ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΝΙΚΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

# Διαδικτυακή Εφαρμογή Αναζήτησης και Κριτικής Εστιατορίων Βασισμένο σε Χάρτες Google

Διπλωματική Εργασία

**Κανελλόπουλος Χρυσοβαλάντης**

Εξεταστική επιτροπή:

Επ. Καθ. Αντώνιος Δεληγιαννάκης

Επ. Καθ. Μιχαήλ Λαγουδάκης

Επ. Καθ. Αικατερίνη Μανιά

**Οκτώβριος 2010**

Στην οικογένειά μου και στους πραγματικούς φίλους που με στηρίζουν όλα αυτά τα χρόνια

# Ευχαριστίες

Σε αυτό το σημείο θα ήθελα να ευχαριστήσω ορισμένα άτομα που συνέβαλλαν αποφασιστικά στην ανάπτυξη και συγγραφή της παρούσας εργασίας.

Καταρχήν θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή μου, κύριο Αντώνη Δεληγιαννάκη, για την ορθή και πολύτιμη καθοδήγησή του σε όλα τα στάδια υλοποίησης αυτής της διπλωματικής εργασίας. Επίσης θα ήθελα να ευχαριστήσω τον καθηγητή Μιχάλη Λαγουδάκη και την καθηγήτρια Κατερίνα Μανιά για τον χρόνο που αφιέρωσαν για την ανάγνωση και αξιολόγηση της παρούσας εργασίας και για τις εποικοδομητικές παρατηρήσεις τους.

Ιδιαίτερες ευχαριστίες στην κυρία Ξένια Αράπη, για την βοήθειά της στο στήσιμο του server που φιλοξένησε δοκιμαστικά την εφαρμογή μου.

Τέλος ευχαριστώ όλους τους γνωστούς και φίλους που αφιέρωσαν λίγο από τον χρόνο τους ώστε να δοκιμάσουν την εφαρμογή και να μου επισημάνουν σφάλματα και παραλήψεις τα οποία με βοήθησαν πολύ κατά την διαδικασία του debugging.

# Περίληψη

Στην παρούσα διπλωματική υλοποιήσαμε μια διαδικτυακή εφαρμογή παρουσίασης και κριτικής εστιατορίων. Εκμεταλλευτήκαμε το Google Maps API ώστε να μπορέσουμε να παρέχουμε στους χρήστες και χωρική πληροφορία. Στην εφαρμογή παρέχονται δυνατότητες αναζήτησης με βάση χωρικά δεδομένα με χρήστη του χάρτη Google, όπως και με συμβατικά κριτήρια που έχουν να κάνουν με το είδος της κουζίνας και με την συνολική βαθμολογία που έχει συγκεντρώσει κάθε εστιατόριο. Τα αποτελέσματα εμφανίζονται στον χάρτη και ταυτόχρονα σε αριθμημένη λίστα, με αντιστοίχιση των αριθμών της λίστας με τους αριθμούς των markers του χάρτη. Στους χρήστες παρέχεται η δυνατότητα σχολιασμού και βαθμολόγησης των εστιατορίων καθώς και η δυνατότητα ανταλλαγής προσωπικών μηνυμάτων.

Η υλοποίηση της εφαρμογής έγινε με χρήση τεχνολογίας Java Server Pages (JSP) και Java servlet ακολουθώντας την αρχιτεκτονική Model View Controller (MVC). Η αρχιτεκτονική αυτή μας επιτρέπει να συνδυάσουμε τις δύο παραπάνω τεχνολογίες ώστε να εκμεταλλευτούμε τα πλεονεκτήματα κάθε μίας. Το βασικό πλεονέκτημα των servlets είναι η χρήση java, με την πληθώρα από διαθέσιμες βιβλιοθήκες, για την διαχείριση των αιτημάτων HTTP, ενώ το πλεονέκτημα των JSP είναι η δυνατότητα παραγωγής διαφορετικών αρχείων εξόδου (HTML) από το ίδιο αρχείο JSP, ανάλογα με τις παραμέτρους που έχουν παραχθεί από την επεξεργασία του αιτήματος HTTP.

# Περιεχόμενα

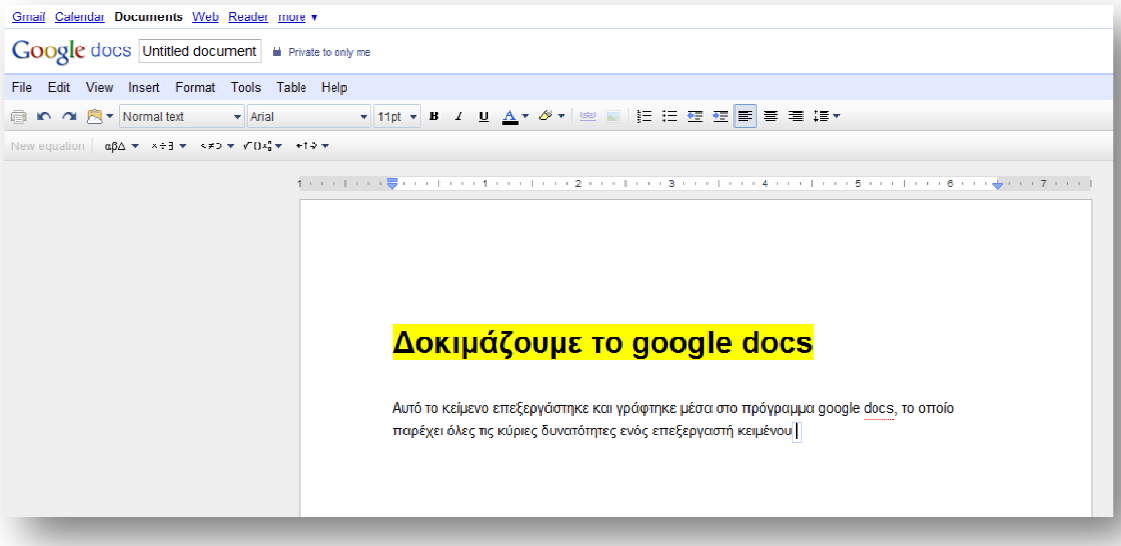
# Ευρετήριο εικόνων

# 1. Εισαγωγή

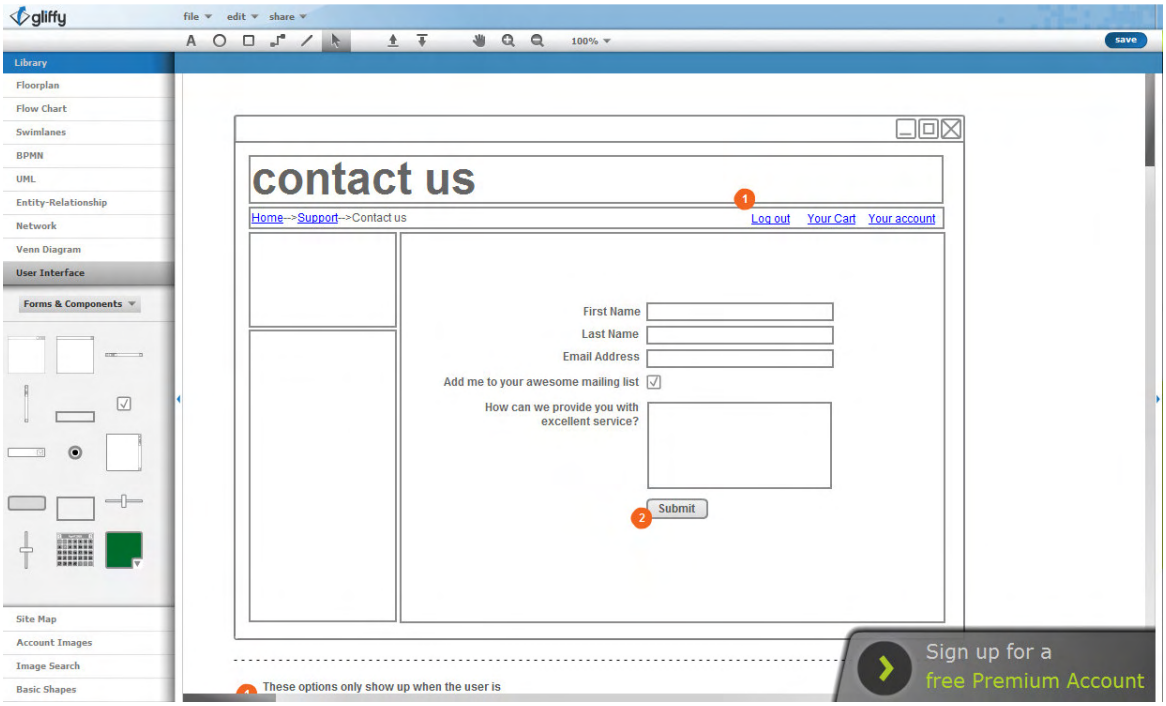
## 1.1. Το διαδίκτυο σήμερα

Στην σημερινή εποχή το διαδίκτυο έχει γίνει βασικό αγαθό-υπηρεσία στον αναπτυγμένο αλλά και αναπτυσσόμενο κόσμο. Η διείσδυσή του διαδικτύου παγκοσμίως τον Ιούνιο του 2010, ήταν στο 28,7% του πληθυσμού, καταγράφοντας αύξηση κατά 444,8% τα τελευταία 10 χρόνια, σύμφωνα με το Internet World Stats. Η αλλαγή, που έχει επιφέρει σε πάρα πολλούς τομείς της ζωής μας είναι τεράστια.

