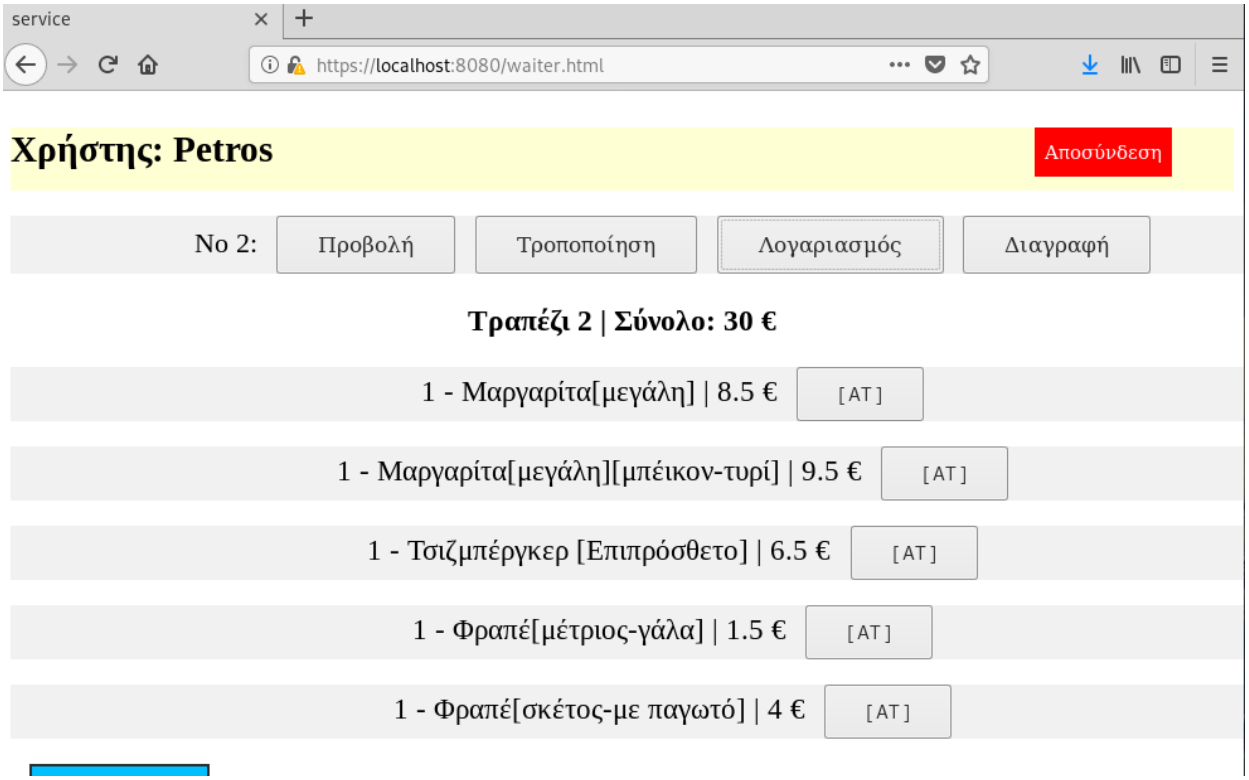
Εικόνα 3.15

Πατώντας το κουμπί «**Τροποποίηση**», μεταβαίνουμε στην οθόνη παραγγελίας (αναλύεται αργότερα), όπως θα συνέβαινε και αν είχαμε πατήσει «Νέο Τραπέζι», με τη διαφορά ότι η εφαρμογή λαμβάνει από τη βάση δεδομένων την υπάρχουσα παραγγελία επιτρέποντας μας να προσθέτουμε και να αφαιρούμε προϊόντα. Στη λειτουργία αυτή τα επιπλέον προϊόντα που προστίθονται συνοδεύονται από την επισήμανση «ΕΠΙΠΡΟΣΘΕΤΟ» και τα προϊοντα που αφαιρούνται δημιουργούν μια γενική παρατήρηση παραγγελίας της μορφής «ΑΦΑΙΡΕΘΗΚΕ <ΟΝΟΜΑ ΕΙΔΟΥΣ>». Ακόμη μετά την ολοκλήρωση της τροποποίησης, το χρώμα του τραπεζιού αλλάζει σε κίτρινο για όλες τις συνδεδεμένες συσκευές ανεξαρτήτως ρόλου, υποδυκνίωντας έτσι ότι η παραγγελίας μόλις τροποποιήθηκε.

Επιλέγοντας την επιλογή «**Λογαριασμός**», με ρόλο “waiter” ειδοποιούμε όλες τις συσκευές του δικτύου ότι το τραπέζι, ζήτησε λογαριασμό αλλάζοντας τον κωδικό χρώματος του τραπεζιού σε πράσινο, στη συνέχεια οι συσκευή με ρόλο “owner” μπορεί με την ίδια ένδειξη να υπολογήσει το ποσό του λογαριασμού.

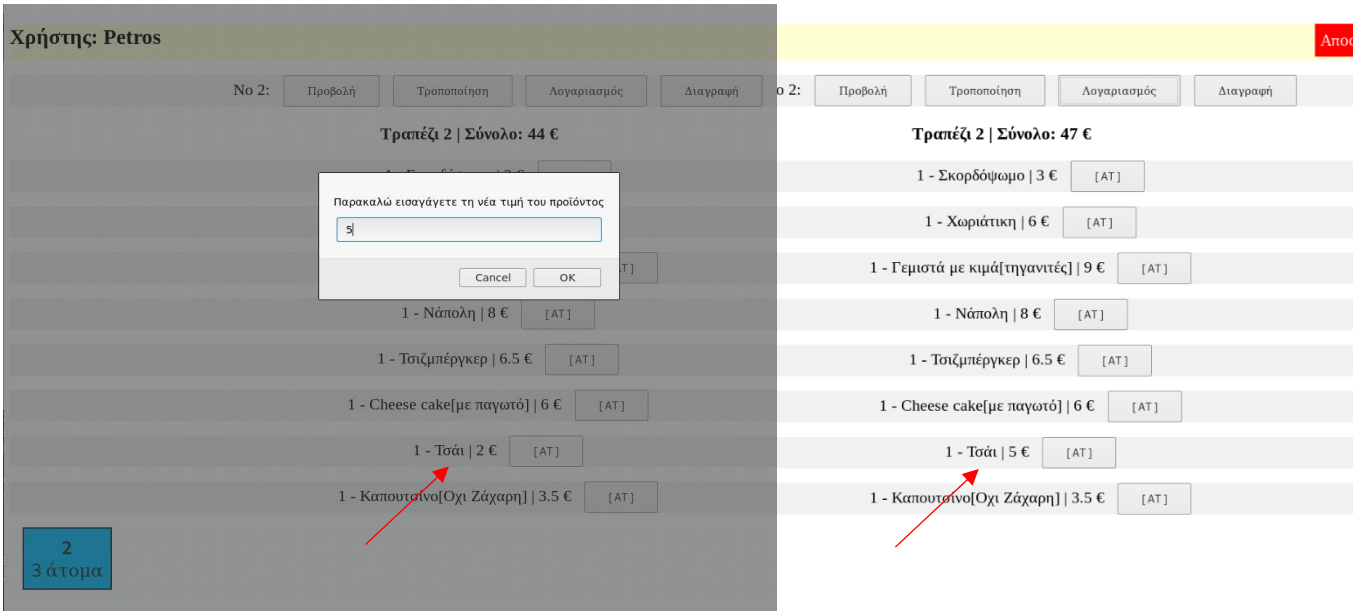


Εικόνα 3.16 Αίτηση λογαριασμού, Ρόλος ‘waiter’



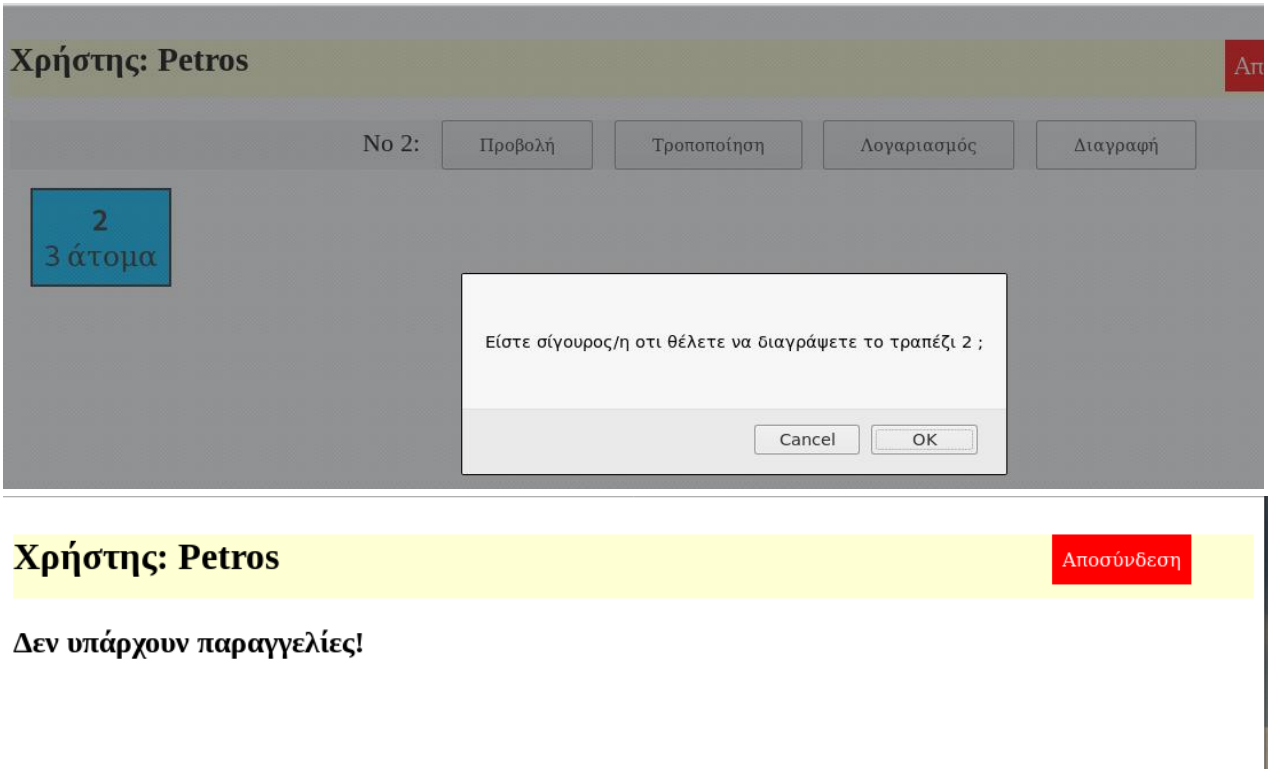
Εικόνα 3.17 Υπολογισμός λογαριασμού, Ρόλος ‘owner’

Ο ιδιοκτήτης (ρόλος “owner”), έχει επίσης την επιλογή της θέσπισης ειδικής τιμής σε προϊοντα. Το παραπάνω είναι χρήσιμο σε περίπτωση καποιας ειδικής παραγγελίας.



Εικόνα 3.18 Αλλαγή τιμής και αποτέλεσμα

Συνεχίζοντας με τις λειτουργίες του ρόλου “owner”, παρατηρούμε επιπλέον την επιλογή «Διαγραφή», η οποία διαγράφει ένα τραπέζι



Εικόνα 3.19 Διαγραφή τραπεζιού και αποτέλεσμα

3.2.2 - Κώδικας Κεντρικής οθόνης

Κώδικας: Αρχικός κώδικας

```
425         created : function(){
426             this.reqlogin();
427             this.gettables();
428         },
429         mounted : function () {
430             this.$nextTick(function () {
431                 window.setInterval(() => {
432                     this.gettables();
433                 },7000);
434             })
435         }
```

Εικόνα 3.20 Ο κώδικας που εκτελείται πρώτα με την είσοδο στην κεντρική οθόνη

Μεταβλητές data component

```
    data: {
        username : '',
        role : '',
        props : '',
        mode : '',
        customprice : 0,
        index : 0,
        createtable : false,
        tablenumber : '',
        peoplenumber : null,
        tables : [],
        currenttable : [],
        total : 0
    },
```

Εικόνα 3.21 Μεταβλητές data component (Global variables)

Μέθοδος reqlogin: Vue

Πρίν ολοκληρωθεί η διαμόρφωση της σελίδας από τον περιηγητή, εκτελείται ο κώδικας που περιλαμβάνεται στο component “created” της Vue. Εκεί καλείται η μέθοδος “reqlogin”, η οποία υπάρχει για λόγους ασφαλείας στον κώδικα.

Για να αναλύσουμε τη λειτουργία της, θα μεταβούμε στο “methods” vue component.

```
206         methods: {
207             reqlogin : function(){
208                 let that = this;
209                 axios.post('/public/askauth/')
210                 .then(function (response) {
211                 // handle success
212                 if(response.data.answer != "ok"){
213                     window.location.href = response.data.page + ".html";
214                 }
215                 that.username = response.data.username;
216                 that.role = response.data.userrole;
217                 let role = that.role;
218                 if(role == 'boufe')
219                     that.index = 0;
220                 else if(role == 'kitchen')
221                     that.index = 1;
222                 else if(role == 'waiter')
223                     that.index = 2;
224                 else if(role == 'owner')
225                     that.index = 3;
226 
227                 })
228                 .catch(function (error) {
229                 // handle error
230                 console.log(error);
231                 });
232             },
```

Εικόνα 3.22

Όπως είδαμε στο κομμάτι της εισόδου, εφόσον έχει πραγματοποιηθεί με επιτυχία η είσοδος χρήστη στο σύστημα θα πρέπει, να υπάρχουν τα στοιχεία του αποθηκευμένα στο session που ορίσαμε στο αρχείο app.js της εφαρμογής μας. Παρόλο που στον front-end κώδικα της σελίδας εισόδου, δεν φαίνεται άμεσα το όνομα του αρχείου της κεντρικής σελίδας, προς την οποία εκτελούμε ανακατεύθυνση και αρα είναι εξαιρετικά αδύνατον κάποιος να μαντέψει ή να μάθει το όνομα αυτό εμείς παρόλα αυτά είναι καλό να έχουμε θωρακίσει την εφαρμογή από χρήστες που μπορεί να πληκτρολογήσουν στην μπάρα διευθυνσης την τοποθεσία του αρχείου της κεντρικής σελίδας, δίχως να έχουν προηγουμένως συνδεθεί. Η συνάρτηση reqlogin, ελέγχει αν υπάρχουν στην παρούσα σύνοδο αποθηκευμένα τα στοιχεία username και role, εκτελώντας αίτημα ajax στον διακομιστή στη διεθύθυνση ‘/public/askauth’. Στη περίπτωση που τα στοιχεία δεν υπάρχουν, σημαίνει ότι ο χρήστης προσπαθεί να έχει προσβάση στην κεντρική σελίδα δίχως να έχει εκτελέσει σύνδεση και άρα ανακατευθύνεται

στην σελίδα εισόδου. Στην περίπτωση που τα στοιχεία βρεθούν, τότε αποθηκεύονται σε αντίστοιχες μεταβλητές στο data component του αντικειμένου Vue.

Η συνάρτηση αυτή είναι το πρώτο πράγμα που εκτελείται κάθε φορά που ανοίγουμε την κεντρική σελίδα.