Η ευρεία πλέον διαθεσιμότητα των υπηρεσιών του διαδικτύου έχει μεταβάλλει τον τρόπο που χρησιμοποιούμε τους υπολογιστές. Το μεγαλύτερο ποσοστό τον χρηστών σήμερα χρησιμοποιεί τον υπολογιστή σαν ένα τερματικό του διαδικτύου. Για πολλούς ανθρώπους ένας υπολογιστής χωρίς πρόσβαση στο δίκτυο είναι άχρηστος. Αυτό οφείλεται στην μεγάλη πλέον διαθεσιμότητα εφαρμογών που δεν τρέχουν στον υπολογιστή μας, αλλά σε κάποιο απομακρυσμένο υπολογιστή του δικτύου. Η πρόσβαση σε αυτές τις εφαρμογές γίνεται μέσω του παγκοσμίου ιστού (world wide web), μέσω του φυλλομετρητή του υπολογιστή μας. Τέτοιες εφαρμογές είναι οι εφαρμογές Google Docs, Gmail, youTube, gliffy, ms webMessenger κτλ.




Εικόνα 1. Η online εφαρμογή επεξεργασίας κειμένου της Google




Εικόνα 2. Η online εφαρμογή δημιουργίας διαγραμμάτων Gliffy

## 1.2. Υπηρεσίες χαρτογράφησης μέσω διαδικτύου

Η άνθιση λοιπόν του διαδικτύου και των τεχνολογιών πληροφορικής, οδήγησε στην προσφορά διαδικτυακών υπηρεσιών που πριν ελάχιστα χρόνια ήταν διαθέσιμες μόνο σε κυβερνήσεις και ερευνητικούς οργανισμούς. Μια τέτοια υπηρεσία είναι και η *υπηρεσία χαρτογράφησης μέσω διαδικτύου (web mapping)*.

Με τον όρο *Web Mapping* αναφερόμαστε στην διαδικασία του σχεδιασμού, υλοποίησης, παραγωγής και διανομής χαρτών μέσω του παγκοσμίου ιστού ( World Wide Web) και των παραγώγων του [1].

Η χρήση του διαδικτύου ως μέσο διάθεσης των χαρτών μπορεί να θεωρηθεί σαν μία τεράστια εξέλιξη στην χαρτογραφία, στην οποία δημιουργεί πολλές νέες δυνατότητες όπως:

- *οι χάρτες πραγματικού χρόνου:* Αν ο χάρτης δημιουργείται αυτόματα από βάσεις δεδομένων, τότε μπορούν να εμφανίζονται πληροφορίες σχεδόν σε πραγματικό χρόνο. Μόλις τα δεδομένα γίνουν διαθέσιμα στην βάση, τότε αυτόματα εμφανίζονται και στον χάρτη που τραβάει πληροφορίες από αυτή.

- *η φθηνότερη διανομή των χαρτών μέσω του διαδικτύου:* Παλαιότερα έπρεπε να αγοράσεις τον χάρτη τυπωμένο, ή να αγοράσεις και να εγκαταστήσεις το απαραίτητο λογισμικό. Σήμερα αρκεί ένα απλό τερματικό που να παρέχει υπηρεσίες διαδικτύου.

- *συχνή ενημέρωση των δεδομένων, η οποία γίνεται φθηνότερη και ευκολότερη:* εφόσον τα δεδομένα αντλούνται κάθε φορά από κάποια κεντρική βάση, αρκεί απλά να ενημερωθεί αυτή. Δεν χρειάζεται να γίνει επέμβαση στο τερματικό του χρήστη.

- *δυνατότητα προσωποποιημένου περιεχομένου στον χάρτη:* Κάθε χρήστης μπορεί να επιλέγει τις πληροφορίες που θα εμφανίζονται στον χάρτη, όπως και να προσθέσει δικές του προσωπικές πληροφορίες

- *πολλαπλές πηγές δεδομένων:* Μπορούν να συνδυαστούν δεδομένα από διάφορες πηγές, ενώ πολλές φορές δίνεται και η δυνατότητα στο κοινό να προσθέτει δεδομένα.

- *ευκολότερος διαμοιρασμός γεωγραφικών πληροφοριών:* Κάθε χρήστης μπορεί να διανείμει προσωπικές ή μη γεωγραφικές πληροφορίες εύκολα λόγω της ευρείας διαθεσιμότητας των χαρτών σε όλους.

Εκτός βέβαια από τις πολλές νέες δυνατότητες, η ευρεία χρήση των υπηρεσιών χαρτογράφησης μέσω διαδικτύου εγείρει και διάφορα άλλα τεχνικά και ηθικά ζητήματα:

- *Ζητήματα αξιοπιστίας:* Η αξιοπιστία του διαδικτύου και των web servers δεν είναι ακόμα αρκετά καλή. Ειδικά αν ο χάρτης βασίζεται σε εξωτερικές, κατανεμημένες πηγές δεδομένων, πολλές φορές δεν είναι εγγυημένη η διαθεσιμότητα των πληροφοριών.

- *Ζητήματα ταχύτητας του δικτύου:* οι χάρτες μέσω διαδικτύου χρειάζονται αρκετά υψηλές ταχύτητες μεταφοράς δεδομένων.

- *Υψηλό κόστος γεωγραφικών δεδομένων:* Αν εξαιρεθούν οι ΗΠΑ όπου τα δεδομένα είναι διαθέσιμα σχετικά φθηνά από την κυβέρνηση, σε άλλες περιοχές όπως η Ευρώπη είναι ακριβά, καθώς διατίθενται από ιδιωτικές εταιρίες. Επίσης σε άλλες αναπτυσσόμενες, κυρίως, χώρες δεν έχουν συλλεχτεί τέτοια δεδομένα.

- *Απαιτούν μεγάλη πολυπλοκότητα για να αναπτυχθούν.*

- *Ζητήματα πνευματικής ιδιοκτησίας:* Οι άνθρωποι είναι επιφυλακτικοί να δημοσιοποιήσουν γεωγραφικά δεδομένα που διαθέτουν, ειδικά σε περιοχές που αυτά σπανίζουν και είναι ακριβά, διότι φοβούνται πως θα χρησιμοποιηθούν χωρίς να δώσουν οι ίδιοι τις απαραίτητες άδειες χρήσης.

- *Ζητήματα προστασίας προσωπικών δεδομένων:* Με λεπτομερείς πληροφορίες διαθέσιμες και με συνδυασμό πολλών πηγών δεδομένων, είναι δυνατή η συλλογή πολλών προσωπικών πληροφοριών διαφόρων ατόμων.

## 1.3. Στόχος της διπλωματικής

Σε αυτή την εργασία θέλαμε να εκμεταλλευτούμε την αυξανόμενη διαθεσιμότητα της χωρικής πληροφορίας, ώστε να διευκολύνουμε τον τελικό χρήστη στην εύρεση σημείων ενδιαφέροντος. Ως σημείο ενδιαφέροντος στην παρούσα εργασία επιλέξαμε τα εστιατόρια. Η επιλογή αυτή έγινε λόγω του ότι υπάρχει αρκετή διαθέσιμη πληροφορία για αυτά, ενώ λόγω του ότι τα εστιατόρια είναι και αντικείμενο κριτικής, μας δόθηκε η δυνατότητα να δημιουργήσουμε και ένα social κομμάτι που δίνει την δυνατότητα σχολιασμού στους χρήστες. Τελικά θέλαμε να δημιουργήσουμε μια εφαρμογή που θα ήταν εύχρηστη και θα στηριζόταν στην κοινότητα που θα σχηματιζόταν γύρο από αυτή ώστε να παρέχει έγκυρες πληροφορίες.

## 1.4. Εύρεση και ακρίβεια των δεδομένων

Οι διευθύνσεις και τα στοιχεία των εστιατορίων ανακτήθηκαν από αντίστοιχες σελίδες μέσω αυτόματου κώδικα που δημιουργήσαμε. Σε πολλά βέβαια εστιατόρια δεν υπήρχαν διαθέσιμες οι γεωγραφικές πληροφορίες. Σε αυτό μας βοήθησε η υπηρεσία γεωεντοπισμού της Google. Η συγκεκριμένη υπηρεσία μπορεί από την διεύθυνση να σου επιστρέψει συντεταγμένες σημείου. Η ακρίβεια των δεδομένων που παρέχει η δωρεάν έκδοση της υπηρεσίας δεν είναι και η καλύτερη δυνατή, καθώς η εταιρία έχει επιλέξει να δίνει πιο ακριβείς πληροφορίες στους χρήστες που πληρώνουν για αυτή. Η ακρίβεια των δεδομένων μπορεί να διορθωθεί μέσα από τις παρατηρήσεις της κοινότητας των χρηστών. Οι διαχειριστές έχουν την δυνατότητα να διορθώνουν τα δεδομένα.

## 1.5. Σύντομη περιγραφή των λειτουργιών

Η εφαρμογή υποστηρίζει 4 τρόπους αναζήτησης εστιατορίων. Οι 2 πρώτοι είναι μέσω πεδίων αναζήτησης με βάση την κουζίνα και την περιοχή αντίστοιχα. Ο τρίτος τρόπος εκμεταλλεύεται τον χάρτη για να κάνει την αναζήτηση. Αυτή γίνεται με βάση συγκεκριμένη περιοχή που ορίζει ο χρήστης μέσω του χάρτη. Η τέταρτη επιλογή αναζήτησης είναι μέσω της λίστας με όλες τις περιοχές και όλα τα είδη κουζίνας που περιέχει η βάση δεδομένων.

Στα αποτελέσματα όλων των παραπάνω αναζητήσεων παρέχεται η δυνατότητα φιλτραρίσματος των αποτελεσμάτων ώστε να είναι ευκολότερη η εύρεση των εστιατορίων που πραγματικά μας ενδιαφέρουν. Τα διαθέσιμα φίλτρα είναι το είδος κουζίνας, η περιοχή, και η αξιολόγηση που έχει γίνει από τους υπόλοιπους χρήστες.

Στους χρήστες παρέχεται η δυνατότητα εγγραφής στο σύστημα ώστε να έχουν δικαίωμα να σχολιάσουν και να βαθμολογήσουν τα εστιατόρια. Επίσης παρέχεται η δυνατότητα ανταλλαγής προσωπικών μηνυμάτων μεταξύ των εγγεγραμμένων χρηστών.

Τέλος στους χρήστες με δικαιώματα διαχειριστή παρέχονται επιπλέον δυνατότητες ελέγχου. Τους δίνεται η δυνατότητα τροποποίησης υπάρχοντος εστιατορίου και η προσθήκη νέου.

Ακόμα έχουν την δυνατότητα να ενημερώνουν τις γεωγραφικές πληροφορίες όλων των εστιατορίων που δεν διαθέτουν γεωγραφικά δεδομένα, με αυτόματο τρόπο. Επίσης έχουν δυνατότητα διαγραφής των σχολίων οποιουδήποτε χρήστη, όπως και δυνατότητα διαγραφής λογαριασμών άλλων χρηστών ή προβιβασμό υπάρχοντος απλού λογαριασμού σε λογαριασμό με δικαιώματα διαχειριστή.

## 1.6. Διάρθρωση των επόμενων κεφαλαίων

Στο Κεφάλαιο 2 αναφερόμαστε στην υπηρεσία Google Maps. Κάνουμε μια σύντομη αναδρομή στην ανάπτυξή της ως σήμερα, αναφέροντας όλες τις προσθήκες που έγιναν στην υπηρεσία με χρονολογική σειρά. Επίσης αναφέρουμε τις βασικές συναρτήσεις του Google Maps API και παραδείγματα χρήσης αυτών.

Στο Κεφάλαιο 3 αρχικά περιγράφουμε την λειτουργία της τεχνολογίας Java Servlet και τι πλεονεκτήματα μας προσφέρει. Στη συνέχεια αναφέρουμε τα προβλήματα που αυτή η τεχνολογία δημιούργησε και δίνουμε ως εναλλακτική λύση την τεχνολογία Java Server Pages (JSP). Τέλος αναφέρουμε τα μειονεκτήματα της τεχνολογίας JSP σε σχέση με την τεχνολογία servlet και αναπτύσσουμε την αρχιτεκτονική Model View Controller (MVC) η οποία έχει την δυνατότητα να συνδυάσει τις δύο παραπάνω τεχνολογίες ώστε να εκμεταλλευτεί τα πλεονεκτήματα και των δύο.

Στο Κεφάλαιο 4 παρουσιάζουμε αναλυτικά όλες τις λειτουργίες της εφαρμογής που υλοποιήσαμε. Το κεφάλαιο αυτό χωρίζεται σε τρία κυρίως μέρη. Το πρώτο παρουσιάζει τις λειτουργίες που είναι διαθέσιμες χωρίς να χρειάζεται εγγραφή στο σύστημα, το δεύτερο τις λειτουργίες που είναι διαθέσιμες μόνο σε εγγεγραμμένους χρήστες και το τρίτο τις λειτουργίες που είναι διαθέσιμες μόνο σε εγγεγραμμένους χρήστες με δικαιώματα διαχειριστή.

Στο Κεφάλαιο 5 αναλύουμε πως υλοποιήθηκαν τα σημαντικότερα σημεία της εφαρμογής. Αρχικά παρουσιάζουμε το διάγραμμα οντοτήτων συσχετίσεων και το σχεσιακό σχήμα της βάσης δεδομένων της εφαρμογής. Στη συνέχεια παρουσιάζουμε πως έγινε η υλοποίηση της αρχιτεκτονικής MVC στην εφαρμογή μας, με ποιο τρόπο γίνεται η επικοινωνία της εφαρμογής με τη βάση δεδομένων και πως γίνεται η διαχείριση των αιτημάτων του χρήστη. Τέλος παρουσιάζουμε τον τρόπο με τον οποίο χρησιμοποιήσαμε την τεχνολογία JSP για την προβολή των πληροφοριών στον φυλλομετρητή του χρήστη.

Τέλος στο Κεφάλαιο 6 παρουσιάζουμε μερικές εφαρμογές παρόμοιες με την δική μας και τις κύριες δυνατότητές τους και στην συνέχεια επισημαίνουμε τα πλεονεκτήματα της εφαρμογής μας σε σχέση με αυτές τις εφαρμογές.

# 2. Υπηρεσία Χαρτογράφησης μέσω διαδικτύου της Google

Η πιο ευρέως διαδεδομένη υπηρεσία χαρτογράφησης μέσω διαδικτύου είναι η υπηρεσία Google Maps [3] (επίσημα γνωστή σαν Google Local). Η υπηρεσία διατίθεται δωρεάν για μη εμπορική χρήση. Η υπηρεσία παρέχεται μέσω του ιστότοπου http://maps.google.com ή μέσω του API που η εταιρία έχει δώσει διαθέσιμο στο κοινό και με χρήση του είναι δυνατή η ενσωμάτωση των υπηρεσιών Google Maps σε οποιαδήποτε ιστοσελίδα.

Σύμφωνα με έναν από τους δημιουργούς (Lars Rasmussen) του, το Google Maps «είναι ένας τρόπος να οργανώνεις τις πληροφορίες του κόσμου γεωγραφικά.» Η υπηρεσία παρέχει οδικούς χάρτες σχεδόν για όλο τον κόσμο, πλοήγηση για διαδρομή με τα πόδια, με αυτοκίνητο ή με δημόσιες συγκοινωνίες, καθώς και επαγγελματικούς οδηγούς για διάφορες χώρες. Επίσης παρέχεται η δυνατότητα να βλέπεις υψηλής ανάλυσης δορυφορικές εικόνες στις περισσότερες περιοχές ή να βλέπεις συνδυασμένα δορυφορικές εικόνες με πληροφορίες χάρτη επάνω.

Η υλοποίηση του API έγινε με χρήση javascript. Καθώς ο χρήστης μετακινεί τον χάρτη, κατεβαίνουν αυτόματα οι νέες εικόνες από τον server και ενσωματώνονται στον χάρτη. Όταν γίνεται μια αναζήτηση, τα αποτελέσματα εμφανίζονται στην πλευρική μπάρα και στον χάρτη δυναμικά χωρίς να χρειάζεται να γίνει επαναφόρτωση της σελίδας [2].

## 2.1. Χρονική αναδρομή στην ανάπτυξη του Google Maps

Η αρχή της υπηρεσίας ήταν μια εφαρμογή χαρτογράφησης που δημιουργήθηκε το 2003 από τους Lars και Jen Rasmussen για λογαριασμό της εταιρίας τους, Where 2 Technologies.

Τον Οκτώβριο του 2004 η Google Inc εξαγοράζει την εταιρία Where 2, και μετατρέπει την εφαρμογή σε υπηρεσία web mapping με την ονομασία Google Maps. Η υπηρεσία ανακοινώθηκε πρώτη φορά στο ιστολόγιο της Google στις 8 Φεβρουαρίου 2005. Αρχικά υποστήριζε τους φυλλομετρητές Internet Explorer και Mozilla.

Τον Απρίλιο του 2005 προστέθηκε η δυνατότητα προβολής δορυφορικών φωτογραφιών της προβαλλόμενης περιοχής. Επίσης παρουσιάστηκε η εφαρμογή Google Ride Finder. Η εφαρμογή εμφάνιζε, με χρήση χαρτών του Google Maps, σε πραγματικό χρόνο τις θέσεις επιλεγμένων ταξί και λιμουζίνων σε συγκεκριμένες πόλεις των ΗΠΑ.