Μέθοδος reqlogin: Express Js

```
102 router.post('/public/askauth/', function(req, res){
103     let user = req.session.user;
104     let role = req.session.role;
105     if(user){
106         res.send({answer : "ok" , username : user, userrole : role});
107     }
108     else{
109         res.send({answer : "kick", page : "login"});
110     }
111 });
```

Εικόνα 3.23

Μέθοδος getTables: Vue

Η μέθοδος “getTables” καλείται επαναληπτικά ανα επτά δεφτερόλεπτα καθόλη τη διάρκεια της παραμονής του χρήστη στην κεντρική οθόνη. Η μέθοδος είναι υπεύθυνη για την ανάκτηση όλων των τραπεζιών που είναι καταχωρημένα στη βάση δεδομένων. Ο λόγος που εκτελείται επαναληπτικά είναι γιατί η συλλογή τραπεζιών στη βάση δεδομένων δύναται να αλλάξει ανα πάσα στιγμή, από τις συνδεδεμένες συσκευές με ρόλο σερβιτόρου ή ιδιοκτήτη, συνεπώς χρειάζεται τακτικός έλεγχος ώστε τα δεδομένα να είναι πάντα ενημερωμένα. Τα επτά δεπτερόλεπτα δεν επιλέχθηκαν για κάποιο πολύ συγκεκριμένο λόγο. Γενικότερα θα θέλαμε το χρόνικο διάστημα που παεμβάλεται ανάμεσα σε κάθε επανάληψη να μην είναι πολύ μικρό για λόγους αποφυγής υπερφότρωσης του δικτύου, αλλα ούτε και πολύ μεγάλο, καθώς επιθυμούμε τακτική ανανέωση και ενημέρωση των δεδομένων.

```
246             gettables : function(){
247                 let that = this;
248                 axios.post('/public/gettables/')
249                 .then(function (response) {
250                 // handle success
251                 if(response.data != "table not found"){
252                     that.tables = response.data;
253                 }
254                 else
255                     console.log("something went wrong at the server");
256                 })
257                 .catch(function (error) {
258                 // handle error
259                 console.log(error);
260                 })
261             },
```

Εικόνα 3.24

Η μέθοδος εκτελέι ένα ajax αίτημα στο διακομιστή (διεύθυνση '/public/getTables/') και αποθηκεύει τα δεδομένα που της επιστρέφονται στη μεταβλητή tables που έχουμε ορίσει στο data component. Η μεταβλητή tables είναι πίνακας που αποθηκεύει πολλαπλά αντικείμενα json τύπου table (αναλύεται στο 3.3.2)

```
router.post('/public/gettables/', function(req, res){
    bend.getTables().then((value) => {
        if(value){
            res.send(value);
        }
        else{
            res.send("table not found");
        }
    });
```

Εικόνα 3.25 Express Router για τη μέθοδο gettables

```
    const dbpromise = new Promise((resolve, reject) => {
        var url = "mongodb://localhost:27017/";
        const client = new MongoClient(url, { useNewUrlParser: true });

        client.connect((err => {
            const collection = client.db("verna").collection("tables");
            collection.find((err, result) => {
                if(err) throw err;
                var tables = {};
                if(result != null){
                    tables = result.toArray();
                }
                else{
                    resolve(null);
                }
                if (tables) resolve(tables);
                else reject({Error: 'something went wrong'});

            });
            client.close();
        }));
    });
    dbpromise.then((err, result) => {
        if (err) console.log(err);
            return result;
    });
    return dbpromise;
};
```

Εικόνα 3.26 Node js κώδικας για τη μέθοδο gettables

Μέθοδος getTables: Express js και Node js

Για τον παραπάνω κώδικα δεν απαιτείται κάποια ειδική αναφορά, καθώς η διαδικασίες έχουν ήδη περιγραφεί. Η μοναδική ουσιαστική διαφορά βρίσκεται στο αίτημα προς τη βάση το οποίο είναι πιο απλό καθώς λαμβάνει όλα τα έγγραφα της συλλογής "tables".

Κώδικας: Επάνω κομμάτι σελίδας (TopPage)

TopPage: Κώδικας HTML

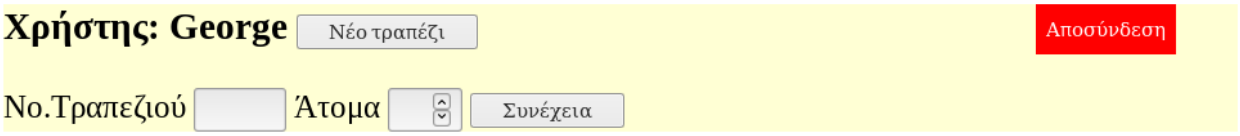
```
<div id='toppage'>
    <h3 style="min-height:5px;">Χρήστης: {{username}}
        <button v-if="role=='waiter'"v-on:click='togglecreate'>Νέο τραπέζι</button>
        <button id="logb" v-on:click='logout'>Αποσύνδεση</button>
    </h3>
    <div id="tableoptions" v-if='createtable'>
        No.Τραπεζιού <input type="text" v-model='tablenumber' size="4">
        Άτομα <input type="number" v-model='peoplenumber' min="1" max="30" size="4">
    <button v-on:click='newtable'>Συνέχεια</button>
    </div>
</div>
```

Εικόνα 3.27

Ξεκινάμε από το επάνω μέρος της σελίδας με αναγνωριστικό id “toppage”, οπού αναγράφεται το όνομα του συνδεδεμένου χρήστη κάνωντας χρήση της μεταβλητής {{username}} (Εικόνα 3.27 γραμμή 14).

Το στοιχείο h3 είναι parent component δύο επιμέρους στοιχείων τύπου “button”, υπεύθυνα για την δημιουργία νέου τραπεζιού και για την αποσύνδεση χρήστη αντίστοιχα, τα κουμπιά αυτά έιναι συδεδεμένα με συμβάντα και καλούν μεθόδους του αντικειμένου Vue εφόσον πατηθούν.

Παρακάτω φαίνεται άλλο ένα στοιχείο div με αναγνωριστικό “tableoptions”, το οποίο εμφανίζεται μόνο εφόσον η μεταβλητή “createtable” έχει αληθής τιμή. Εντός αυτού εντοπίζουμε δύο στοιχεία τύπου “input”, στα οποία μπορεί ο χρήστης να εισάγει το νούμερο του νέου τραπεζιού και τον αριθμό των ατόμων που κάθονται σε αυτό, ενώ βλέπουμε επίσης ακόμη ένα κουμπί «Συνέχεια» που μας οδηγεί στην επόμενη οθόνη.



Εικόνα 3.28

Ας αναλύσουμε τη λειτουργία των μεθόδων που καλούνται σε αυτό το τμήμα html

TopPage: Κώδικας Vue, Express js και node js

Αρχικοποιήσεις μεταβλητών στο data component

Οι μεταβλητές tablenumber και peoplenumber είναι συνδεδεμένες με τις αντίστοιχες εισόδους (input) στον κώδικα html, χρησιμοποιώντας τη δομή v-bind

```
199             createtable : false,
200             tablenumber : '',
201             peoplenumber : null,
202             tables : [],
```

Εικόνα 3.29

Η μέθοδος togglecreate, εναλλάσει την τιμή της μεταβλητής createtable, σε true ή false, καθιστώντας έτσι το τμήμα html με αναγνωριστικό “tableoptions” ορατό ή όχι αντίστοιχα.

```
416         togglecreate : function(){
417             if(this.createtable){
418                 this.createtable = false;
419             }
420             else
421                 this.createtable = true;
422         }
```

Εικόνα 3.30

Η μέθοδος logout, αποσυνδέει το χρήστη από το σύστημα και αρχικοποιεί τη σύνοδο (session), στις αρχικές τιμές της.

```
233         logout : function(event){
234             axios.post('/public/logout/')
235             .then(function (response) {
236             // handle success
237             if(response.data.answer == "ok"){
238                 window.location.href = response.data.page + ".html";
239             }
240             })
241             .catch(function (error) {
242             // handle error
243             console.log(error);
244             })
245         },
```

```
97  router.post('/public/logout/', function(req, res){
98      req.session.reset();
99      res.send({answer : "ok", page : "login"});
100 });
```

Εικόνα 3.31

Τέλος η μέθοδος newtable που συμβάλει στην δημιουργία νέου τραπεζιού, αρχικά εκτελεί έλεγχο για να αποφευχθεί η περίπτωση να μην έχει οριστεί τιμή στη μεταβλητή “tablenumber”, καθώς το πεδίο αριθμός τραπεζιού είναι απαραίτητο. Στη συνέχεια εκτελείται αναζήτηση στον πίνακα με τα υπάρχοντα τραπέζια “tables”, ώστε να διαπιστωθεί αν αριθμός τραπεζιού που ο χρήστης προσπαθεί να δημιουργήσει υπάρχει ήδη. Σε περίπτωση διπλότυπου νούμερου, εμφανίζεται κατάλληλο μήνυμα λάθους και η διαδικασία σταματάει. Σε διαφορετική περίπτωση, αν δηλαδή ο αριθμός τραπεζιού εμφανίζεται πρώτη φορά, η συνάρτηση εκτελεί ανακατεύθυνση του περιηγητή στην οθόνη παραγγελίας προσθέτοντας παράλληλα τις τιμές

του, αριθμού τραπεζιού και του αριθμού ατόμων στην διεύθυνση url της οθόνης παραγγελίας για μελλοντική χρήση τους.

```
289             newtable : function(){
290                 let i;
291                 let length = this.tables.length;
292                 let exists = false;
293                 if(this.tablenumber==''){
294                     alert("Παρακαλώ συμπληρώστε αριθμό τραπεζιού");
295                     return;
296                 }
297                 for(i=0; i<length; i++){
298                     if(this.tables[i].tnumber == this.tablenumber){
299                         exists = true;
300                         break;
301                     }
302                 }
303                 if(exists){
304                     alert("Το τραπέζι " + this.tablenumber + " υπάρχει ήδη!");
305                     this.tablenumber = null;
306                 }
307                 else{
308                     let url = '/order.html?tn=' + this.tablenumber + '?tp=' + this.peoplenumber;
309                     window.location = url;
310                 }
311             },
```

Εικόνα 3.32

Κώδικας: Μεσαίο τμήμα κεντρικής οθόνης (επιλογές ανά ρόλο)

Σε αυτό το σημείο θα αναλυθούν οι μέθοδοι που αντιστοιχούν στις επιλογές προβολής και επεξεργασίας των υπάρχοντων τραπεζιών ανάλογα με το ρόλο του συνδεδεμένου χρήστη.

Role propetries: HTML

```
<div class="props" v-if="props == 'owner'">
    <ul>
        <li>
            No {{currenttable.tnumber}}:
            <button class='propb' v-on:click='vieworder'>Προβολή</button>
            <button class='propb' v-on:click='changeorder'>Τροποποίηση</button>
            <button class='propb' v-on:click='calccheck'>Λογαριασμός</button>
            <button class='propb' v-on:click='removetable'>Διαγραφή</button>
        </li>
    </ul>
</div>
<div class="props" v-if="props == 'waiter'">
    <ul>
        <li>
            No {{currenttable.tnumber}}:
            <button class='propb' v-on:click='vieworder'>Προβολή</button>
            <button class='propb' v-on:click='changeorder'>Τροποποίηση</button>
            <button class='propb' v-on:click='requestcheck'>Λογαριασμός</button>
            <button class='propb' v-on:click=''>Έτοιμο</button>
        </li>
    </ul>
</div>
<div class="props" v-else-if="(props == 'kitchen || props == 'boufe')">
    <ul>
        <li>
            No {{currenttable.tnumber}}:
            <button class='propb' v-on:click='vieworder'>Προβολή</button>
            <button class='propb' v-on:click=''>Σερβίρω</button>
        </li>
    </ul>
</div>
```

Εικόνα 3.34

Παρακάτω θα αναλυθούν οι μέθοδοι: “vieworder”, “changeorder”, “calccheck”, “changeprice”, “removetable” και “toggleorset” που εκτελούν, «προβολή παραγγελίας», «τροποποίηση

Εικόνα 3.34

παραγγελίας», «υπολογισμό λογαριασμού», «αίτημα λογαριασμού» κ’ «διαγραφή τραπεζιού».

ViewOrder: Κώδικας HTML

```
<!-- KITCHEN -->
<div class="vieworder" v-else-if="mode=='view' && role=='kitchen'">
    <h3 v-if="currenttable.tgnotes">! {{currenttable.tgnotes}} !</h3>
    <strong>Ορεκτικά-Σαλάτες:</strong><br>
    <ul v-for="item in currenttable.torder[0]">
        <li>
            <span v-if="item.ftype[1] == 'hot' || item.fname.includes('πρώτο')">*</span> {{item.quant}} {{item.fname}}
        </li>
    </ul>
    <strong>Κυρίως:</strong>
    <ul v-for="item in currenttable.torder[1]">
        <li>
            <template v-if="item.ftype[1] == 'cold' || item.ftype[0] == 'salads'">{{item.quant}} {{item.fname}}</template>
            <template v-else>* {{item.quant}} {{item.fname}}</template>
        </li>
    </ul>
</div>

<!-- WAITER -->
<div class="vieworder" v-else-if="mode=='view' && role=='waiter'">
    <h3 v-if="currenttable.tgnotes">! {{currenttable.tgnotes}} !</h3>
    <strong>Ορεκτικά-Σαλάτες:</strong><br>
    <ul v-for="item in currenttable.torder[0]">
        <li>
            <span v-if="item.ftype[1] == 'bread'">*</span> {{item.quant}} {{item.fname}}
        </li>
    </ul>
    <strong>Κυρίως:</strong><br>
    <ul v-for="item in currenttable.torder[1]">
        <li>
            {{item.quant}} {{item.fname}}
        </li>
    </ul>
    <strong>Ποτά-Γλυκά:</strong>
    <ul v-for="item in currenttable.torder[2]">
        <li>
            <span v-if="item.ftype[0] == 'drinks'">*</span> {{item.quant}} {{item.fname}}
        </li>
    </ul>
</div>
```

Εικόνα 3.36

Η μέθοδος «προβολή» παρουσιάζει την παραγγελία του επιλεγμένου τραπεζιού κάθε φορά, σε μορφή κατακόρυφης λίστας, ταξινομημένη σε τρείς πίνακες που αντιπροσωπεύουν τα «Ορεκτικά-Σαλάτες» - «Κυρίως» - «Ποτά-Γλυκά». Αυτό πραγματοποιείται κάνοντας χρήση της δομής v-for, η οποία μας επιτρέπει να παρουσιάζουμε ατομικά κάθε στοιχείου που

συμπεριλαμβάνεται στους πίνακες παραγγελίας του εκάστοτε τραπεζιού. Παράλληλα υλοποιούνται ανάλογα με το ρόλο του χρήστη και διάφοροι έλεγχοι ώστε τα αντικείμενα ενδιαφέροντος εντός της παραγγελίας, να συνοδεύονται από τον χαρακτήρα '*', ενώ ταυτόχρονα εξαλείφονται από την προβολή αντικείμενα που μπορεί να είναι αδιάφορα για ένα ρόλο. Για παράδειγμα, κατά την προβολή μιας παραγγελίας με τη συσκευή σερβιτόρου, θα παρατηρήσουμε στο αριστερό μέρος κάθε αντικειμένου είδους «ποτού» τον χαρακτήρα '*'. Ανοίγοντας όμως την ίδια παραγγελία από τη συσκευή κουζίνας, θα παρατηρήσει κανείς ότι τα ποτά αυτά δεν εμφανίζονται, ενώ οι χαρακτήρες '*' έχουν μετακινηθεί σε διαφορετικού είδους αντικείμενα.

vieworder - toggleorset: Κώδικας Vue

```
393             vieworder : function(){
394                 let mode = this.mode;
395                 this.mode = this.toggleorset(mode,"view");
396             },
397             toggleorset : function(value,set){
398             if(value){
399                 if(value == true)
400                     return false;
401                 else
402                     return "";
403             }
404             else
405                 return set;
406             },
```

Εικόνα 3.37

Ο κώδικας Vue για την μέθοδο «προβολή» είναι αρκετά σύντομος και πρακτικά είναι υπεύθυνος μόνο για την αλλαγή τιμής της μεταβλητής "mode", η οποία έχει αρχικοποιηθεί στο data component. Η αλλαγή πραγματοποιείται κάνωντας χρήση της βοηθητικής συνάρτησης "toggleorset", που κατασκευάστηκε για να έτσι ώστε να αλλάζει την τιμή μιας μεταβλητής.