Τον Ιούνιο του 2005 παρουσιάστηκε το πρώτο API και ξεκίνησε η λειτουργία της υπηρεσίας στην Ιαπωνία, περιλαμβάνοντας οδικούς χάρτες για όλη την χώρα. Στις 22 Ιουλίου 2005 ενσωματώθηκε η επιλογή Hybrid View. Με αυτή την επιλογή μπορούσες να βλέπεις της πληροφορίες του χάρτη πάνω σε δορυφορικές φωτογραφίες.

Ως τις 2 Ιανουαρίου του 2006 η υπηρεσία παρείχε οδικούς χάρτες για της ΗΠΑ, τον Καναδά, το Ενωμένο Βασίλειο, την Ιαπωνία και ορισμένες πόλεις της Δημοκρατίας της Ιρλανδίας. Επίσης

προστέθηκαν χάρτες για την περιοχή του Τορίνου, λόγο της έναρξης των Χειμερινών Ολυμπιακών Αγώνων του 2006. Επίσης στις 23 Ιανουαρίου 2006 το Google Maps άρχισε να χρησιμοποιεί την ίδια βάση δεδομένων δορυφορικών φωτογραφιών με το Google Earth.

Στις 3 Απριλίου 2006 κυκλοφόρησε η δεύτερη έκδοση του Google Maps API. Στις 11 Ιουνίου 2006 έγινε μια από τις πιο σημαντικές προσθήκες στο API, η υπηρεσία γεωεντοπισμού (geocoding). Δινόταν δηλαδή η δυνατότητα δίνοντας μια διεύθυνση να μας επιστρέφονταν οι γεωγραφικές συντεταγμένες της διεύθυνσης. Αυτή ήταν μια από της πιο περιζήτητες λειτουργίες τότε.

Στις 14 Ιουνίου 2006 παρουσιάστηκε το Google Maps Enterprise, μια εμπορική υπηρεσία βασισμένη στο Google Maps API, με λιγότερους περιορισμούς, πιο παραμετροποιήσιμο περιβάλλον και υποστήριξη intranet [4]. Επιπρόσθετα τον επόμενο μήνα, Ιούλιο, η Google άρχισε να περιλαμβάνει στα αποτελέσματα της μηχανής αναζήτησης που έχει, δεδομένα από το Google Maps, τα οποία παρουσιάζονταν μέσα σε μικρούς χάρτες με χρήση του API της υπηρεσίας.

Στις 19 Δεκεμβρίου 2006 προστέθηκε η δυνατότητα να μπορείς να ορίσεις πολλαπλούς προορισμούς στον πλοηγό που ενσωματώνει η υπηρεσία.

Τον Φεβρουάριο του 2007 ξεκίνησαν να εμφανίζονται στον χάρτη κτίρια και στάσεις του μετρό σε Νέα Υόρκη Ουάσινγκτον, Λονδίνο, Σαν Φρανσίσκο και μερικές ακόμα πόλεις. Επίσης στις 28 Φεβρουαρίου 2007 ξεκίνησε η λειτουργία της υπηρεσίας Google Traffic Info, η οποία παρείχε πληροφορίες κυκλοφοριακής ροής σε πραγματικό χρόνο για τις 30 μεγαλύτερες πόλεις των ΗΠΑ.

Στις 18 Μαΐου 2007 προστέθηκε η δυνατότητα αναζήτησης στην γειτονιά. Ουσιαστικά δύνεται η δυνατότητα να δίνεις την περιοχή στην οποία σε ενδιαφέρει, ώστε τα αποτελέσματα να περιορίζονται μόνο σε αυτή [5].

Στις 29 Μαΐου 2007 προστέθηκε στο API η δυνατότητα να δίνονται οδηγίες πλοήγησης. Η δυνατότητα αυτή υπήρχε στην υπηρεσία της Google, αλλά δεν ήταν διαθέσιμη σε τρίτους μέσω του API [6]. Την ίδια ημέρα προστέθηκε η λειτουργία Street View στις πόλεις Σαν Φρανσίσκο, Νέα Υόρκη, Λας Βέγκας, Μαϊάμι και Ντένβερ. Η λειτουργία αυτή σου δίνει την δυνατότητα να έχεις οπτική 360 μοιρών σε διάφορα σημεία των δρόμων της πόλης [7].

Στις 28 Ιουνίου 2007 δίνεται η δυνατότητα να μεταβάλλεις την διαδρομή που έχει υπολογιστεί αυτόματα για την πλοήγηση απλά κάνοντας κλικ επάνω της και σέρνοντας στο σημείο όπου θες να περάσει. Η διαδρομή μεταβάλλεται σε πραγματικό χρόνο χωρίς να χρειαστεί να γίνει επαναφόρτωση της σελίδας [8].

Στις 31 Ιουλίου 2007 ανακοινώθηκε η υποστήριξη του hCart microformat, ένα HTML πρότυπο για την αναπαράσταση διευθύνσεων και γεωγραφικών δεδομένων. Έτσι τα αποτελέσματα των αναζητήσεων θα παράγονταν πλέον σε αυτή την μορφή [9].

Στις 13 Σεπτεμβρίου 2007 ανακοινώθηκε η προσθήκη 54αρων νέων χωρών στον χάρτη [10]. Τον ίδιο μήνα ανακοινώθηκε η προσθήκη του γεωφυσικού χάρτη (terrain) [11].

Στις 18 Μαρτίου 2008 δόθηκε η δυνατότητα σε κάθε χρήστη να διορθώνει τα σημεία ενδιαφέροντος που υπάρχουν, αλλά και να δημιουργεί καινούργια [12]. Επίσης στις 2 Απριλίου 2008 προστίθενται ισοϋψείς γραμμές στον γεωφυσικό χάρτη (terrain) [13].

Στις 14 Μαΐου 2008 ανακοινώθηκε το API για εφαρμογές flash [14], ενώ στις 22 Ιουλίου 2008 ανακοινώνεται η δυνατότητα πλοήγησης για πεζούς[15]. Επίσης στις 6 Αυγούστου 2008 ανακοινώνεται η λειτουργία της υπηρεσίας Street View στην Αυστραλία και στην Ιαπωνία [16].

Η επόμενη σημαντική εξέλιξη έγινε στις 29 Αυγούστου 2008 όπου Google υπογράφει συμφωνία με την εταιρία GeoEye, σύμφωνα με την οποία η εταιρία θα προμηθεύει αποκλειστικά την Google με δορυφορικές εικόνες [17].

Ακόμα μια σημαντική λειτουργία ανακοινώθηκε στις 22 Οκτωβρίου 2008, όταν προστέθηκε η δυνατότητα ανάποδου γεωεντοπισμού (reverse geocoding). Δηλαδή δίνοντας της συντεταγμένες σου επιστρέφεται η διεύθυνση του σημείου αυτού [18]. Τέλος στις 18 Νοεμβρίου 2008 ανακοινώθηκε η υποστήριξη του Adobe AIR για το Google Maps Flash API [19].

## 2.2. Οι κυριότερες συναρτήσεις του Google Maps API που χρησιμοποιήθηκαν και η λειτουργία τους

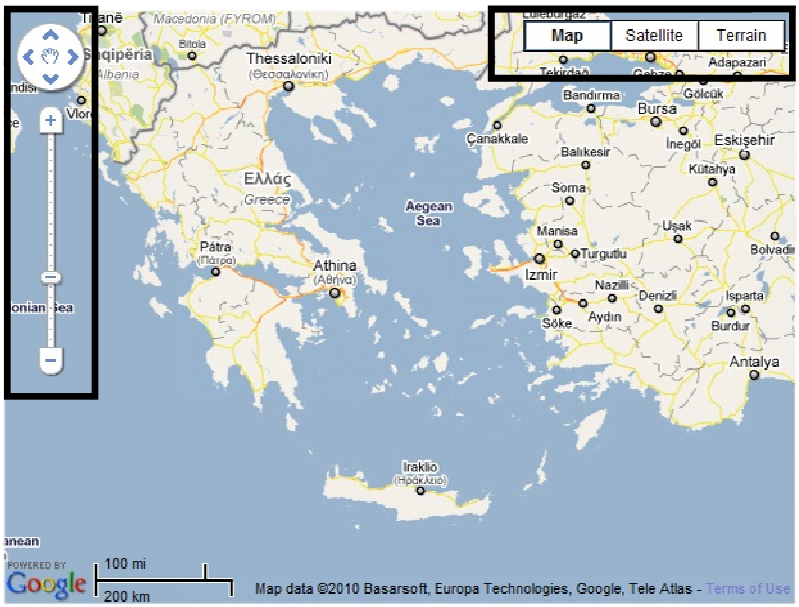
Το κυρίαρχο στοιχείο του API είναι ο ίδιος ο χάρτης. Ο χάρτης φιλοξενείται σαν αντικείμενο σε ένα DIV tag που έχουμε ορίσει στον κώδικα HTML, στο σημείο που θέλουμε να φαίνεται. Στην Εικόνα 3, στην γραμμή 22, βλέπουμε τον DIV που περιέχει το αντικείμενο που αναπαριστά τον χάρτη. Το αντικείμενο δημιουργείται με την συνάρτηση ***GMap2(container:Node, opts?:GMapOptions)***. Η παράμετρος *Node* είναι το στοιχείο DIV που θα περιέχει τον χάρτη. Το δεύτερο όρισμα, opts, είναι μία λίστα παραμέτρων που μπορούμε να αρχικοποιήσουμε κατά την δημιουργία του αντικειμένου του χάρτη. Η παράμετρος αυτή είναι προαιρετική [20]. Η κλήση της παραπάνω συνάρτησης φαίνεται στην Γραμμή 14 της Εικόνα 3 όπου δημιουργούμε το βασικό αντικείμενο του χάρτη στο DIV με όνομα map_canvas.