Η συνάρτηση toggleorset λαμβάνει ώς πρώτο όρισμα τη μεταβλητή προς επεξεργασία και επιστρέφει την αλλαγμένη τιμή της.

- Από true σε false και αντίστροφα, εάν πρόκειται για boolean μεταβλητή
- Από οτιδίποτε άλλο σε κενό αλφαριθμιτικό (τύπου reset)
- Σε περίπτωση που η μεταβλητή δεν έχει τιμή, τότε καταχώρείται ώς τιμή το δεύτερο όρισμα της συνάρτησης.

Ο κώδικας "this.mode = this.toggleorset(mode,'view');" θέτει την τιμή της μεταβλητής mode σε 'view' κάνοντας έτσι ορατά τα τμήματα κώδικα html που εξαρτώνται από την μεταβλητή αυτή.

Changeorder: Κώδικας Vue

```
311             changeorder : function(){
312                 let number = this.currenttable.tnumber;
313                 let url = '/order.html?ch=' + number;
314                 window.location = url;
315             },
```

Εικόνα 3.38

Προκειμένου να αναλυθεί ο παραπάνω κώδικας θα πρέπει προηγουμένως να αναφέρουμε τη μεταβλητή “currenttable”, η οποία λαμβάνει τιμή όποτε ο χρήστης επιλέγει ένα τραπέζι και είναι ίση με το αντικείμενο json που αντιπροσωπεύει το τραπέζι αυτό.

Λαμβάνουμε την τιμή currenttable.tnumber που ισούται με τον αριθμό τραπεζιού και τη χρησιμοποιούμε για να εκτελέσουμε ανακατεύθυνση στην οθόνη παραγγελίας εισχωρόντας την στην διεύθυνση του περιηγητή προς μελλοντική χρήση.

Calccheck: Κώδικας HTML

```
<div class="vieworder" v-else-if="mode=='check' && role=='owner'">
    <h3 style="text-align: center">Τραπέζι {{currenttable.tnumber}} | Σύνολο: {{total}} €</h3>
    <ul v-for="item in currenttable.torder[0]">
        <li>
            {{item.quant}} - {{item.fname}} | {{item.fprice * item.quant}} €
            <button :name= "item.fname" category="starters" v-on:click="changeprice($event)">[ AT ]</button>
        </li>
    </ul>
    <ul v-for="item in currenttable.torder[1]">
        <li>
            {{item.quant}} - {{item.fname}} | {{item.fprice * item.quant}} €
            <button :name= "item.fname" category="maincourses" v-on:click="changeprice($event)">[ AT ]</button>
        </li>
    </ul>
    <ul v-for="item in currenttable.torder[2]">
        <li>
            {{item.quant}} - {{item.fname}} | {{item.fprice * item.quant}} €
            <button :name= "item.fname" category="rest"v-on:click="changeprice($event)">[ AT ]</button>
        </li>
    </ul>
</div>
```

Εικόνα 3.39

Για τον υπολογισμό του λογαριασμού εκτελούμε σχεδόν την ίδια διαδικασία με αυτή της προβολής παραγγελίας, με τη διαφορά πως ελέγχουμε με τη δομή v-if αν οι τιμές των μεταβλητών mode και role είναι ίσες με “check” και “owner” (θυμηθείτε ότι κατά την προβολή η μεταβλητή mode είχε τιμή ίση με “view”). Στη συνέχεια εμφανίζουμε τα στοιχεία της παραγγελίας ώς εξής: <Ποσότητα> - <Προϊον> | <Τιμή*Ποσότητα> € , σε μοφή κατακόρυφης λίστας. (Εικόνα 3.17)

Calccheck: Κώδικας Vue

```
346             calccheck : function(){
347                 let i;
348                 let total=0;
349                 let order = this.currenttable.torder[0].concat(this.currenttable.torder[1], this.currenttable.torder[2]);
350                 for(i=0; i<order.length;i++){
351                     total += order[i].fprice * order[i].quant;
352                 }
353                 let mode = this.mode;
354                 this.mode = this.toggleorset(mode,"check");
355                 this.total = total;
356             },
```

Εικόνα 3.40

Η συνάρτηση calccheck, αρχικά συνενώνει τους πίνακες παραγγελίας torder[0], torder[1] και torder[2] (Ορεκτικά-Κυρίως Γλυκά), φτιάχνοντας ένα πίνακα που περιέχει τη συνολική παραγγελία με όνομα order (τοπική μεταβλητή). Στη συνέχεια χρησιμοποιούμε μια επαναληπτική δομή for, ώστε να προσπελάσουμε όλα τα στοιχεία της παραγγελίας και προσθέτουμε κάθε φορά το γινόμενο της τιμής του κάθε στοιχείου επί την ποσότητα αυτού στην τοπική μεταβλητή total που κρατάει το συνολικό χρηματικό αντίτιμο (Εικόνα 3.4, γραμμή 351). Τέλος κάνουμε χρήση της βοηθητικής συνάρτησης toggleorset, ώστε να θέσουμε την τιμή της μεταβλητής mode σε “check”, κάνοντας έτσι ορατά τα αντίστοιχα στοιχεία html που εξαρτώνται από την τιμή της. (Εικόνα 3.39)

ChangePrice: Κώδικας Vue

Εντός της λειτουργίας υπολογισμού λογαριασμού, βρίσκεται άλλη μια επιπλέον λειτουργία, αυτή της αλλαγής τιμής, που ενεργοποιείται πατώντας το κουμπί Α.Τ (Εικόνα 3.18).

```
changeprice : function(e){
    let newprice = window.prompt("Παρακαλώ εισαγάγετε τη νέα τιμή του προϊόντος","");
    let foodname = e.target.name;
    let category = e.target.getAttribute('category');
    let tablenumber = this.currenttable.tnumber;

    if(isNaN(newprice)){
       alert("Επιτρέπονται μονο αριθμιτικές τιμές ");
       return;
    }
    else{
       axios.post('/public/storenewprice/', {tablenumber : tablenumber, foodname : foodname, category: category, newprice : newprice})
           .catch(function (error) {
           // handle error
           console.log(error);
       });
       window.location.reload();
    }
},
```

Εικόνα 3.41

Η μέθοδος changeprice, αρχικά δημιουργεί τέσσερις νέες τοπικές μεταβλητές: Η: newprice, λαμβάνει την επιθυμητή τιμή για το προϊόν από την είσοδο χρήστη σε αναδυόμενο παράθυρο, η foodname, λαμβάνει το όνομα του προϊόντος που επιλέχθηκε χρησιμοποιώντας το πεδίο html :name (Το αναγνωριστικό ‘:’ πριν από ένα πεδίο html επιτρέπει τη χρήση Vue για την εισαγωγή κάποιας μεταβλητή εντός του πεδίου αυτού. Βλέπε εικόνα 3.39), η category μια μεταβλητή που

χρησιμοποιούμε για να διαπιστώσουμε σε ποιον υπό-πίνακα της παραγγελίας ανήκει το προϊόν , και η tablenumber που είναι απλά ο αριθμός του τραπεζιού. Στη συνέχεια εκτελούμε έναν έλεγχο για το αν η τιμή της μεταβλητής newprice είναι αριθμητική και έπειτα ένα αίτημα ajax μεταφέροντας όλα τα παραπάνω στοιχεία στον διακομιστή.

ChangePrice: Κώδικας Express js και Node js

```
router.post('/public/storenewprice/', function(req, res){
    bend.storenewprice(req.body.tablenumber, req.body.foodname, req.body.category, req.body.newprice);
    res.send('ok');
});
```

Εικόνα 3.42

Παραπάνω βλέπουμε το δρομολογητή να καλεί τη συνάρτηση storenewprice, περνώντας τα κατάλληλα ορίσματα.

```
exports.storenewprice = function(tablenumber, foodname, category, newprice){
    const dbpromise = new Promise((resolve, reject) => {
        var url = "mongodb://localhost:27017/";
        const client = new MongoClient(url, { useNewUrlParser: true });


        client.connect((err => {
            const collection = client.db("varna").collection("tables");
            collection.findOne({'tnumber':tablenumber}).then(function(table){
                let i;
                if(category == "starters"){
                    for(i=0; i<table.torder[0].length; i++){
                        if(table.torder[0][i].fname == foodname){
                            table.torder[0][i].fprice = newprice;
                        }
                    }
                }
                else if(category == "maincourses"){
                    for(i=0; i<table.torder[1].length; i++){
                        if(table.torder[1][i].fname == foodname){
                            table.torder[1][i].fprice = newprice;
                        }
                    }
                }
                else if(category == "rest"){
                    for(i=0; i<table.torder[2].length; i++){
                        if(table.torder[2][i].fname == foodname){
                            table.torder[2][i].fprice = newprice;
                        }
                    }
                }
                //Update table with new order
                collection.updateOne({'tnumber' : tablenumber} , { $set: { 'torder' : table.torder } } , write_concern = {'w':1});
                client.close();
            });
        }));
    });
    dbpromise.then((err, result) => {
        if (err) console.log(err);
            return result;
    });
    return dbpromise;
};
```

Εικόνα 3.43

Η μέθοδος storenewprice αφού κάνει σύνδεση με τη βάση δεδομένων, ελέγχει την τιμή της μεταβλητής category. Αν η τιμή αυτή είναι ίση με “starters”, τότε καταλαβαίνουμε ότι το αντικείμενο του οποίου θέλουμε να αλλάξει η τιμή ανήκει στον πίνακα «ορεκτικά – σαλάτες»

και άρα κοιτάμε επαναληπτικά κάθε στοιχείου της παραγγελίας αυτού του πίνακα ώστε να δούμε αν κάποιο από τα ονόματα των προϊόντων που περιλαμβάνει,, ταιριάζει με το όνομα του αντικειμένου που επιθυμούμε να υποστεί αλλαγή. Με αντίστοιχο τρόπο δρούμε και αν η τιμή του category είναι “maincourses” ή “rest”. Εφόσον έχουμε εντοπίσει το αντικείμενο τότε θέτουμε την τιμή του (fprice) να είναι ίση με τη νέα τιμή που πήραμε από το χρήστη (newprice), όπως φαίνεται στην εικόνα 3.43 στις γραμμές 168,175 και 182. Τέλος εκτελούμε μια εργασία update στη βάση δεδομένων η οποία αντικαθιστά ολόκληρη την παραγγελία με τη νέα που περιέχει την ανανεωμένη τιμή του προϊόντος (γραμμή 187).

RemoveTable: Κώδικας Vue

```
removetable : function(){
    if(confirm("Είστε σίγουρος/η οτι θέλετε να διαγράψετε το τραπέζι " + this.currenttable.tnumber + " ;")){
        axios.post('/public/removetable/', {tablenumber : this.currenttable.tnumber})
        .catch(function (error) {
        // handle error
        console.log(error);
        });
    }},
```

Εικόνα 3.45

Ο κώδικας στη συσκεύη του πελάτη για τη μέθοδο removetable, αποτελείται απλά από την ερώτηση προς τον χρήστη για το αν είναι σίγουρος για την ενέργεια διαγραφής τραπεζιού και στη συνέχεια ένα αίτημα ajax τύπου post στη διεύθυνση ‘/public/removetable/’ , δίνοντας ώς όρισμα μόνο το νούμερο του παρόντος τραπεσιού με ονομασία ‘tablenumber’.

RemoveTable: Κώδικας Express js και Node js

```
router.post('/public/removetable/', function(req, res){
    bend.removetable(req.body.tablenumber);
    res.send('ok');
});
```

```
37  exports.removetable = function(tablenumber){
38      var url = "mongodb://localhost:27017/";
39      const client = new MongoClient(url, { useNewUrlParser: true });
40      let index;
41
42      client.connect((err => {
43          const collection = client.db("verna").collection("tables");
44          try{
45              collection.deleteOne({tnumber : tablenumber} , write_concern = {'w':1});
46          }catch(e){
47              console.log(e);
48          }
49          client.close();
50      }));
```

Εικόνα 3.46 Κώδικας express js και node js για τη μέθοδο removerable

Ο δρομολογητής που «ακούει» για συμβάντα στην κατάλληλη διεύθυνση , καλεί τη μέθοδο removetable που βρίσκεται στο αρχείο backend.js και περνάει ως όρισμα τον αριθμό τραπεζιού που έλαβε από τη συσκευή πελάτη. Στη συνέχεια εκτελείται κατάλληλο αίτημα αφαίρεσης καταχώρησης από τη συλλογή tables στη βάση δεδομένων με βάση τον αριθμό τραπεζιού

Κάτω μέρος κεντρικής οθόνης: HTML

```
<!--COLOR CODES: BLUE - Normal State (Seen)
                 RED - New Order (Not opened)
                 YELLOW - Order Changed(Addon/Removal)
                 GREEN- Requested bill/Ready to eat(Only for Kitchen)
                 GREY - Served
                 BLACK - Complaint

-->
<div id="tables" v-for='table in tables'>
    <button :name='table.tnumber' class='tableb' v-bind:class="{ newtableb : table.tnew[index] ,
                                                                 addonb : (table.talert[index] && table.talert[4].includes('addon')) ,
                                                                 requestcheckb : (table.talert[index] && table.talert[4].includes('requestcheck') && role != 'kitchen' && role != 'boufe') }"
                                                                 v-on:click='tableprops'><strong>{{table.tnumber}}</strong><br>
                                                                 {{table.tpeople}} άτομα
    </button>
</div>
```

Εικόνα 3.47

Ο παραπάνω κώδικας είναι υπεύθυνος για την παρουσίαση των τραπεζιών στο κάτω τμήμα της κεντρικής οθόνης. Χρησιμοποιώντας τη δομή v-for έχουμε πρόσβαση σε όλα τα αντικείμενα του πίνακα tables, ο οποίος παίρνει τιμές κάθε φορά που εκτελείται η συνάρτηση gettables. Για κάθε τραπέζι λοιπόν δημιουργούμε ένα κουμπί με όνομα τον αριθμό του τραπεζιού (‘ :name=table.tnumber ‘), ενώ η κλάση css που θα καθορίσει την εμφάνιση του, αρχικά έχει οριστεί να είναι η ‘tableb’ (μπλε χρώμα). Τέλος στο κουμπί αναγράφονται και το νούμερο τραπεζιού και ο αριθμός ατόμων, με τη χρήση των μεταβλητών του αντικειμένου table, ‘table.tnumber’ και ‘table.tpeople’.

Η κλάση css παρόλα αυτά μπορεί να αλλάξει δυναμικά αν πληρούνται οι συνθήκες που έχουν οριστεί εντός του αναγνωριστικού v-bind:class της vue, που συσχετίζει μια css κλάση με ένα στοιχείο html εφόσον ισχύει κάποια συνθήκη με τον εξής τρόπο ‘v-bind:class = {css_classname : grade=0}’

Σύστημα ειδοποιήσεων με κωδικούς χρώματος (Color Codes)

Το αντικείμενο table, περιέχει ένα πεδίο που έχουμε ονομάσει ‘talert’ και είναι πίνακας πέντε στοιχείων. Το πεδίο αυτό κωδικοποιεί διάφορες καταστάσεις στο κάθε τραπέζι. Τα πρώτα τέσσερα στοιχεία, αντιπροσωπεύουν τους ρόλους μέσα στο κατάστημα, δηλαδή μπουφέ (αν υπάρχει), κουζίνα, σερβιτόροι και ιδιοκτήτης, αυτά είναι τύπου boolean και έχουν αρχικά τιμή true. Η αληθής τιμή πρακτικά σημαίνει ότι ο ρόλος πρέπει να ειδοποιηθεί για την εκάστοτε ειδοποίηση. Η τελευταία θέση του πίνακα είναι αλφαριθμητικό και αντιπροσωπεύει την κατάσταση (alert) στην οποία βρίσκεται ένα τραπέζι.