Μετά την δημιουργία του αντικειμένου του χάρτη πρέπει υποχρεωτικά να ορίσουμε το κέντρο του, δηλαδή το σημείο που θα δείχνει αρχικά. Αυτό γίνεται με χρήση της συνάρτησης ***setCenter(center:GLatLng, zoom?:Number, type?:GMapType)***. Η παράμετρος *center* είναι οι γεωγραφικές συντεταγμένες του κέντρου του χάρτη, και πρέπει να δοθεί σαν αντικείμενο της κλάσης *GLatLng*, ή οποία ουσιαστικά περιέχει 2 αριθμούς, το γεωγραφικό πλάτος και μήκος, και διαθέτει συναρτήσεις ανάθεσης και λήψης τιμών (setters και getters) [21]. Οι άλλες 2 παράμετροι της *setCenter* είναι προαιρετικές. Η πρώτη προαιρετική παράμετρος,(*zoom* ), είναι ένας αριθμός από το 1 έως το 18, και ορίζει το αρχικό zoom του χάρτη με κέντρο πάντα το σημείο center. Η παράμετρος *type* χρησιμοποιείται αν θέλουμε να ορίσουμε κάποιο είδος χάρτη πέρα από τους βασικούς (map, satellite, terrain) που μας δίνει έτοιμους το API [22]. Στην Γραμμή 15 της Εικόνα 3 φαίνεται η κλήση της παραπάνω συνάρτησης που ορίζει το κέντρο του χάρτη μας στο σημείο (37.4419, -122.1419) και το αρχικό επίπεδο ζουμ στο 13.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xmlns:v="urn:schemas-microsoft-com:vml">
  <head>
    <meta http-equiv="content-type" content="text/html; charset=utf-8"/>
    <title>Google Maps API Sample</title>
    <script
    src="http://maps.google.com/maps?file=api&amp;v=2&amp;sensor=false&amp;key=ABQIAA
    type="text/javascript"></script>
    <script type="text/javascript">

    function initialize() {
      if (GBrowserIsCompatible()) {
        var map = new GMap2(document.getElementById("map_canvas"));
        map.setCenter(new GLatLng(37.4419, -122.1419), 13);
      }
    }

    </script>
  </head>
  <body onload="initialize()" style="font-family: Arial;border: 0 none;">
    <div id="map_canvas" style="width: 500px; height: 300px"></div>
  </body>
</html>

```

Εικόνα 3. Κώδικας που μας παράγει ένα βασικό χάρτη

Το επόμενο βασικό στοιχείο που πρέπει να αρχικοποιηθεί είναι το UI του χάρτη. Το UI αποτελεί τα στοιχεία ελέγχου του χάρτη που έχει στην διάθεσή του ο χρήστης ώστε να αλληλεπιδράσει με τον χάρτη. Με την συνάρτηση ***setUIToDefault()***, ορίζουμε μια σειρά από προκαθορισμένα στοιχεία ελέγχου του χάρτη που δίνει το API. Με αυτή την επιλογή ο χάρτης μας έχει το ίδιο control panel με τον χάρτη που εμφανίζεται στην σελίδα χαρτών της Google. Βέβαια μπορούμε να ορίσουμε δικό μας UI, από τα controls που μας δίνει το API, με τη συνάρτηση ***addControl(control:GControl, position?:GControlPosition)***. Η παράμετρος *control* είναι αντικείμενο την κλάσης GControl [23], η οποία διαθέτει μεθόδους για την δημιουργία διαφόρων UI για τον έλεγχο του χάρτη από τον χρήστη. Η παράμετρος *position* έχει να κάνει με την θέση του στοιχείου ελέγχου που δημιουργούμε πάνω στον χάρτη και είναι προαιρετική. Στην Εικόνα 4 φαίνεται ο χάρτης με τα default στοιχεία ελέγχου, τα οποία είναι επισημασμένα με τα μαύρα ορθογώνια.




Εικόνα 4. Ο χάρτης του Google Maps API με τα default στοιχεία ελέγχου

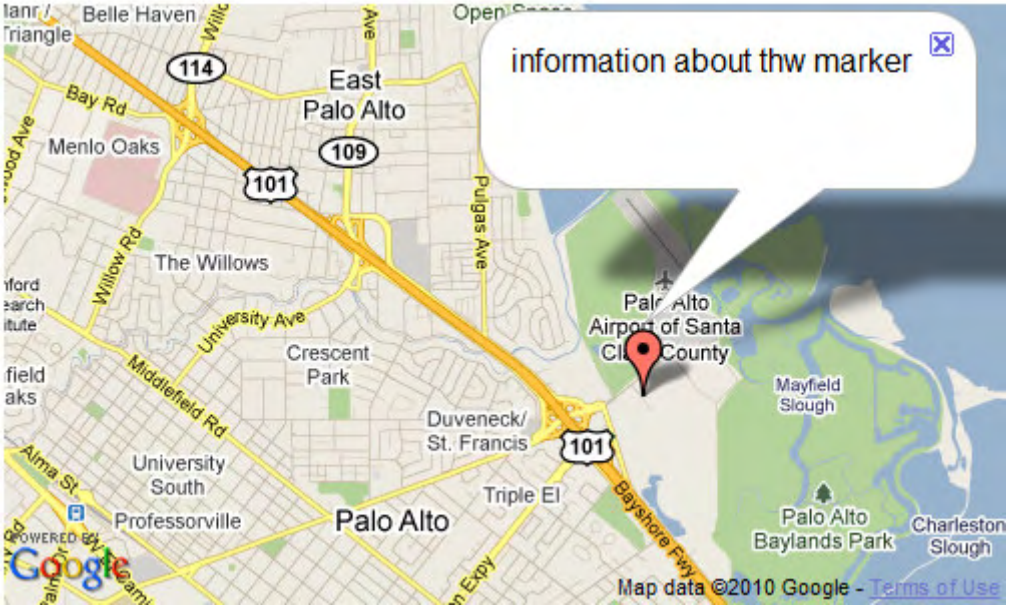
Έχοντας αρχικοποιήσει το βασικό αντικείμενο, τον χάρτη, μπορούμε να το χρησιμοποιήσουμε για να εμφανίσουμε πάνω σε αυτό πληροφορίες. Η κύρια πληροφορία που μπορεί να αναπαρασταθεί πάνω σε ένα χάρτη είναι ένα σημείο. Μπορούμε να επισημάνουμε ένα σημείο δημιουργώντας ένα marker πάνω στην χάρτη. Αυτό γίνεται με τη χρήση της κλάσης GMarker [24]. Με την συνάρτηση ***GMarker(latlng:GLatLng, opts?:GMarkerOptions)*** μπορούμε να δημιουργήσουμε ένα αντικείμενο GMarker. Η πρώτη παράμετρος, *latLng*, είναι οι γεωγραφικές συντεταγμένες του σημείου που θέλουμε να επισημάνουμε στο χάρτη. Η παράμετρος opts είναι προαιρετική και αντιστοιχεί σε ένα σύνολο παραμέτρων που καθορίζει τις ιδιότητες του αντικειμένου που δημιουργούμε [25]. Η κυριότερες επιλογές που μας δίνει αυτή η παράμετρος είναι ο ορισμός δικού μας εικονιδίου για τον marker, να επιλέξουμε αν θα μπορεί ο χρήστης να μετακινήσει το αντικείμενο πάνω στον χάρτη ή όχι, καθώς και διάφορες άλλες επιλογές που έχουν να κάνουν με κάποια οπτικά εφέ που έχει φτιάξει η Google. Ο marker προστίθεται στο χάρτη με κλήση της addOverlay(marker). Στην Εικόνα 5 βλέπουμε ένα κομμάτι κώδικα που δημιουργεί ένα τυχαίο marker πάνω στον χάρτη. Στις Γραμμές 16 ως 20 υπολογίζονται τα όρια του κομματιού του χάρτη που είναι ορατός, σε γεωγραφικές συντεταγμένες. Στη συνέχεια στις 21 ως 27 με βάση τα παραπάνω όρια, επιλέγονται τυχαίες συντεταγμένες και δημιουργείται ένα τυχαίο marker με χρήση των συναρτήσεων που αναφέρθηκαν παραπάνω.

Μια ενδιαφέρουσα λειτουργία που μας δίνει το API είναι η δυνατότητα ορισμού Events, τα οποία να δηλώνουν τις ενέργειες που θέλουμε να εκτελούνται σε διάφορες ενέργειες του χρήστη, όπως όταν κάνει κλικ πάνω στο χάρτη, ή μετακινεί τον cursor του ποντικιού του πάνω από ένα marker. Events μπορούν να οριστουν σε διάφορα αντικείμενα, ένα από τα οποία είναι και οι markers. Τα events ορίζονται από το namespace GEvent [26]. Για να ορίσουμε ένα event στον marker μας χρησιμοποιούμε τη συνάρτηση ***GEvent.addListener(source:Object, event:String, handler:Function)***. Η πρώτη παράμετρος, *source*, αναφέρεται στο αντικείμενο

στο οποίο θέλουμε να δημιουργήσουμε το event, στην περίπτωσή μας αυτό είναι ο marker. Το δεύτερο όρισμα, *event*, αναφέρεται στην ενέργεια του χρήστη πάνω στο αντικείμενο, η οποία θα πυροδοτεί την εκτέλεση της συνάρτησης που ορίζεται στο τρίτο όρισμα. Οι ενέργειες αυτές είναι διαφορετικές για κάθε αντικείμενο. Οι πιο συνηθισμένες είναι το click, dblclick, mouseover κτλ. Τώρα η συνάρτηση του τρίτου ορίσματος μπορεί να κάνει πολλά πράγματα, από το να ενημερώνει κάποιες μεταβλητές, μέχρι να προσθέτει νέα αντικείμενα στον χάρτη. Ένα σύνηθες event για τους markers είναι το άνοιγμα του παραθύρου πληροφοριών (info window) του marker όταν ο χρήστης κάνει κλικ πάνω του. Αυτό ορίζεται στην συνάρτηση που δίνεται ως τρίτο όρισμα. Στην Εικόνα 5 έχουμε ένα κομμάτι κώδικα το οποίο δημιουργεί ένα τυχαίο marker, και ένα event ώστε όταν κάνουμε κλικ πάνω στον marker να ανοίγει το info window. Το event δημιουργείται με χρήση των συναρτήσεων που περιγράψαμε παραπάνω, στις Γραμμές 24 ως 26. Στην Εικόνα 6 βλέπουμε το αποτέλεσμα του κώδικα της Εικόνα 5.