Για παράδειγμα σε περίπτωση που ένα τραπέζι τροποποιήσει την παραγγελία του παραγγέλνοντας κάτι επιπλέον ή αφαιρώντας από την υπάρχουσα παραγγελία, τότε στο αντικείμενο table θα αποθηκευτεί ο πίνακας talert(true, true, true, true, “addon”), αυτό μας ενημερώνει ότι η ειδοποίηση-alert είναι “τροποποίηση παραγγελίας-addon” και ότι όλοι χρειάζεται να το μάθουν αυτό. Αν τώρα υποθετικά η κουζίνα πατήσει επάνω στο τροποποιημένο τραπέζι, ο πίνακας θα λάβει τη μορφή talert(true,**false**,true, true,”addon”) και συνεπώς εξαιτίας των συνθηκών ελέγχου στον κώδικα html το τραπέζι θα έχει διαφορετικό χρώμα για την κουζίνα ή οποία μόλις ειδοποιήθηκε και διαφορετικό για όλους τους άλλους.

Η επαναφορά του προεπιλεγμένου χρώματος μετά την επιλογή ενός τραπεζιού, γίνεται με την τροποποίηση του πίνακα talert ή tnew στη βάση δεδομένων.

Ο πίνακας tnew περιλαμβάνεται επίσης στα αντικείμενα τύπου table παρόμοια δομή με τον talert, με τη διαφορά ότι περιλαμβάνει μόνο δύο θέσεις καθώς αφορά μόνο τους ρόλους παρασκευής των προϊόντων, όπως κουζίνα και μπουφέ ενώ ο σκοπός του είναι να ενημερώνει μόνο για την άφιξη ενός νέου τραπεζιού.

Tableprops: Κώδικας Vue

```
tableprops : function(event){
    let i;
    let length = this.tables.length;
    let table;
    let role = this.role;

    for(i=0; i<length; i++){
        if(event.target.name == this.tables[i].tnumber){
            table = this.tables[i];
            break;
        }
    }

    this.currenttable = table;

    //If new table is pressed then mark it as seen (feature only available to kitchen and boufet)
    if(role == "kitchen" || role == "boufe"){
        if(table.tnew[this.index] == true){
            axios.post('/public/settoseen/', {tablenumber : table.tnumber , whosaw : role})
            .catch(function (error) {
            // handle error
            console.log(error);
            });
        }
    }
    if(table.talert[this.index] == true){
        axios.post('/public/alertseen/', {tablenumber : table.tnumber , whosaw : role})
            .catch(function (error) {
            // handle error
            console.log(error);
            });
    }

    this.props = this.toggleorset(this.props, role);

    if(this.mode){
        this.mode = this.toggleorset(this.mode);
    }
},
```

Εικόνα 3.48

Η μέθοδος αρχικά θέτει την τιμή της μεταβλητής currenttable ίση με το αντικείμενο τύπου table που επιλέχθηκε, κάνοντας έλεγχο με βάση τον αριθμό τραπεζιού. Στη συνέχεια εκτελείται έλεγχος για το ρόλο του συνδεδεμένου χρήστη καθώς και για το αν υπάρχει ειδοποίηση για την οποία δεν έχει ενημερωθεί. Εάν η τιμή talert[this.index] ή tnew[this.index] δεν είναι αληθής σημαίνει πως δεν χρειάζεται ενημέρωση του συγκεκριμένου χρήστη και έτσι δεν πραγματοποιείται κάποια ενέργεια πέραν της αλλαγής της μεταβλητής props σύμφωνα με το ρόλο έτσι ώστε να εμφανιστούν οι κατάλληλες επιλογές. Να αναφέρουμε ότι η μεταβλητή index είναι ένας αριθμός από 0 έως 3 και ορίζεται στην συνάρτηση requirelogin (Εικόνα 3.22), με βάση το ρόλο. Στην περίπτωση που ο χρήστης επιλέξει ένα τραπέζι για το οποίο έχει συμβεί αλλαγή

για την οποία δεν είναι ενήμερος, ή στην περίπτωση που επιλέγει ένα τραπέζι για πρώτη φορά, πραγματοποιούνται κατάλληλα αιτήματα ajax στο διακομιστή στις διευθύνσεις '/public/alertseen/' ή '/public/settoseen/' . Ως όρισμα δίνουμε τον αριθμό τραπεζιού και το ρόλο που ειδοποιήθηκε.

Οι αντίστοιχοι δρομολογητές κατά τον γνωστό τρόπο καλούν τις μεθόδους settoseen ή alertseen από το αρχείο backend.js . Τέλος οι συναρτήσεις στον κώδικα διακομιστή δεν επιστρέφουν κάποια τιμή παρά μόνο αλλάζουν τα κατάλληλα πεδία των πινάκων tnew ή talert στη θέση που ορίζεται από το ρόλο του χρήστη κάθε φορά.

Tableprops: Κώδικας Node js

```
exports.settoseen = function(tablenumber, role){
    var url = "mongodb://localhost:27017/";
    const client = new MongoClient(url, { useNewUrlParser: true });
    let index;

    client.connect((err => {
        const collection = client.db("verna").collection("tables");
        if(role == 'boufe'){
            try {
                collection.updateOne({tnumber : tablenumber} , { $set: { "tnew.0" : false } }  , write_concern = {'w':1});
            }catch(e){
                console.log(e);
            }
        }
        else if(role == 'kitchen'){
            try {
                collection.updateOne({tnumber : tablenumber} , { $set: { "tnew.1" : false } } , write_concern = {'w':1});
            }catch(e){
                console.log(e);
            }
        }
        client.close();
    }));
};
```

Εικόνα 3.49 Μέθοδος settoseen

```
exports.alertseen = function(tablenumber, role){
    var url = "mongodb://localhost:27017/";
    const client = new MongoClient(url, { useNewUrlParser: true });
    let index;

    client.connect((err => {
        const collection = client.db("verna").collection("tables");
        if(role == 'boufe'){
            try {
                collection.updateOne({tnumber : tablenumber} , { $set: { "talert.0" : false } }  , write_concern = {'w':1});
            }catch(e){
                console.log(e);
            }
        }
        else if(role == 'kitchen'){
            try {
                collection.updateOne({tnumber : tablenumber} , { $set: { "talert.1" : false } } , write_concern = {'w':1});
            }catch(e){
                console.log(e);
            }
        }
        else if(role == 'waiter'){
            try {
                collection.updateOne({tnumber : tablenumber} , { $set: { "talert.2" : false } } , write_concern = {'w':1});
            }catch(e){
                console.log(e);
            }
        }
        else if(role == 'owner'){
            try {
                collection.updateOne({tnumber : tablenumber} , { $set: { "talert.3" : false } } , write_concern = {'w':1});
            }catch(e){
                console.log(e);
            }
        }
        client.close();
```

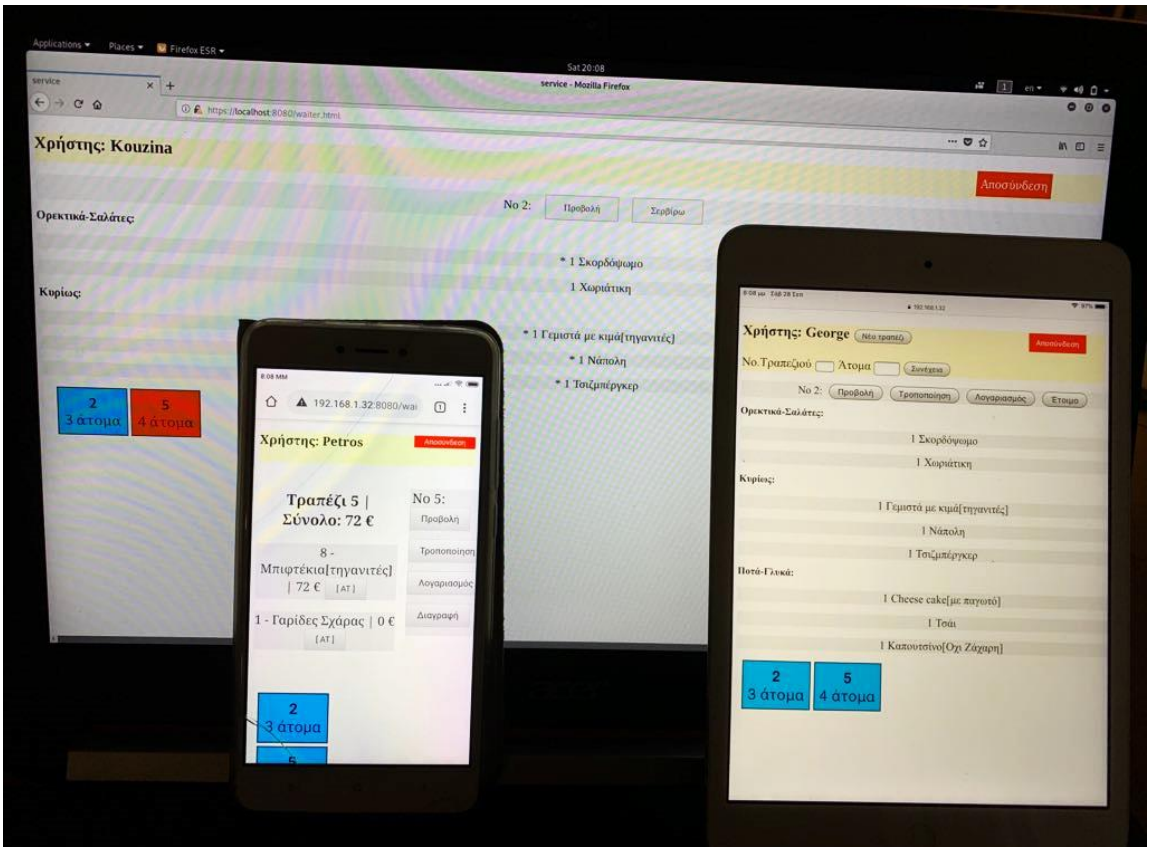
Εικόνα 3.50 Μέθοδος alertseen

3.2.3 Σχεδιασμός

Ο κώδικας css είναι τέτοιος ώστε ο σχεδιασμός της εφαρμογής να λαμβάνει υπόψιν, τόσο το είδος της συσκευής (Σταθερός/φορητός υπολογιστής, tablet, κινητό κτλ.), όσο και από τον προσανατολισμό της οθόνης σε περίπτωση φορητής συσκευής.

Σκοπός είναι να παρουσιάζουμε όσο περισσότερες χρήσιμες πληροφορίες, δίχως όμως να μπερδέψουμε το χρήστη γεμίζοντας την οθόνη του με πάρα πολλά στοιχεία. Επιλέξαμε ως εκ τούτου έναν απλό σχεδιασμό με διακριτά μέρη που δεν κουράζει.

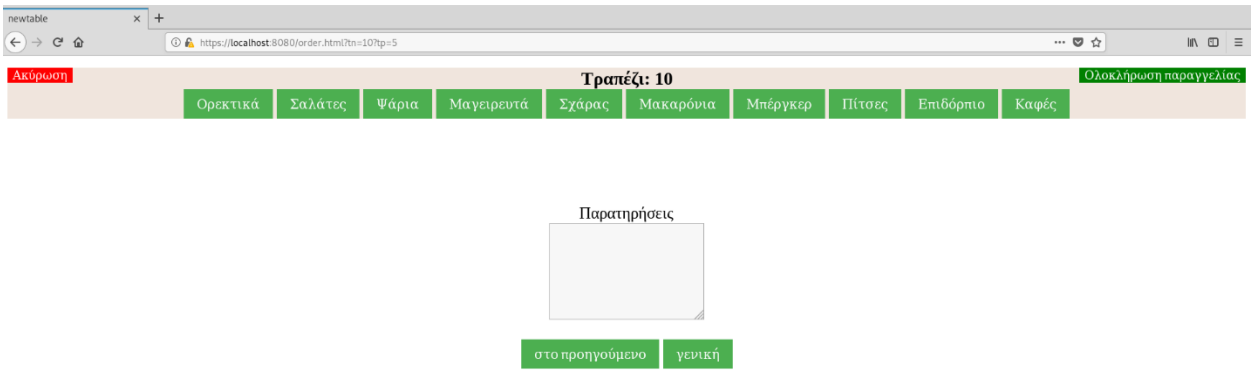
Για τη γενική όψη της κεντρικής οθόνης δεν χρησιμοποιήθηκαν παραπάνω από τρία χρώματα ενώ και αυτά ακόμη δεν είναι έντονα έτσι ώστε να μην αποσπούν άσκοπα την προσοχή του χρήστη. Τα μοναδικά σκοπίμως έντονα χρώματα «ντύνουν» στο κάτω μέρος της οθόνης τα τραπέζια, όταν έχει συμβεί κάποια αλλαγή και θέλουμε ο χρήστης να ειδοποιηθεί.



Εικόνα 3.48 Responsive CSS

3.3 Οθόνη παραγγελίας

3.3.1 Σύνοψη λειτουργίας από την οπτική του χρήστη



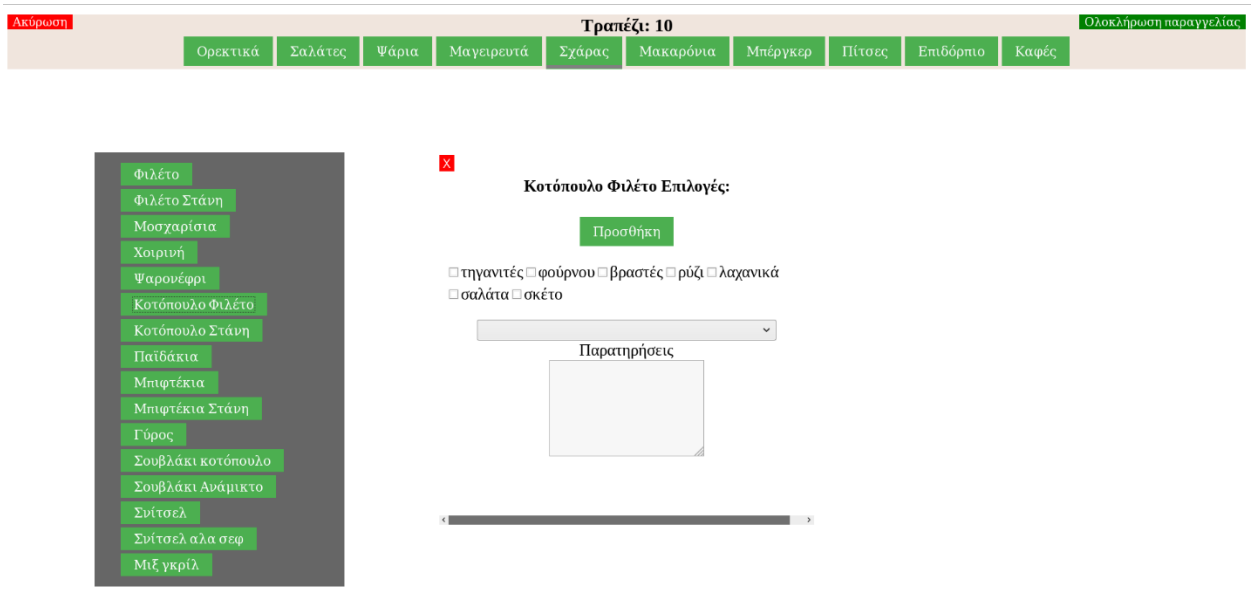
Εικόνα 3.49

Στην οθόνη παραγγελίας μπορούμε να εισέλθουμε έχοντας ρόλο “waiter”, με τη δημιουργία νέου τραπεζιού ή τροποποίηση, καθώς και με ρόλο “owner” με τροποποίηση. Στο επάνω μέρος εμφανίζονται οι κατηγορίες του καταλόγου, σε μορφή οριζόντιας λίστας. Ακόμη αναγράφεται ο αριθμός τραπεζιού ενώ υπάρχουν και οι λειτουργίες ακύρωση και ολοκλήρωση παραγγελίας. Στο κέντρο της οθόνης μπορούμε να εισάγουμε παρατηρήσεις είτε γενικές, είτε στο τελευταίο είδος που έχει εισαχθεί στην παραγγελία.

Πατώντας σε μια από τις κατηγορίες, εμφανίζονται τα αντίστοιχα προϊόντα.

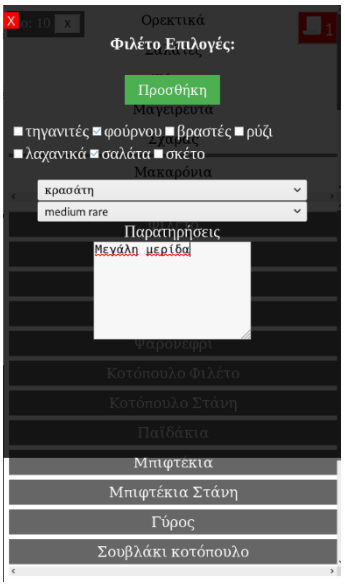
Εικόνα 3.50

Η επιλογή οποιουδήποτε προϊόντος έχει ως αποτέλεσμα είτε την άμεση προσθήκη του στην παραγγελία, είτε στις περισσότερες περιπτώσεις την εμφάνιση των διαθέσιμων επιλογών του.

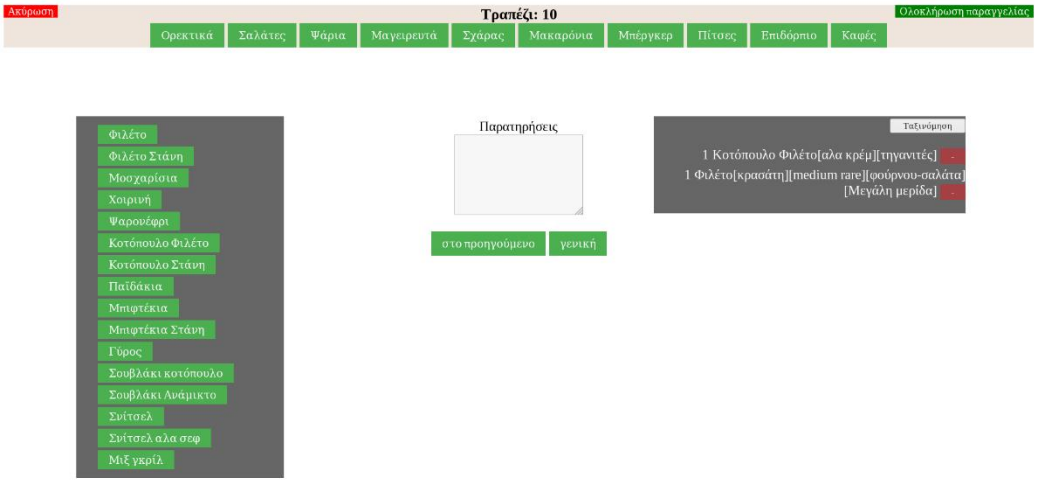


Εικόνα 3.51 Επιλογές Desktop

Αφού επιλέξουμε τις παραμέτρους όπως επιλογή γαρνιτούρας, σάλτσα, ψήσιμο κ.α., μπορούμε να πληκτρολογήσουμε μια παρατήρηση στο αντίστοιχο πεδίο αν το επιθυμούμε. Επιλέγοντας προσθήκη, το αντικείμενο προστίθεται στην παραγγελία μαζί με τις επιλογές μας.



Εικόνα 3.52 Οθόνη παραγγελίας και επιλογές σε mobile συσκευή.



Εικόνα 3.53

Όταν έχουμε προσθέσει όλα τα επιθυμητά προϊόντα πατάμε ολοκλήρωση παραγγελίας ώστε να ολοκληρωθεί η δημιουργία/τροποποίηση του τραπεζιού και να επιστρέψουμε στην κεντρική οθόνη.

3.3.2 Κώδικας οθόνης παραγγελίας

Σε αυτό το κεφάλαιο θα αναλυθούν οι περισσότερες από τις λειτουργίες της οθόνης παραγγελιών όπως: Η δομή των αντικειμένου τύπου food, η δομή των αντικειμένων τύπου table, οι παρατηρήσεις, ο τρόπος υπολογισμού των επιπροσθέτων ειδών, οι διάφοροι αυτοματισμοί που λαμβάνουν χώρα κατά την τροποποίηση παραγγελίας καθώς και η ταξινόμηση των ειδών σε «πρώτα», «δεύτερα/κυρίως» και «υπόλοιπα».

Να αναφέρουμε ότι επιλέχθηκε σκοπίμως η πλειοψηφία των υπολογιστικών ενεργειών που δεν σχετίζονται με τη βάση δεδομένων, να αναλαμβάνονται από τη συσκευή του πελάτη και όχι από τον διακομιστή. Μερικοί από τους λόγους για τους οποίους επιλέχθηκε αυτή η πορεία ανάπτυξης είναι, η σχεδόν άμεση ανταπόκριση της εφαρμογής στις ενέργειες του χρήστη δίχως τη συνεχή ανάγκη ανταλλαγής δεδομένων μέσα στο δίκτυο, η αποφυγή υπερφόρτωσης της συσκευής του διακομιστή και συνεπώς καθυστέρησης της εφαρμογής και ακόμη, το γεγονός πως στις μέρες μας, έως και οι φορητές συσκευές κατέχουν παραπάνω από αρκετή επεξεργαστική ισχύ ώστε να πληρούν με άνεση τις ανάγκες τις παρούσας εφαρμογής.

Αρχικές λειτουργίες

```
585     created : function(){
586         this.reqlogin();
587         //table details (hardcode becuse we know the url structure)
588         let match;
589         match = window.location.href.match(/tn=(\w+)/);
590         if(match==null){
591             match = window.location.href.match(/ch=(\w+)/);
592             this.tablenumber = match[1];
593             this.getorder();
594         }
595         else{
596             this.tablenumber = match[1];
597             match = window.location.href.match(/tp=(\d+)/);
598             this.peoplenumber = match[1];
599         }
600     }
601 })
```

Εικόνα 3.54

Εντός του created component, εντοπίζουμε για δεύτερη φορά τη μέθοδο reqlogin (Εικόνα 3.22) ενώ επιπλέον, υπάρχει και κώδικας για λεκτική ανάλυση της μπάρας διεύθυνσης url.

Αυτή η ανάλυση γίνεται για να λάβουμε από τη διεύθυνση στοιχεία όπως, τον αριθμό τραπεζίου και τον αριθμό ατόμων (Εικόνα 3.54, γραμμή 589). Σε περίπτωση που βρεθεί στο url δομή που να αντιστοιχεί σε: '/**tn**=(tablenumber)/, τότε λαμβάνουμε και εισχωρούμε την τιμή που ακολουθεί το αναγνωριστικό tn στην τοπικής μεταβλητή match και στη συνέχεια στη μεταβλητή tablenumber του data component της Vue (γραμμές 595-598). Σε περίπτωση όμως που δεν εντοπιστεί τέτοια δομή (γραμμή 590), τότε συμπεραίνουμε πως δεν εισήλθαμε στην οθόνη παραγγελίας από τη λειτουργία 'Νέο τραπέζι', άλλα έχουμε τροποποίηση υπάρχουσας παραγγελίας και άρα θα λάβουμε την τιμή του αριθμού τραπεζιού από την αλληλουχία χαρακτήρων '/**ch**=(tablenumber)/' (γραμμή 591), στην περίπτωση τροποποίησης καλούμε επιπλέον και τη μέθοδο getorder, ώστε να λάβουμε τα αντικείμενα της παραγγελίας προς τροποποίηση. Με αντίστοιχο τρόπο λαμβάνουμε και τον αριθμό ατόμων στην περίπτωση δημιουργίας νέου τραπεζιού.

Μέθοδος getorder (για τροποποίηση παραγγελίας) – Κώδικας Vue – Node js

```
490         getorder : function(){
491             let that = this;
492             axios.post('/public/getorder/', {number : this.tablenumber})
493             .then(function (response) {
494                 if(response.data != 'ordernotfound'){
495                     that.order = response.data.order;
496                     that.sortorder();
497                     if(that.starters.length != 0 || that.maincourses.length != 0){
498                         that.addon = true;
499                     }
500                     if(response.data.tgnotes != ''){
501                         that.hasGnotes = true;
502                         that.godnotes = response.data.tgnotes;
503                     }
504 
505                 }
506                 else
507                     console.log("order not present");
508 
509             })
510             .catch(function (error) {
511             // handle error
512             console.log(error);
513             });
514         },
```

Εικόνα 3.55 Μέθοδος getorder, κώδικας Vue js

Παρατηρήσεις:

- Η συνάρτηση έχει σκοπό την λήψη της παραγγελίας από τη βάση δεδομένων, χρησιμοποιώντας τον αριθμό τραπεζιού ως όρισμα για το αίτημα προς τη βάση. Μετά την επιτυχή ολοκλήρωση της έχουμε την ταξινομημένη παραγγελία του αντίστοιχου τραπεζιού ενώ η μεταβλητή addon του data component της Vue έχει τιμή 'true', έτσι ώστε να έχει γνώση η εφαρμογή πως εκτελείται τροποποίηση παραγγελίας.

- Εκτός από την παραγγελία επιστρέφονται επίσης και οι γενικές παρατηρήσεις από το πεδίο tgnotes (table general notes), του αντικειμένου τύπου table.

- Γραμμή 491: Μετά από την ολοκλήρωση της ασύγχρονης συνάρτησης ajax post request. Το αναγνωριστικό “this”, παύει να «αναφέρεται» στο αντικείμενο Vue, αλλά αναφέρεται πλέον στην ίδια την ασύγχρονη συνάρτηση καθώς αποτελεί αυτή αποτελεί το parent component της ανώνυμης συνάρτησης που έπεται (γραμμή 493). Για αυτό το λόγο χρησιμοποιούμε την τοπική μεταβλητή “that” και της δίνουμε την αρχική τιμή του this. Η μεταβλητή that πλέον αποτελεί δείκτη προς το αντικείμενο Vue και μας επιτρέπει την προσπέλαση των μεθόδων και μεταβλητών του ακόμη και εντός της συνάρτησης then.

```
230 exports.getorder = function(number){
231     const dbpromise = new Promise((resolve, reject) => {
232         var url = "mongodb://localhost:27017/";
233         const client = new MongoClient(url, { useNewUrlParser: true });
234 
235         client.connect((err => {
236             const collection = client.db("verna").collection("tables");
237             collection.findOne({tnumber: number}, (err, result) => {
238                 if(err) throw err;
239                 let order;
240                 let notes;
241                 if(result != null){
242                     let starters = result.torder[0];
243                     let maincourses = result.torder[1];
244                     let rest = result.torder[2];
245                     notes = result.tgnotes;
246                     order = starters.concat(maincourses,rest);
247                 }
248                 else{
249                     resolve(null);
250                 }
251                 if (order) resolve({order,notes});
252                 else reject({Error: 'something went wrong'});
253 
254             });
255             client.close();
256         }));
257     });
258     dbpromise.then((err, result) => {
259         if (err) console.log(err);
260             return result;
261     });
262     return dbpromise;
263 };
```

Εικόνα 3.56 Μέθοδος getorder, κώδικας Node Js

- Παρατηρήσεις

Η διαδικασίες για την λήψη δεδομένων από τη βάση δεδομένων είναι παρόμοιες με προηγούμενα παραδείγματα.

Κατάλογος και επιλογές ολοκλήρωσης και ακύρωσης παραγγελίας – Κώδικας HTML

```
<div id='mainmenu'>
    <div id="toppage">
        <h3 style="height:5px;">Τραπέζι: {{tablenumber}}<button id="logb" v-on:click='window.history.back();'>Ακύρωση</button>
        <button id="placeorderb" v-on:click='completeorder'>Ολοκλήρωση παραγγελίας</button>
        </h3>
    </div>
    <!--Horizontal menu category list-->
    <div id = "page">
        <button class='menubutton' name='starters' v-on:click='getfood'>Ορεκτικά</button>
        <select style="width:auto" name="starterlist" v-if="isStarters" v-on:change='getfood'>
            <option value="all">Όλα</option>
            <option value="hot">Ζεστά</option>
            <option value="cold">Κρύα</option>
            <option value="soup">Σούπες</option>
        </select>
        <button class='menubutton' name='salads' v-on:click='getfood'>Σαλάτες</button>
        <button class='menubutton' name='fish' v-on:click='getfood'>Ψάρια</button>
        <button class='menubutton' name='cooked' v-on:click='getfood'>Μαγειρευτά</button>
        <button class='menubutton' name='grill' v-on:click='getfood'>Σχάρας</button>
        <button class='menubutton' name='spaghetti' v-on:click='getfood'>Μακαρόνια</button>
        <button class='menubutton' name='burgers' v-on:click='getfood'>Μπέργκερ</button>
        <button class='menubutton' name='pizza' v-on:click='getfood'>Πίτσες</button>
        <button class='menubutton' name='desserts' v-on:click='getfood'>Επιδόρπιο</button>
        <select name="dessertlist" v-if="isDesserts" v-on:change='getfood'>
            <option value="all">Όλα</option>
            <option value="sweets">Γλυκά</option>
            <option value="icecream">Παγωτά</option>
        </select>
        <button class='menubutton' name='coffee' v-on:click='getfood'>Καφές</button>
    </div>
</div>
```

Εικόνα 3.57 HTML κώδικας επάνω μέρους οθόνης παραγγελιών, ορατός σε Desktop συσκευές και tablet.

```
<!--MOBILE-->
<div id = "pagemobile">
        <span id="tableinfo">
            No: {{tablenumber}}
            <button style="width: 40px;height:30px; font-size: 15px; color: black;" v-on:click='window.history.back();'>X</button>
        </span>
        <button class='menubutton' name='starters' v-on:click='getfood'>Ορεκτικά
        <select style="width:auto" name="starterlist" v-if="isStarters" v-on:change='getfood'>
            <option value="all">Όλα</option>
            <option value="hot">Ζεστά</option>
            <option value="cold">Κρύα</option>
            <option value="soup">Σούπες</option>
        </select></button>
        <button class='menubutton' name='salads' v-on:click='getfood'>Σαλάτες</button>
        <button class='menubutton' name='fish' v-on:click='getfood'>Ψάρια</button>
        <button class='menubutton' name='cooked' v-on:click='getfood'>Μαγειρευτά</button>
        <button class='menubutton' name='grill' v-on:click='getfood'>Σχάρας</button>
        <button class='menubutton' name='spaghetti' v-on:click='getfood'>Μακαρόνια</button>
        <button class='menubutton' name='burgers' v-on:click='getfood'>Μπέργκερ</button>
        <button class='menubutton' name='pizza' v-on:click='getfood'>Πίτσες</button>
        <button class='menubutton' name='desserts' v-on:click='getfood'>Επιδόρπιο
        <select name="dessertlist" v-if="isDesserts" v-on:change='getfood'>
            <option value="all">Όλα</option>
            <option value="sweets">Γλυκά</option>
            <option value="icecream">Παγωτά</option>
        </select>
        </button>
        <button class='menubutton' name='coffee' v-on:click='getfood'>Καφές</button>
</div>
```

Εικόνα 3.58 HTML κώδικας επάνω μέρους οθόνης παραγγελιών, ορατός σε κινητά με "portait" προσανατολισμό οθόνης

Ο παραπάνω κώδικας είναι κυρίως υπεύθυνος για την εμφάνιση του καταλόγου του καταστήματος σε μορφή οριζόντιας λίστας. Πατώντας επάνω σε οποιοδήποτε είδος καλείται η μέθοδος getfood, που λαμβάνει τα προϊόντα του καταλόγου από τη βάση δεδομένων, αναλόγως με την κατηγορία. Στον κώδικα συμπεριλαμβάνονται επίσης οι λειτουργίες της ακύρωσης, η οποία εκτελεί μια απλή ανακατεύθυνση στην προηγούμενη σελίδα, καθώς και της ολοκλήρωση παραγγελίας που θα αναλυθεί αργότερα.