```
1  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
2  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
3  <html xmlns="http://www.w3.org/1999/xhtml"
4  xmlns:v="urn:schemas-microsoft-com:vml">
5    <head>
6      <meta http-equiv="content-type" content="text/html; charset=utf-8"/>
7      <title>Google Maps API Sample</title>
8      <script src="http://maps.google.com/maps?file=api&amp;v=2&amp;sensor=false
9      <script type="text/javascript">
10     function initialize() {
11       if (GBrowserIsCompatible()) {
12         var map = new GMap2(document.getElementById("map_canvas"));
13         map.setCenter(new GLatLng(37.4419, -122.1419), 13);
14
15         // Add 10 markers to the map at random locations
16         var bounds = map.getBounds();
17         var southWest = bounds.getSouthWest();
18         var northEast = bounds.getNorthEast();
19         var lngSpan = northEast.lng() - southWest.lng();
20         var latSpan = northEast.lat() - southWest.lat();
21         var point = new GLatLng(southWest.lat() + latSpan * Math.random(),
22                                 southWest.lng() + lngSpan * Math.random());
23         var marker= new GMarker(point);
24         GEvent.addListener(marker, "click", function() {
25         marker.openInfoWindowHtml("information about this marker");
26         });
27         map.addOverlay(marker);
28       }
29     }
30     </script>
31   </head>
32   <body onload="initialize()"style="font-family: Arial;border: 0 none;">
33     <div id="map_canvas" style="width: 500px; height: 300px"></div>
34   </body>
35 </html>
```

Εικόνα 5. Κώδικας που δημιουργεί ένα τυχαίο marker και ένα event ώστε όταν πατήσουμε τον marker να ανοίξει το info window




Εικόνα 6. Ο χάρτης που έχουμε ως αποτέλεσμα του κώδικα της προηγούμενης εικόνας

Εκτός από τον ίδιο τον χάρτη, η πιο χρήσιμη λειτουργία που μας παρέχει το API είναι η δυνατότητα γεωεντοπισμού ( geocoding ). Η λειτουργία αυτή είναι αυτόνομη, δεν απαιτεί την ύπαρξη αντικειμένου-χάρτη για να λειτουργήσει, και μας δίνει την δυνατότητα να δώσουμε μια διεύθυνση και να πάρουμε ως αποτέλεσμα τις γεωγραφικές συντεταγμένες του σημείου που υποδεικνύει αυτή ή το αντίστροφο ( γνωστό και ως reverse geocoding ). Η υπηρεσία διατίθεται από την Google με περιορισμούς στον ημερήσιο αριθμό αιτήσεων για geocoding ( 2500 αιτήσεις ημερησίως για κάθε IP ), καθώς και στην ρυθμό αποστολής των αιτήσεων. Η υπηρεσία μπορεί να αιτηθεί με 2 τρόπους:

1. Με αίτημα http στη διεύθυνση http://maps.google.com/maps/api/geocode/*output?parameters* όπου το *output* αντικαθίσταται με την επιθυμητή κωδικοποίηση του αντικειμένου εξόδου ( json ή xml ) και το *parameters* με μία λίστα υποχρεωτικών ή προαιρετικών παραμέτρων. Η πρώτη παράμετρος είναι η διεύθυνση ( address ) που θέλουμε να εντοπίσουμε ή γεωγραφικές συντεταγμένες ( latlng ) για την περίπτωση του reverse geocoding και είναι υποχρεωτική. Η επόμενη παράμετρος είναι η bounds η οποία αναπαριστά ένα ορθογώνιο περιέχοντας τις συντεταγμένες της νοτιοδυτικής και βορειοανατολικής κορυφής του διαχωρισμένες με (|) και είναι προαιρετική. Η περιοχή που δηλώνει το ορθογώνιο θα επηρεάζει αλλά δεν περιορίζει μόνο σε αυτή τα αποτελέσματα. Η τρίτη παράμετρος είναι η *region* η οποία είναι προαιρετική και περιέχει ένα κωδικό χώρας σύμφωνα με το ISO 3166-1. Η παράμετρος αυτή επηρεάζει τα αποτελέσματα αλλά δεν τα περιορίζει μόνο στην περιοχή που αυτή ορίζει. Η τελευταία παράμετρος είναι η *sensor,* η οποία είναι υποχρεωτική και παίρνει τιμές true ή false υποδεικνύοντας μας αν η αίτηση προέρχεται από συσκευή η οποία περιέχει αισθητήρα θέσης.

   Η παραπάνω αίτηση μας επιστρέφει είτε ένα αντικείμενο json είτε ένα αρχείο xml ανάλογα τι έχουμε ζητήσει στην αίτησή μας. Και στις 2 περιπτώσεις επιστρέφονται οι ίδιες πληροφορίες με διαφορετική διαμόρφωση [27].

Για παράδειγμα, μπορουμε να κάνουμε αίτηση στην διεύθυνση http://maps.google.com/maps/api/geocode/json?address=1600+Amphitheatre+Parkway,+Mountain+View,+CA&sensor=false. Τότε θα πάρουμε ως αποτέλεσμα τον παρακάτω κώδικα, που ορίζει ένα αντικείμενο json:

```
{
  "status": "OK",
  "results": [ {
    "types": [ "street_address" ],
    "formatted_address": "1600 Amphitheatre Pkwy, Mountain View, CA 94043,
USA",
    "address_components": [ {
      "long_name": "1600",
      "short_name": "1600",
      "types": [ "street_number" ]
    }, {
      "long_name": "Amphitheatre Pkwy",
      "short_name": "Amphitheatre Pkwy",
      "types": [ "route" ]
    }, {
      "long_name": "Mountain View",
      "short_name": "Mountain View",
      "types": [ "locality", "political" ]
    }, {
      "long_name": "San Jose",
      "short_name": "San Jose",
      "types": [ "administrative_area_level_3", "political" ]
    }, {
      "long_name": "Santa Clara",
      "short_name": "Santa Clara",
      "types": [ "administrative_area_level_2", "political" ]
    }, {
      "long_name": "California",
      "short_name": "CA",
      "types": [ "administrative_area_level_1", "political" ]
    }, {
      "long_name": "United States",
      "short_name": "US",
      "types": [ "country", "political" ]
    }, {
      "long_name": "94043",
      "short_name": "94043",
      "types": [ "postal_code" ]
    } ],
    "geometry": {
      "location": {
        "lat": 37.4220323,
        "lng": -122.0845109
      },
      "location_type": "ROOFTOP",
      "viewport": {
        "southwest": {
          "lat": 37.4188847,
          "lng": -122.0876585
        },
        "northeast": {
          "lat": 37.4251799,
          "lng": -122.0813633
        }
```

```
        }
      }
    } ]
}
```

2. Η άλλη επιλογή είναι η χρήση του αντικειμένου GClientGeocoder που υποστηρίζεται στο Google Maps API. Με χρήση της συνάρτησης ***getLocations(address:String, callback:function)*** εκτελούμε γεωεντοπισμό της διεύθυνσης του ορίσματος address. Το αντικείμενο που επιστρέφεται επεξεργάζεται από την συνάρτηση που δίνουμε ως δεύτερο όρισμα.

   Στην Εικόνα 7 βλέπουμε ένα κομμάτι κώδικα javascript που στις Γραμμές 5 και 6 δημιουργεί τον χάρτη και στη Γραμμή 7, ένα αντικείμενο GClientGeocoder. Με χρήση αυτού του αντικειμένου και της συνάρτησης που ορίσαμε παραπάνω εκτελεί την λειτουργία geocoding. Στη Γραμμή 33 καλεί την συνάρτηση getLocations με όρισμα την διεύθυνση που διάβασε από μια φόρμα της HTML. Στην συνέχεια τον έλεγχο αναλαμβάνει η συνάρτηση addAddressToMap, η οποία χειρίζεται το αντικείμενο που επιστρέφει η getLocations. Στη Γραμμή 15 ελέγχει αν το αντικείμενο περιέχει αποτελέσματα η όχι. Αν περιέχει, δημιουργεί ένα marker στο σημείο του αποτελέσματος, αλλιώς μας ειδοποιεί με ένα alert message.