Μέθοδος getfood (για λήψη προϊόντων από τον κατάλογο) – Κώδικας Vue js και Node js

```
189         getfood : function(event){
190             //Color selected button to highlight choice
191             if(this.previousEvent == 0){
192                 event.target.style.borderBottom = "thick solid grey";
193             }else{
194                 this.previousEvent.style.borderBottom = "none";
195                 event.target.style.borderBottom = "thick solid grey";
196             }
197             this.previousEvent = event.target;
198 
199             //Clear previous options
200             if(this.hasOptions){
201                 this.resetvalues();
202             }
203             if(event.target.name == "starterlist" || event.target.name == "dessertlist"){
204                 if(event.target.value != "all"){
205                     this.type = event.target.value;
206                 }
207                 else{
208                     if(event.target.name == "starterlist")
209                         this.type = "starters";
210                     else if(event.target.name == "dessertlist")
211                         this.type = "desserts";
212                 }
213             }
214             else{
215                 this.type = event.target.name;
216                 if(this.type == "starters"){
217                     this.isStarters = true;
218                 }
219                 else if(this.type == "desserts"){
220                     this.isDesserts = true;
221                 }
222                 else{
223                     this.isStarters = false;
224                     this.isDesserts = false;
225                 }
226             }
227 
228             let that = this;
229             axios.post('/public/getfood/', {foodtype : this.type})
230             .then(function (response) {
231             // handle success
232             if(response.data != "foodnotfound"){
233                 //console.log(response.data);
234                 that.food = response.data;
235             }
236             else
237                 console.log("Not in the menu");
238             })
239             .catch(function (error) {
240             // handle error
241             console.log(error);
242             })
243         },
```

Εικόνα 3.58 Μέθοδος getfood, κώδικας Vue js

Παρατηρήσεις:

- Οι γραμμές 190 έως και 197 υπάρχουν για σχεδιαστικούς λόγους καθώς υπογραμμίζουν με γκρι γραμμή την επιλεγμένη κατηγορία από τον κατάλογο κάθε φορά.

- Εκτελούμε επαναφορά (Γραμμή 201), στις σε ορισμένες μεταβλητές στο data component και συγκεκριμένα στη μεταβλητή hasOptions, διότι θέλουμε κάθε φορά που αλλάζουμε κατηγορία να εξαφανίζονται όσα πιθανά γραφικά στοιχεία παραμετροποίησης ενός είδους που μπορεί να βρίσκονται στην οθόνη.

- Γραμμές 203 έως και 226: Στο συγκεκριμένο τμήμα κώδικα εκτελούνται διάφοροι έλεγχοι ώστε, πρώτον να λειτουργήσουν τα στοιχεία select της html (Εικόνα 3.54) τα οποία κατηγοριοποιούν περεταίρω τις κατηγορίες «Ορεκτικά» και «Επιδόρπιο» (κρύα- ζεστά και γλυκά-παγωτά αντίστοιχα) και δεύτερον για να λάβει τιμή η μεταβλητή type του data component, ή οποία χρησιμοποιείται αργότερα ως όρισμα για το αίτημα ajax που θα ακολουθήσει. Η εμφάνιση των δύο προαναφερθέντων στοιχείων select εξαρτάται από την τιμή των μεταβλητών “isStarters” και “isDesserts” του data component.

- Να αναφέρουμε όποια και τιμή να καταλήξει στην μεταβλητή type, θα είναι τέτοια ώστε να προβλέπεται από το πεδίο ftype των αντικειμένων τύπου food στη βάση δεδομένων, ώστε να επιστραφεί τελικώς ένας πίνακας με όλα τα φαγητά της κατηγορίας. (Αποθήκευση στον πίνακα food, μεταβλητή του Vue data component)

```
265 exports.getFood = function(type){
266     const dbpromise = new Promise((resolve, reject) => {
267         var url = "mongodb://localhost:27017/";
268         const client = new MongoClient(url, { useNewUrlParser: true });
269 
270         client.connect((err => {
271             const collection = client.db("verna").collection("menu");
272             collection.find({ftype: type}, (err, result) => {
273                 if(err) throw err;
274                 var fooditems = {};
275                 if(result != null){
276                     fooditems = result.toArray();
277                 }
278                 else{
279                     resolve(null);
280                 }
281                 if (fooditems) resolve(fooditems);
282                 else reject({Error: 'something went wrong'});
283 
284             });
285             client.close();
286         }));
287     });
288     dbpromise.then((err, result) => {
289         if (err) console.log(err);
290             return result;
291     });
292     return dbpromise;
293 };
```

Εικόνα 3.59 Μέθοδος getfood, κώδικας Node Js

Προϊόντα κατηγορίας (menu items) - Πεδίο παρατηρήσεων – Προϊόντα παραγγελίας (order items) – Κώδικας HTML

```
79          <div id="centerpage">
80              <div id="menuitems" v-if="food[0]">
81              <ul id="menulist">
82                  <li v-for="f in food">
83                      <button class='arbutton' :name='f.fname' v-on:click='addorder'>{{f.fname}}</button>
84                  </li>
85              </ul>
86              </div>
87              <div id="generaloptions" v-if="!hasOptions">
88                  Παρατηρήσεις<br>
89                  <textarea v-model='gnotes' name="gnotes" rows="6"></textarea><br><br>
90                  <button class='menubutton' name='notetoprevious' v-on:click='notetoprevious'>στο προηγούμενο</button>
91                  <button class='menubutton' name='generalnote' v-on:click='generalnote'>γενική</button>
92                  <h6 id="gnotes" v-if="hasGnotes">Σημειώσεις: {{godnotes}}</h6>
93              </div>
94              <div id="foodoptions" v-if="hasOptions">
95                  <button id="closebutton" v-on:click="resetvalues">X</button>
96                  <h4>{{foodtoadd.fname}} Επιλογές:</h4>
97                  <button class='menubutton' name='place' v-on:click='placeorder'>Προσθήκη</button>
98                  <ul v-for="opt in options[0]">
99                      <li><input type="checkbox" v-on:change='fillcheckboxes' :name="opt">{{opt}}</li>
100                 </ul><br><br><br>
101                 <select v-if="is2D" v-model='selected'>
102                     <option v-for="opt in options[1]" :name="opt">{{opt}}</option>
103                 </select>
104                 <select v-if="is3D" v-model='selected2'>
105                     <option v-for="opt in options[2]" :name="opt">{{opt}}</option>
106                 </select><br>
107                 Παρατηρήσεις<br>
108                 <textarea v-model='fnotes' name="fnotes" rows="6"></textarea><br><br>
109             </div>
110
111             <div id="orderitems" v-if="order[0]">
112             <button v-on:click="sortorder">Ταξινόμηση</button>
113             <ul id="orderlist">
114                 <li v-for="o in order" v-if="o.ftype">
115                     {{o.quant}} {{o.fname}} <button class='orderremove' :name='o.fname' v-on:click='remorder'>-</button>
116                 </li>
117             </ul>
118             </div>
```

Εικόνα 3.60

Παραπάνω εντοπίζεται ο κώδικας, υπεύθυνος για:

- Την εμφάνιση των προϊόντων του καταλόγου ανά κατηγορία (γραμμές 80-86)

- Το πεδίο παρατηρήσεων με λειτουργίες «Γενική παρατήρηση ή «παρατήρηση στο προηγούμενο». (γραμμές 87-92)

- Το πεδίο επιλογών με διάφορα στοιχεία html όπως “checkboxes” και “select” που αλλάζουν δυναμικά, αναλόγως με το είδος των επιλογών κάθε προϊόντος (γραμμές 94-109)

- Την εμφάνιση των προϊόντων που έχουν εισαχθεί στην παραγγελία (κατακόρυφη λίστα). (γραμμές 111-118)

Μέθοδοι προσθήκης στην παραγγελία: Κώδικας Vue Js

Οι μέθοδοι που εκτελούνται όταν επιλέγεται η προσθήκη ενός είδους στη παραγγελία είναι οι εξής:

Μέθοδος Addorder

```
addorder : function(event){
    this.resetvalues();
    this.unfillcheckboxes();
    //Grab object
    for(let i = 0; i < this.food.length; i++){
        if(this.food[i]['fname'] == event.target.name){
            this.foodtoadd = this.food[i];
            break;
        }
    }
    //Check has options
    this.hasOptions = this.checkoptions(this.foodtoadd);
    if(this.hasOptions){
        return;
    }

    //Add to order
    this.placeorder();
},
```

Εικόνα 3.61

1) Αρχικοποίηση ορισμένων μεταβλητών του data component (resetvalues), προς επανάχρηση.

2) Αρχικοποίηση του πίνακα με τα επιλεγμένα πεδία σε είσοδο checkbox προς πιθανή εκ ‘νέου χρήση του.

3) Έλεγχος του πίνακα food για την εύρεση του αντικειμένου που επιλέχθηκε και εισχώρηση του πεδίου ονόματος ‘fname’ στη μεταβλητή foodtoadd του data component.

4) Κλήση της μεθόδου checkoptions, για να εμφανιστούν οι επιλογές του προϊόντος στην οθόνη (αν διαθέτει).

```
resetvalues : function(){
    this.hasOptions = false;
    this.is2D = false;
    this.is3D = false;
    this.selected = '';
    this.selected2 = '';
    this.fnotes = '';
    this.gnotes = '';
    this.options[0] = [];
    this.options[1] = [];
    this.options[2] = [];
    this.foodtoadd = [];
    this.checkboxes = [];
}
```

Εικόνα 3.62 Μέθοδος resetvalues

```
unfillcheckboxes : function(){
    let inputs, index;

    inputs = document.getElementsByTagName('input');
    for (index = 0; index < inputs.length; ++index) {
        if(inputs[index].checked){
            inputs[index].checked = false;
        }
    }
},
```

Εικόνα 3.63 Μέθοδος unfillcheckboxes

- Αν το προϊόν διαθέτει επιλογές, εκτελούμε έξοδο από τη συνάρτηση καθώς απαιτούνται επιπλέον ενέργειες πριν την τελική προσθήκη.

- Αν το προϊόν δεν διαθέτει επιλογές, τότε γίνεται κλήση της μεθόδου placeorder για να εκτελεστεί η τελική προσθήκη στην παραγγελία.

Μέθοδος checkoptions (Ελεγχος για επιλογές)

```
329     checkoptions : function(f){
330         //Rest of the options have to be global variables so that placeorder can increase the food price accoordingly
331         let garnishes = ["τηγανιτές", "φούρνου", "βραστές", "ρύζι", "λαχανικά" , "σαλάτα", "σκέτο"];
332         let cooking = ["blue", "rare", "medium rare", "medium well", "well done"];
333 
334         if(f.ftype[0] == "starters" || f.ftype == "salads"){
335             if(f.foptions && (f.foptions[f.foptions.length-1] != "για δεύτερο"))
336                 f.foptions.push("για δεύτερο");
337             else if(!f.foptions)
338                 f.foptions = ["για δεύτερο"];
339         }
340         if(f.foptions){
341             if(f.foptions[0] == "garnish" && f.foptions[1] == "sauce" && f.foptions[2] == "cooking"){
342                 this.is2D = true;
343                 this.is3D = true;
344                 this.options[0] = garnishes;
345                 this.options[1] = this.sauces;
346                 this.options[2] = cooking;
347             }
348             else if(f.foptions[0] == "garnish" && f.foptions[1] == "sauce"){
349                 this.is2D = true;
350                 this.options[0] = garnishes;
351                 this.options[1] = this.sauces;
352             }
353 
354             else if(f.foptions[0] == "garnish" && f.foptions[1] == "cooking"){
355                 this.is2D = true;
356                 this.options[0] = garnishes;
357                 this.options[1] = cooking;
358             }
359 
360             else if(f.foptions[0] == "size" && f.foptions[1] == "ingr"){
361                 this.is2D = true;
362                 this.options[0] = this.ing;
363                 this.options[1] = this.sizes;
364             }
365 
366             else if(f.foptions[0] == "garnish"){
367                 this.is2D = false;
368                 this.options[0] = garnishes;
369             }
370 
371             else{
372                 this.is2D = false;
373                 this.options[0] = f.foptions;
374                 this.options[1] = "";
375             }
376 
377             return true;
378         }
379         else
380             return false;
381     },
```

Εικόνα 3.64

1) Αρχικοποίηση πινάκων “garnish” και “cooking” (γαρνιτούρα-ψήσιμο). Άλλοι πίνακες που φιλοξενούν τα επιπρόσθετα ήδη και τις σάλτσες έχουν αρχικοποιηθεί ως global μεταβλητές στο data component, ώστε να έχει πρόσβαση σε αυτές η μέθοδος placeorder για να εφαρμοστεί επιπλέον χρέωση.

2) Έλεγχος των πεδίων ftype και foptions του αντικειμένου food που δώθηκε ως όρισμα στη μέθοδο, έτσι ώστε να εμφανιστούν οι κατάλληλες επιλογές. Οι επιλογές είναι συνήθως αποθηκευμένες σε μεταβλητές στη συσκευή πελάτη (γραμμές 331-332), ενώ στη βάση δεδομένων φυλάσεται μόνο μια λέξη κλειδί ώστε να γίνει αργότερα συσχέτιση.

Σπανιότερα όταν οι επιλογές είναι λίγες σε αριθμό, τότε τις αποθηκεύουμε όλες στη βάση δεδομένων. Για παράδειγμα αν το φαγητό είναι ορεκτικό (ftype=”starters”), τότε οι επιλογές που θα έχει συνήθως δεν υπερβαίνουν τις δύο σε αριθμό, το ίδιο ισχύει και για τις σαλάτες (ftype=”salads”). Σε αύτα τα αντικείμενα προσθέτουμε επιπλέον την επιλογή «για δέυτερο» που αν ενεργοποιηθεί επισημαίνει ότι τα αντικείμενα πρέπει να σερβιριστούν μαζί με τα κυρίως πιάτα και όχι ως «πρώτα». (γραμμές 334-337)

Στην περίπτωση που το είδος αντικειμένου είναι άλλο, τότε δύναται οι επιλογές του να είναι πολυάριθμες. Για παράδειγμα σε ένα αντικείμενο food με ftype=”grill”, κρατάμε στη βάση δεδομένων λέξεις κλειδιά όπως “cooking” , “garnish” και “sauce”. Στη σύνέχεια με δομές ελέγχου αντιστοιχίζουμε με τους κατάλληλους πίνακες που έχουμε ορίσει και που κρατάνε όλες τις επιλογές και τις εμφανιζουμε στην οθόνη. Οποιαδήποτε άλλη παραμετροποίηση μπορεί να πραγματοποιηθεί χειροκίνητα μέσα από μια γραπτή παρατήρηση, στο πεδίο παρατηρήσεων.

Οι πίνακες που έχουν δημιουργηθεί συνολικά είναι: [sauces] για τις σάλτσες, [ing] για τα υλικά , [sizes] για τα μεγέθη, [garnish] για τις γαρνιτούρες , [cooking] για το ψήσιμο.

3) Γραμμές 340-378: Ελέγχουμε όλους τους πιθανούς συνδιασμούς λέξεων κλειδιών του πεδίου f.foptions και εμφανίζουμε τις κατάλληλες επιλογές. Οι μεταβλητές is2d, is3D είναι τύπου boolean και είναι υπεύθυνες για την εμφανιση ή όχι πεδίων select στην html (Εικόνα 3.60, γραμμές 101-106).

4) Η μεταβλητή options αποτελεί πίνακα στο data component και λειτουργεί με τον εξής τρόπο. Τα αντικείμενα του options[0] χρησιμοποιούνται πάντα για να γεμίσουν στοιχεία checkbox της html (Εικόνα 3.60, γραμμές 98-100). Τα αντικείμενα του options[1], γεμίζουν το πρώτο στοιχείο select, εφόσον υπάρχουν και η μεταβλητή is2D είναι αληθής (Εικόνα 3.60, γραμμές 101-103). Τέλος τα αντικείμενα του options[2], γεμίζουν το δεύτερο στοιχείο select, εφόσον υπάρχουν και η μεταβλητή is3D είναι αληθής (Εικόνα 3.60, γραμμές 104-106).

Μέθοδος placeorder (Τελικη προσθήλη στην παραγγελία):

```
263         placeorder : function(){
264             let fpe = this.foodtoadd.fname; //fpe = food plus extras
265             let foodtype = this.foodtoadd.ftype;
266             let finalprice = this.foodtoadd.fprice;
267             if(this.selected && (this.selected != '')){
268                 fpe += "[" + this.selected + "]";
269                 if(this.sauces.includes(this.selected)){
270                     if(this.selected == "χωρίς σάλτσα"){
271 
272                     }else{
273                         finalprice += 1.5;
274                     }
275                 }
276                 if(this.foodtoadd.ftype == 'pizza'){
277                     if(this.selected == "μεγάλη"){
278                         finalprice += 1;
279                         if(this.foodtoadd.fname.includes("Χωριάτικη"))
280                             finalprice += 1;
281                     }
282                 }
283             }
284             if(this.selected2 && (this.selected2 != '')){
285                 fpe += "[" + this.selected2 + "]";
286             }
287             if(this.checkboxes.length!=0){
288                 fpe += "[";
289                 for(let i=0; i < this.checkboxes.length; i++){
290                     fpe += this.checkboxes[i] + "-";
291                     if(this.ing.includes(this.checkboxes[i])){
292                         finalprice += 0.5;
293                     }
294                     else if(this.checkboxes[i] == "με παγωτό"){
295                         finalprice += 1;
296                     }
297                     else if(this.checkboxes[i] == "με μπείλις"){
298                         finalprice += 2.5;
299                     }
300                     else if(this.sizes.includes(this.checkboxes[i])){
301                         //Salad
302                         if(this.foodtoadd.ftype == "salads"){
303                             finalprice -= 2;
304                         }
305                     }
306                 }
307                 fpe = fpe.slice(0,-1);
308                 fpe += "]";
309             }
310             if(this.fnotes != ''){
311                 fpe += "[" + this.fnotes + "]";
312             }
313             if(this.addon){
314                 fpe += ' [Επιπρόσθετο]'
315             }
316 
317             for(let i = 0; i < this.order.length; i++){
318                 if(this.order[i]['fname'] == fpe){
319                     this.order[i]['quant'] += 1;
320                     this.orderitems++;
321                     return;
322                 }
323             }
324             this.orderitems++;
325             this.order.push({'fname' : fpe , 'quant' : 1, 'ftype' : foodtype , fprice : finalprice});
326             //Inform basket has new item in case of mobile view
327             document.getElementById("basket").style.backgroundColor = "red";
328         },
```

Εικόνα 3.66

1) Αρχικοποίηση των τοπικών μεταβλητών (γραμμές 264-266).

- fpe (food plus extras) με αρχική τιμή το πεδίο fname του αντικειμένου foodtoadd (Εικόνα 3.61), πρακτικά το όνομα του προϊόντος.

- foodtype, με τιμή το πεδίο ftype του αντικειμένου foodtoadd.

- finalprice, με τιμή το πεδίο fprice του αντικειμένου foodtoadd (Αρχική τιμή του προϊόντος).

2) Γραμμές 267-315: Προσάρτηση στη μεταβλητή fpe των επιπλέον ειδών-επιλογών και παρατηρήσεων ανάλογα με τις επιλογές του χρήστη από τα πεδία input(checkbox) και select της html. Ακόμη, προσαυξήσεις στην τελική τιμή finalprice, που εξαρτώνται από τα επιπρόσθετα.

Στο τελικό όνομα του προϊόντος fpe, μπορεί να προστεθεί και αυτόματα η λέξη «Επιπρόσθετο», αν πραγματοποιούμε τροποποίηση παραγγελίας και η μεταβλητή addon είναι αληθής (Εικόνα 3.55, γραμμές 477-479).

3) Γραμμές 317-325: Τελική προσθήκη του προϊόντος στην παραγγελία. Το τελικό προϊόν μπορούμε να φανταστούμε ότι είναι ένα αντικείμενο τύπου food με πεδία food('fname'=fpe, 'fprice'=finalprice, 'ftype'=foodtype), από τέτοια αντικείμενα αποτελείται και ο πίνακας order, μεταβλητή του data component. Στο τέλος εκτελούμε έλεγχο στον πίνακα της υπάρχουσας παραγγελίας και αν βρεθεί αντικείμενο με το ίδιο όνομα, τότε αυξάνουμε το πεδίο quant(ποσότητα), του αντικειμένου αυτού κατά ένα. Αν το αντικείμενο προστίθεται για πρώτη φορά τότε το προσθέτουμε στον πίνακα order ως νέα καταχώρηση (γραμμή 325).

Πεδίο παρατηρήσεων – Μέθοδος γενικής παρατήρησης και παρατήρησης στο προηγούμενο

```
551         notetoprevious : function(){
552             if(this.order.length > 0 && this.gnotes != ''){
553                 this.order[this.order.length-1].fname += "[" + this.gnotes + "]";
554                 this.gnotes = '';
555             }
556         },
557         generalnote : function(){
558             if(this.gnotes != ''){
559                 this.hasGnotes = true;
560                 this.godnotes += ' ' + this.gnotes;
561                 this.gnotes = '';
562             }
563             else if(this.gnotes == '' && this.godnotes != ''){
564                 if(confirm("Καθαρισμός γενικών παρτηρήσεων;")){
565                     this.godnotes = '';
566                     this.hasGnotes = false;
567                 }
568             }
569         },
```

Εικόνα 3.67

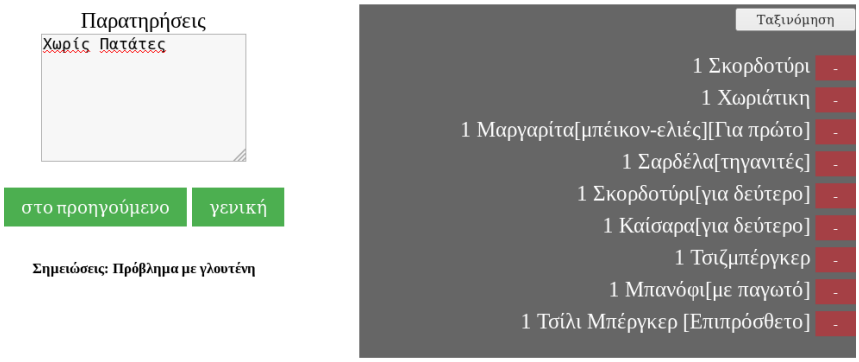
Παρατηρήσεις:

- Το η μεταβλητή godnotes του data component, θα αποτελέσει πεδίο του αντικειμένου table στην ολοκλήρωση παραγγελίας και θα ισούται με τη συνένωση όλων των γενικών σημειώσεων (gnotes).

- Η μεταβλητή hasGnotes του data component, καθιστά ορατό στην html το πεδίο που εμφανίζει τη γενική παρατήρηση κάτω από το πεδίο παρατηρήσεων, όταν είναι αληθής.

- Η λειτουργία παρατήρηση στο προηγούμενο προσθέτει στο πεδίο fname του αμέσως προηγούμενου στην παραγγελία είδους μια παρατήρηση.

Εικόνα 3.68 Γενική παρατήρηση, πριν και μετά την καταχώρηση της



Εικόνα 3.69 – Με την πίεση του πλήκτρου «στο προηγούμενο», το τελευταίο είδος της παραγγελίας θα είχε μορφή: 1 Τσίλι Μπέρκερ[Επιπρόσθετο][Χωρίς Πατάτες]

Μέθοδος αφαίρεσης από παραγγελία (remorder).

```
382         remorder : function(event){
383             if(this.order.length == 0)
384                 return;
385             let i;
386             for(i = 0; i < this.order.length; i++){
387                 if(this.order[i]['fname'] == event.target.name){
388                     if(this.addon){
389 
390                         this.godnotes += " Αφαιρέθηκε[" + this.order[i].fname + "]";
391                         this.hasGnotes = true;
392                     }
393                     let quant = this.order[i].quant;
394                     if(quant > 1){
395                         this.order[i]['quant'] -= 1;
396                         break;
397                     }
398                     else if(quant == 1){
399                         this.order.splice(i,1);
400                         break;
401                     }
402                 }
403             }
404             this.orderitems--;
405         },
```

Εικόνα 3.70

Στον παραπάνω κώδικα, σαρώνουμε τον πίνακα order, για να εντοπιστεί το αντικείμενο προς αφαίρεση. Αν η ποσότητα του αντικειμένου “quant” είναι μεγαλύτερη από ένα, τότε απλά αφαιρούμε ένα από το πεδίο “quant” (ποσότητα) του αντικειμένου food από τον πίνακα order, στην περίπτωση που η ποσότητα είναι ίση με ένα, τότε αφαιρούμε το αντικείμενο εξολοκλήρου από τον πίνακα order. Επιπλέον σε περίπτωση τροποποίησης παραγγελίας, προστίθεται αυτόματα με κάθε αφαίρεση μια γενική σημείωση (gnote), στην παραγγελία με μορφή: «Αφαιρέθηκε + fname»

Μέθοδος ολοκλήρωσης παραγγελίας και εισαγωγή τραπεζιού στη βάση (completeorder)

```
406         completeorder : function(){
407             if(this.order.length == 0){
408                     alert("Μια παραγγελία δεν μπορεί να είναι κενή");
409                     return;
410             }
411             if(confirm("Είστε σίγουρος/η")){
412                 //get order time
413                 let today = new Date();
414                 let time = today.getHours() + ":" + today.getMinutes() + ":" + today.getSeconds();
415                 let alerts = [true, true, true, true, ""];
416                 //sort order table
417                 this.sortorder();
418                 if(this.addon){
419                     alerts[4] += "addon";
420                 }
421
422                 this.table = {tnumber : this.tablenumber ,
423                 tpeople : this.peoplenumber , tnew : [true,true,false,false] ,
424                 torder : [this.starters, this.maincourses, this.rest],
425                 tgnotes : this.godnotes , ttime : time ,tstartersserved : [false, false,""] ,
426                 tmainsserved : [false, ""] , talert : alerts};
427                 /*
428                 talert: First three elements represent who saw amongst boufe, kitchen, waiter(s)
429                 and owner while last element (string) represents the alert itself,
430                 an example would be if a table had an addon and had asked for the bill then the table
431                 would have the following stucture [true, true, true, true "addonbill"],
432                 once someone is informed of the change then one or more of the boolean values will be changed.
433                 ----------------------------------------------------------------------------------------
434                 tnew: Elements represent who saw amongst owner, waiter(s), kitchen and boufet
435                 */
436
437                 axios.post('/public/completeorder/', {table : this.table})
438                 .then(function (response) {
439                     if(response.data == 'ok'){}
440                         window.history.back();
441
442                 })
443                 .catch(function (error) {
444                 // handle error
445                 console.log(error);
446                 });
447             }
448         },
```

Εικόνα 3.71 Complete order - Κώδικας Vue Js

Η συνάρτηση completeorder συλλέγει το σύνολο των στοιχείων που έχουν καταχωρηθεί κατά την παραγγελία και δομεί ένα νέο αντικείμενο τύπου table με τα πεδία που φαίνονται στον παραπάνω κώδικα στις γραμμές 422 έως 426. Εδώ συμπεριλαμβάνονται μεταξύ άλλων στοιχεία όπως, η κατάσταση του τραπεζιού στους πίνακες talert και tnew (Βλέπε σύστημα ειδοποιήσεων σελ.52) , η ώρα αποστολής της παραγγελίας καθώς και η ίδια η παραγγελία ταξινομημένη σε τρείς νέους πίνακες με τη χρήση της μεθόδου sortorder (γραμμή 4.17). Το νέο αντικείμενο τύπου table δίνεται ως όρισμα στη συνάρτηση ajax post αιτήματος (γραμμές 437-446) και καταχωρείται στη βάση δεδομένων όπως φαίνεται παρακάτω.

```
exports.completeorder = function(table){
    var url = "mongodb://localhost:27017/";
    const client = new MongoClient(url, { useNewUrlParser: true });

    client.connect((err => {
        const collection = client.db("verna").collection("tables");
        try {
            collection.updateOne(
              { "tnumber" : table.tnumber },
              {
                $set: {'torder' : table.torder, 'tgnotes' : table.tgnotes , 'talert' : table.talert},
                $setOnInsert: {'tpeople' : table.tpeople , 'tnew' : table.tnew , 'ttime' : table.ttime ,
                'tstartersserved' : table.tstartersserved , 'tmainsserved' : table.tmainsserved}
              },
              { w: "majority", wtimeout: 5000 , upsert : true}
            )

        }catch(e){
            console.log(e);
        }
        client.close();
    }));
};
```

Εικόνα 3.72 – Complete Order: Κώδικας Node Js

Μέθοδος ταξινόμησης παραγγελίας (sortorder)

```
sortorder : function(){
    let order = [];
    order = this.order;
    let sorted = [];
    let starters = [];
    let maincourses = [];
    let rest = [];
    this.orderitems = 0; //Recount them
    let length = order.length;
    let i;
    let temp;
    if(length == 0)
        return;

    for(i=0; i<length;i++){
        temp = order[i].ftype[0];
        if(temp == "starters" || temp == "salads"){
            if(order[i].fname.includes("για δεύτερο")){
                order[i].ftype.push("asmain");
                maincourses.push(order[i]);
            }
            else{
                starters.push(order[i]);
            }
            this.orderitems += order[i].quant;
        }
        else if(temp == "cooked" || temp == "grill" || temp == "fish" || temp == "spaghetti"
        || temp == "pizza" || temp == "burgers" || temp == "special")
        {
            if(order[i].fname.includes("πρώτο") || order[i].fname.includes("πρωτο")){
                order[i].ftype.push("asstarter");
                starters.push(order[i]);
            }
            else{
                maincourses.push(order[i]);
            }
            this.orderitems += order[i].quant;
        }
        else{
            rest.push(order[i]);
            this.orderitems += order[i].quant;
        }

    }
    this.starters = starters;
    this.maincourses = maincourses;
    this.rest = rest;
    sorted = starters.concat(maincourses, rest);
    this.order = sorted;
},
```

Εικόνα 3.73

Η μέθοδος sortorder χωρίζει τον πίνακα order σε τρείς επιμέρους πίνακες για καλύτερη οργάνωση και διαχείριση της παραγγελίας οι πίνακες αυτοί περιέχουν με τη σειρά, τα φαγητά που θα δοθούν στους πελάτες πρώτα (πίνακας starters), τα φαγητά που θα δοθούν δεύτερα (πίνακας maincourses) και όλα τα υπόλοιπα όπως τα ποτά ή τα επιδόρπια (πίνακας rest) (γραμμές 453-455). Τα προϊόντα ταξινομούνται ανάλογα με το είδος τους (πεδίο ftype), παράλληλα λαμβάνεται υπόψιν και η πιθανότητα ο πελάτης να επιθυμεί ένα προϊόν που κανονικά θα συμπεριλαμβανόταν στα «πρώτα» να μεταβεί στα «δεύτερα» και αντίστροφα (γραμμές 466 και 478).