```
var map;
var geocoder;

function initialize() {
 map = new GMap2(document.getElementById("map_canvas"));
 map.setCenter(new GLatLng(34, 0), 1);
 geocoder = new GClientGeocoder();
}

// addAddressToMap() is called when the geocoder returns an
// answer.  It adds a marker to the map with an open info window
// showing the nicely formatted version of the address and the country code.
function addAddressToMap(response) {
 map.clearOverlays();
 if (!response || response.Status.code != 200) {
   alert("Sorry, we were unable to geocode that address");
 } else {
   place = response.Placemark[0];
   point = new GLatLng(place.Point.coordinates[1],
                       place.Point.coordinates[0]);
   marker = new GMarker(point);
   map.addOverlay(marker);
   marker.openInfoWindowHtml(place.address + '<br>' +
     '<b>Country code:</b> ' + place.AddressDetails.Country.CountryNameCode);
 }
}

// showLocation() is called when you click on the Search button
// in the form.  It geocodes the address entered into the form
// and adds a marker to the map at that location.
function showLocation() {
 var address = document.forms[0].q.value;
 geocoder.getLocations(address, addAddressToMap);
}

// findLocation() is used to enter the sample addresses into the form.
function findLocation(address) {
 document.forms[0].q.value = address;
 showLocation();
}
```

Εικόνα 7. Κώδικας javascript για geocoding

Αντίστοιχα, ανάποδος γεωεντοπισμός μπορεί να γίνει με χρήση της συνάρτησης *getLocations(latlng:GLatLng, callback:function)*.

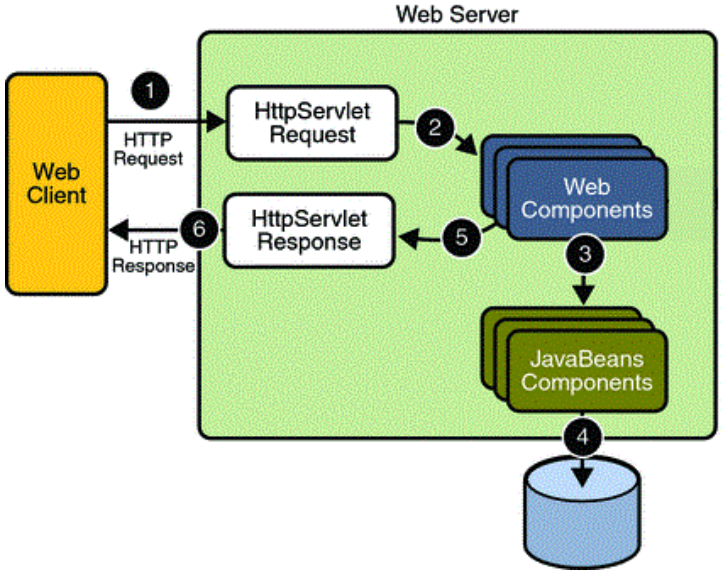
# 3. Java Servlet – Java Server Pages – Model View Controller Architecture

## 3.1. Java Servlet

Τα servlet είναι η πρόταση της java για την επέκταση και ενίσχυση των των υπηρεσιών του διαδικτύου. Προσφέρουν μια ανεξάρτητη από την πλατφόρμα που χρησιμοποιείται μέθοδο για την δημιουργία web-based εφαρμογών. με δυνατότητα χρήσης όλων των API της java, εξασφαλίζοντας όλα τα προνόμια που αυτή παρέχει, όπως η μεταφερσιμότητα, η επαναχρησιμοποίηση του κώδικα και η πολύ καλή απόδοση.

Τα servlet είναι μια κλάση-μέλος του Java Servlet API, ένα πρωτόκολλο μέσω του οποίου μια κλάση Java μπορεί να απαντάει σε αιτήσεις HTTP. Με την χρήση του API σε μια ιστοσελίδα, δίνεται η δυνατότητα να περιέχει η σελίδα δυναμικό περιεχόμενο με χρήση Java. Στην Εικόνα 8 βλέπουμε ένα διάγραμμα που δείχνει σχηματικά πως λειτουργεί ένα servlet.

Η κλάση servlet είναι μία από τις βασικές κλάσεις του API. Ένα αντικείμενο τύπου servlet λαμβάνει αιτήσεις και στέλνει απαντήσεις που εξαρτώνται από την αίτηση που έχει λάβει. Το βασικό αντικείμενο που ορίζεται στο πακέτο **javax.servlet,** δημιουργεί αντικείμενα java για να αναπαραστήσει τις αιτήσεις και τις απαντήσεις όπως επίσης και τις παραμέτρους του configuration του servlet. Το πακέτο αυτό είναι πιο γενικό και βασίζεται στο μοντέλο client-server. Πιο συγκεκριμένα για αιτήσεις http, υπάρχει το πακέτο **javax.servlet.http** το οποίο ορίζει αντικείμενα java για αναπαράσταση Http request και response όπως και αντικείμενα που αναπαριστούν το session (συνεδρία) και επιτρέπουν την διεκπεραίωση πολλαπλών requests μεταξύ client και server [28].

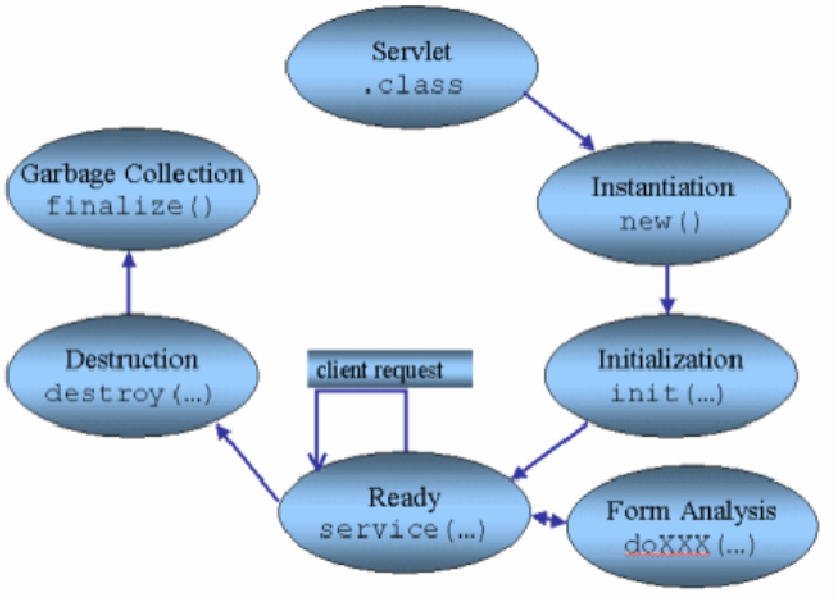



Εικόνα 8. Αλληλεπίδραση μεταξύ web client και web server. Ο client στέλνει ένα http request (1). Το αίτημα προωθείται σε ένα servlet (web component) από τον web container. Το servlet δημιουργεί τα κατάλληλα αντικείμενα (3), επικοινωνεί με την βάση αν χρειάζεται (4) και στέλνει τα αντικείμενα πίσω στον container (5) ο οποίος δημιουργεί το κατάλληλο http response (6).

Την διαχείριση των servlets σε ένα web server την αναλαμβάνει ο web container, το κομμάτι του server που αναλαμβάνει την υλοποίηση της διεπαφής για τα web components που ορίζει η J2EE αρχιτεκτονική. Αυτή η διεπαφή καθορίζει το περιβάλλον εκτέλεσης των web components (ένα από αυτά είναι και τα servlets) και περιλαμβάνει την ασφάλεια, τον συγχρονισμό, τον κύκλο ζωής κτλ [29].

Ο κύκλος ζωής ενός servlet περιλαμβάνει τα ακόλουθα βήματα, τα οποία απεικονίζονται γραφικά στην Εικόνα 9:

1. Η κλάση servlet φορτώνεται στον web container κατά την εκκίνηση του.
2. Ο container καλεί τον constructor για την δημιουργία ενός αντικειμένου servlet.
3. Ο container καλεί την συνάρτηση init() από το αντικείμενο που δημιούργησε, η οποία αρχικοποιεί το αντικείμενο και πρέπει να κληθεί πριν αυτό αρχίσει να εξυπηρετεί αιτήματα. Σε όλη την διάρκεια ζωής του αντικειμένου η init() καλείται μόνο μια φορά.
4. Μετά την αρχικοποίηση το αντικείμενο μπορεί να εξυπηρετεί αιτήματα. Κάθε αίτημα εξυπηρετείται σε δικό του ξεχωριστό νήμα. Για κάθε αίτημα που φτάνει στον web server ο web container καλεί την συνάρτηση service() του αντικειμένου. Η συνάρτηση αυτή, ανάλογα με το είδος του αιτήματος που δέχεται, το παραπέμπει σε μία κατάλληλη μέθοδο για να το χειριστεί. Οι μέθοδοι αυτοί υλοποιούνται από τον προγραμματιστή. Αν κάποια μέθοδος δεν έχει υλοποιηθεί επιστρέφεται μήνυμα λάθους ως απάντηση.
5. Τελικά, ο web container καλεί την συνάρτηση destroy() η οποία θέτει το αντικείμενο εκτός υπηρεσίας και το αποδεσμεύει. Η μέθοδος αυτή, όπως και η init(), καλείται μόνο μια φορά κατά τον κύκλο ζωής ενός αντικειμένου servlet [28].




Εικόνα 9. Ο κύκλος ζωής ενός servlet.

## 3.2. Java Server Pages (JSP)

### 3.2.1. Το πρόβλημα που δημιούργησαν τα servlets

Με την χρήση των servlets δόθηκε η δυνατότητα στους προγραμματιστές να συμπεριλάβουν δυναμικό περιεχόμενο στις σελίδες τους μέσα από την χρήση java και την δυνατότητα να «θυμάται» το σύστημα τις προηγούμενες καταστάσεις. Επίσης έγινε εύκολη η διαχείριση των HTTP requests, η ανάκτηση και επεξεργασία δεδομένων από πολλαπλές πηγές και η διαχείριση ενός ιστορικού πλοήγησης στην εφαρμογή.

Πέρα από όλα αυτά τα θετικά, τα servlets δυσκόλεψαν την δημιουργία και συντήρηση του HTML κώδικα καθώς αυτός παράγεται μέσα από κλήσεις της συνάρτησης println και όχι όπως παραδοσιακά παραγόταν με χρήση ενός επεξεργαστή κειμένου σε ένα ξεχωριστό αρχείο. Το HTML κομμάτι της εφαρμογής είναι πολύ σημαντικό, ειδικά στις μέρες μας που η διεπαφή του χρήστη (user interface), η οποία δημιουργείται από τον HTML κώδικα, παίζει ένα πολύ σημαντικό ρόλο στην επιτυχία μιας εφαρμογής. Λόγω της υπερπροσφοράς εφαρμογών, ειδικά στο web, ο χρήστης έχει την δυνατότητα να διαλέξει αυτή με την οποία κάνει πιο εύκολα την δουλειά του, καθιστώντας το interface ένα σημαντικό παράγοντα στην επιλογή εφαρμογής. Τα servlets έκαναν δύσκολο τον διαχωρισμό μεταξύ του στατικού κώδικα για το interface (HTML) που εκτελείται στον client και του δυναμικού κομματιού της εφαρμογής (java) που εκτελείται στον server [30].

### 3.2.2. Η λύση: Java Server Pages

Με την χρήση των Java Server Pages (jsp) ο στόχος είναι ο διαχωρισμός του δυναμικού κομματιού της εφαρμογής (java) από το στατικό interface (HTML), με ταυτόχρονη διατήρηση των πλεονεκτημάτων που προσφέρουν τα servlets. Στα αρχεία jsp μπορεί κανείς να γράφει τον HTML κώδικα με τον παραδοσιακό τρόπο, έχοντας ταυτόχρονα την δυνατότητα να ενσωματώνει κώδικα java μεταξύ του HTML, περικλείοντάς τον σε ειδικά tags (<% %>), προσθέτοντας έτσι δυναμικό περιεχόμενο. Έτσι τα αρχεία JSP αποτελούνται από 2 βασικά στοιχεία: τα scriptlets και τα markup. Τα markup είναι βασική HTML ή XML, ενώ τα scriptlets είναι οριοθετημένα (μέσα σε <% %> συνήθως) κομμάτια κώδικα java τα οποία μπορεί να αναμειγνύονται με τα markup. Ο κώδικας μέσα σε ένα tag δεν είναι ανάγκη να είναι ολοκληρωμένος, αρκεί στο τέλος του αρχείου ο συνολικός κώδικας όλων των tags να είναι συντακτικά σωστός. Μπορεί δηλαδή να παρεμβάλλονται κομμάτια markup μεταξύ μη ολοκληρωμένων scriptlets. Στην Εικόνα 10 βλέπουμε ένα απλό παράδειγμα ενός αρχείου jsp που τυπώνει την τρέχουσα ημερομηνία. Οι Γραμμές 3-9 αποτελούν ένα scriptlet. Στην γραμμή 10 χρησιμοποιείται το ειδικό tag <%=%> το οποίο περνάει την τιμή της μεταβλητής που περιέχει, στον κώδικα του markup.