Εικόνα 3.74 – Παραγγελία πριν και μετά την ταξινόμηση

Κώδικας: Βάση δεδομένων και τερματικό διακομιστή

Συλλογή menu – παραδείγματα αντικειμένου τύπου food:

Key	Value	Type
_id	ObjectId("5cd31ec31342f4155c431599")	ObjectId
fname	Φιλέτο	String
fprice	21	Double
ftype	Array[1]	Array
0	grill	String
foptions	Array[3]	Array
0	garnish	String
1	sauce	String
2	cooking	String
(18) ObjectId("5cd318631342f4155c43158c")	{ 4 attributes }	Document
_id	ObjectId("5cd318631342f4155c43158c")	ObjectId
fname	Μπριάμ	String
fprice	8	Double
ftype	Array[1]	Array
0	cooked	String
(19) ObjectId("5cd318631342f4155c43158b")	{ 5 attributes }	Document
_id	ObjectId("5cd318631342f4155c43158b")	ObjectId
fname	Κουνέλι	String
fprice	9.5	Double
ftype	Array[1]	Array
foptions	Array[1]	Array
0	garnish	String

Εικόνα 3.75

Συλλογή tables – παράδειγμα αντικειμένου τύπου table

Key	Value	Type
(1) ObjectId("5d9647f2d0b79a0624b78b0f")	{ 10 attributes }	Document
(2) ObjectId("5d96471dd0b79a0624b78abb")	{ 10 attributes }	Document
_id	ObjectId("5d96471dd0b79a0624b78abb")	ObjectId
tnumber	3	String
talert	Array[5]	Array
0	true	Bool
1	true	Bool
2	true	Bool
3	false	Bool
4	addon	String
tgnotes	Πρόβλημα με γλουτένη	String
tmainsserved	Array[2]	Array
tnew	Array[4]	Array
torder	Array[3]	Array
0	Array[3]	Array
1	Array[5]	Array
2	Array[1]	Array
tpeople	2	String
tstartersserved	Array[3]	Array
ttime	22:8:13	String

Εικόνα 3.76

```
POST /public/gettables/ 200 8.001 ms - 894
[
  {
    _id: 5d8f8d399d6c0634ed5b7bf9,
    tnumber: '2',
    talert: [ true, false, false, true, '' ],
    tgnotes: '',
    tmainsserved: [ false, '' ],
    tnew: [ true, false, false, false ],
    torder: [ [Array], [Array], [Array] ],
    tpeople: '3',
    tstartersserved: [ false, false, '' ],
    ttime: '19:41:29'
  }
]
POST /public/gettables/ 200 5.225 ms - 894
```

Εικόνα 3.77 - Αποτέλεσμα μεθόδου gettables στο τερματικό

```
POST /public/getorder/ 200 5.783 ms - 1246
[
  {
    _id: 5cd318631342f4155c431582,
    fname: 'Μουσακάς',
    fprice: 8.5,
    ftype: [ 'cooked' ],
    foptions: [ 'garnish' ]
  },
  {
    _id: 5cd318631342f4155c431583,
    fname: 'Παστίτσιο',
    fprice: 8.5,
    ftype: [ 'cooked' ],
    foptions: [ 'garnish' ]
  },
  {
    _id: 5cd318631342f4155c431584,
    fname: 'Γεμιστά με κιμά',
```

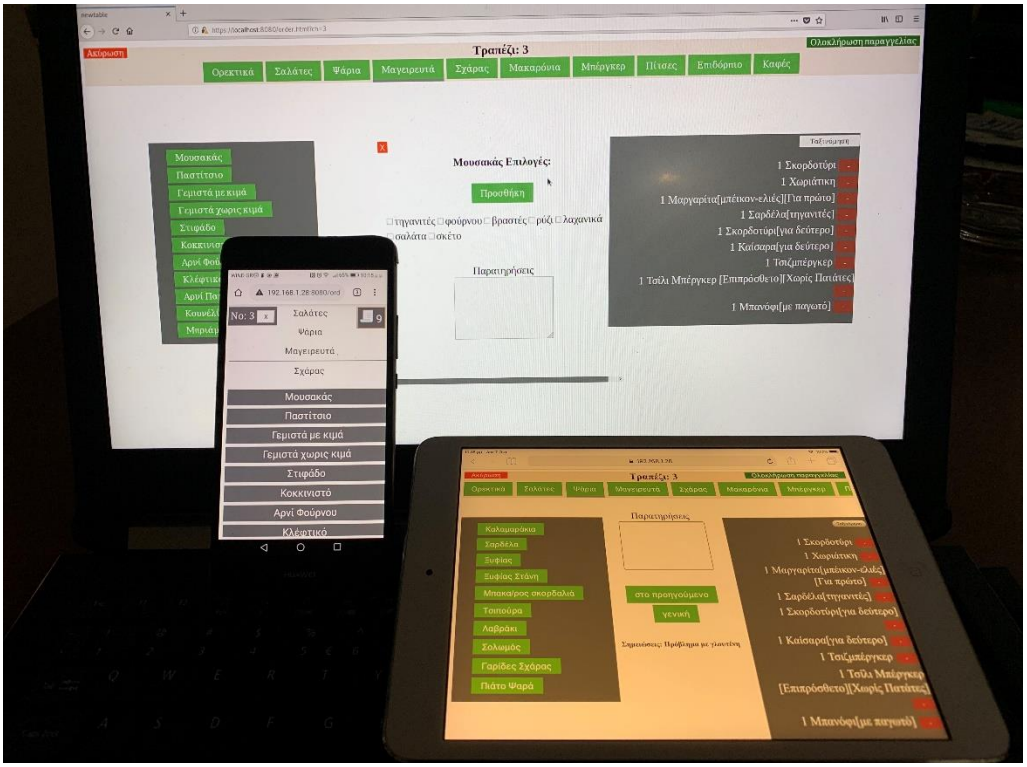
Εικόνα 3.78 – Αποτέλεσμα μεθόδου getfood στο τερματικό

3.3.3 Σχεδιασμός

Η οθόνη παραγγελίας αποτέλεσε σχεδιαστική πρόκληση, καθώς τα στοιχεία που χρειάζεται να εμφανίζονται στην οθόνη είναι πολυάριθμα, σε σχέση με τις προηγούμενες οθόνες. Σε γενικές γραμμές επιτεύχθηκε η διατήρηση του ίδιου σχεδιασμού σε όλες τις συσκευές, με εξαίρεση να αποτελούν τα κινητά σε λειτουργία portrait, όπου και χρειαζόταν ένας νέος σχεδιασμός και ορισμένα επιπλέον στοιχεία html που γίνονταν ορατά μόνο στη συγκεκριμένη περίπτωση. Τέτοια στοιχεία είναι: Μια νέα σελίδα απεικόνισης των περιεχομένων του καταλόγου και των προϊόντων της κάθε κατηγορίας, μια νέα σελίδα για την προβολή της παραγγελίας, ένα πλήκτρο που την ενεργοποιεί (καλάθι αγορών), καθώς και διάφορα πλήκτρα για την έξοδο από τα στοιχεία αυτά.

Ο σχεδιασμός για Σταθερούς υπολογιστές και tablet, είναι χρηστικός αφού περιέχει αρκετές πληροφορίες ώστε να ελαχιστοποιούνται οι απαραίτητες παρεμβάσεις του χρήστη για την εκάστοτε εργασία, χωρίς όμως η οθόνη να είναι ελλιπής. Τα χρώματα που επιλέχθηκαν είναι λίγα σε αριθμό και βοηθούν στο να υπάρχει αντίθεση μεταξύ των στοιχείων στην οθόνη κάνοντας τα ευανάγνωστα.

Ο σχεδιασμός για κινητά σε λειτουργία portrait, είναι μινιμαλιστικός, με μόνο τα απαραίτητα κάθε φορά στοιχεία να εμφανίζονται, η χρωματική παλέτα αυτή τη φορά περιορίζεται κυρίως σε τόνους του γκρι. Ο σχεδιασμός σε αυτήν την περίπτωση, έπρεπε να είναι τέτοιος ώστε να καθιστά εύκολο το χειρισμό με ένα χέρι.



Εικόνα 3.79

Κεφάλαιο 4ο - Μελλοντικές επεκτάσεις

Μερικές ιδέες για την επεκτασιμότητα της εφαρμογής είναι:

- Προσθήκη σελίδας διαχειριστή (Admin Page)

Μετά την αρχική διαμόρφωση της εφαρμογής για να καλύπτει συγκεκριμένα τις ανάγκες του εκάστοτε καταστήματος, θα ήταν καλό να δίναμε την δυνατότητα στους επιχειρηματίες να πραγματοποιούν αλλαγές που κανονικά θα απαιτούσαν παρέμβαση στον πηγαίο κώδικα και στη βάση δεδομένων, μέσα από μια σελίδα διαχειριστή στην οποία θα είχαν μόνο αυτοί πρόσβαση και ιδανικά θα φιλοξενούταν μόνο στη συσκευή τους, κυρίως για λόγους ασφαλείας. Αλλαγές όπως προσθήκη και αφαίρεση από τον κατάλογο, παραμετροποίηση στα στοιχεία του προσωπικού και άλλα, θα μπορούσαν να συμπεριλαμβάνονται σε αυτή την οθόνη.

- Προβολή στατιστικών στοιχείων για τους διάφορους ρόλους

Τα στατιστικά στοιχεία μπορούν να αποτελέσουν ένα χρήσιμο αλλά και συνάμα διασκεδαστικό στοιχείο που σχετίζονται με την εργασία μας. Πληροφορίες όπως, πόσος κόσμος πέρασε από το κατάστημα ανά ήμερα ή ανά σαιζόν. Πόσες παραγγελίες έχει εκτελέσει ο κάθε εργαζόμενος, μια λίστα με τα πιο δημοφιλή προϊόντα, τα έσοδα ανά μέρα άλλα και σε βάθος χρόνου σε μορφή γραφημάτων είναι στοιχεία που βοηθούν σε διάφορες πτυχές μια επιχείρηση.

- Νέος σχεδιασμός οθόνης για την κουζίνα και άλλους ρόλους παρασκευής των προϊόντων

Μια νέα οθόνη που θα ενεργοποιείται με κατάλληλη επιλογή των χρηστών στους ρόλους παρασκευής, θα επιτρέπει την πιο φιλική προβολή των παραγγελιών σε μορφή που θυμίζει σκοπίμως δελτία παραγγελιών τοποθετημένα το ένα δίπλα στο άλλο. Αυτό θα επιτρέπει την ταυτόχρονη προβολή πολλαπλών παραγγελιών, δίχως την ανάγκη για επιλογή των αντίστοιχων τραπεζιών και την αποκλειστική προβολή ενός.

- Υποστήριξη διαχωρισμού παραγγελιών
- Συμβατότητα με θερμικούς εκτυπωτές Bluetooth

- Διαχείριση αποθήκης και stock προϊόντων.

Δυνατότητα καταχώρησης ποσοτήτων προϊόντων και σταδιακής αφαίρεσης τους από το απόθεμα αυτόματα με κάθε παραγγελία. Προειδοποίηση έλλειψης προϊόντος πριν συμβεί.

Μακροπρόθεσμες επεκτάσεις θα μπορούσαν να είναι:

- Συμβατότητα με ταμειακές μηχανές

- Λειτουργία πληρωμής άμεσα από την εφαρμογή: Υποστήριξη PayPal και υπηρεσιών ηλεκτρονικής πληρωμής διαφόρων τραπεζών. Πιθανή εκμετάλλευση της τεχνολογίας NFC για ανέπαφες πληρωμές και χρήση της συσκευής μας ως τερματικό POS.

- Λειτουργία self-order: Κατασκευή ειδικού παραρτήματος της εφαρμογής για τους πελάτες του καταστήματος, στο όποιο θα φαίνεται ο κατάλογος και θα δίνεται η επιλογή απομακρυσμένης παραγγελίας (εντός του τοπικού όμως δικτύου).

- Απομακρυσμένες παραγγελίες εκτός τοπικού δικτύου: Επέκταση της λειτουργίας self order, με τη βοήθεια του διαδικτύου για παραγγελίες από οπουδήποτε. Δυνατότητα διανομής για καταστήματα που το υποστηρίζουν. Διατήρηση προφίλ πελατών και παροχή προσωποποιημένων προτάσεων με βάση τις προτιμήσεις τους καθώς και παροχή δώρων/επιβραβεύσεων.

Βιβλιογραφία – Διαδικτυακές συνδέσεις

Πρωτόκολλο Https

1. https://nodejs.org/en/knowledge/HTTP/servers/how-to-create-a-HTTPS-server/

2. https://robertheaton.com/2014/03/27/how-does-https-actually-work/

Μοντέλο πελάτη-εξυπηρετητή

3. http://edu.net.gr/index.php/sxediasmos-kai-anaptyksi-diadiktyakon-efarmogon/287-client-server-model/file.html

Βάσεις δεδομένων

4. https://opencourses.uoc.gr/courses/pluginfile.php/15366/mod_resource/content/3/2.%20%CE%95%CE%B9%CF%83%CE%B1%CE%B3%CF%89%CE%B3%CE%AE%20%CE%99%CE%99%20-%20%CE%A3%CF%87%CE%B5%CE%B4%CE%AF%CE%B1%CF%83%CE%B7%20%CE%BA%CE%B1%CE%B9%20%CE%91%CF%81%CF%87%CE%B9%CF%84%CE%B5%CE%BA%CF%84%CE%BF%CE%BD%CE%B9%CE%BA%CE%AE%20-%20%CE%9A%CE%B5%CE%AF%CE%BC%CE%B5%CE%BD%CE%BF.pdf

5. https://searchsqlserver.techtarget.com/definition/database/

6. https://www.pliroforiki-edu.gr/unit/ch0202-xaraktiristika-diaxeirisis-basis-dedomenon/

Πληροφορίες για την Javascript

7. https://en.wikipedia.org/wiki/JavaScript/

Ενδιάμεσο λογισμικό

8. http://pdplab.it.uom.gr/teaching/ince_2e_gr/Text/C3/Middleware_3.htm

Node Js Express module

9. https://expressjs.com/en/starter/hello-world.html

10. https://expressjs.com/en/starter/generator.html

11. https://docs.npmjs.com/misc/scripts

12. https://vegibit.com/express-js-beginner-tutorial/

13. https://www.geeksforgeeks.org/model-view-controllermvc-architecture-for-node-applications/

Κρυπτογράφηση και αλγόριθμοι κατακερματισμού

14. https://www.youtube.com/watch?v=4zahvcJ9glg/ (Part 1 and 2)

15. https://brilliant.org/wiki/rsa-encryption/

16. https://www.rootnetsec.com/hash-functions/

Crypto Module

17. https://nodejs.org/api/crypto.html

Πηγές Εικόνων

18. https://www.chegg.com/homework-help/describe-three-tier-client-server-model-chapter-11-problem-7rq-solution-9781305465114-exc

19. https://icons8.com/icons/set/order/