```
1.  <HTML>
2.  <BODY>
3.  <%
4.   // This is a scriptlet.  Notice that the "date"
5.   // variable we declare here is available in the
6.   // embedded expression later on.
7.   System.out.println( "Evaluating date now" );
8.   java.util.Date date = new java.util.Date();
9.  %>
10. Hello!  The time is now <%= date %>
11. </BODY>
12. </HTML>
```

Εικόνα 10. Παράδειγμα ενός απλού JSP που τυπώνει την τρέχουσα ημερομηνία

Από αρχιτεκτονικής πλευράς τα jsp είναι μια υψηλού επιπέδου αφαίρεση των servlets. Στην πραγματικότητα, το αρχείο jsp μετατρέπεται σε ένα servlet, το οποίο ουσιαστικά εξυπηρετεί τα requests, και ο HTML κώδικας που είχε ενσωματωθεί μεταφέρεται στα println για την δημιουργία της εξόδου [31].

### 3.2.3. Μεταγλώττιση και λειτουργία ενός αρχείου JSP

Όπως αναφέρθηκε και παραπάνω, κάθε αρχείο jsp είναι ουσιαστικά ένα servlet. Κατά την πρώτη και μόνο φορά που το αρχείο ζητείται από ένα request, μεταφράζεται σε ένα servlet το οποίο λειτουργεί με τον τρόπο που έχουμε αναφέρει παραπάνω. Από εκεί και πέρα το αρχείο jsp αγνοείται. Για κάθε request που γίνεται προς αυτό εκτελείται το servlet που έχει παραχθεί, το οποίο εξυπηρετεί το request στέλνοντας ένα αρχείο html που παράγεται με βάση το request, τον κώδικα java και το κομμάτι HTML που έχει ενσωματωθεί σε αυτό.

### 3.2.4. Πλεονεκτήματα της τεχνολογίας JSP

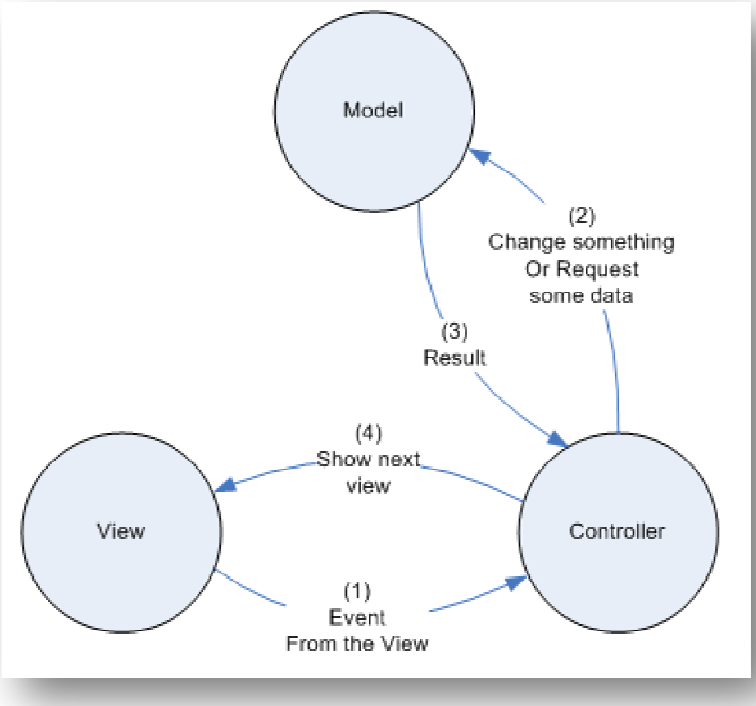
Αν και τεχνικά τα jsp δεν κάνουν τίποτα παραπάνω από ότι μπορούν να κάνουν τα servlets, κάνουν για τους προγραμματιστές πιο εύκολη την δημιουργία, ανάγνωση και συντήρηση του HTML κώδικα. Δίνουν την δυνατότητα να χωριστεί το προγραμματιστικό κομμάτι από αυτό του σχεδιασμού, αφού είναι δυνατό να δημιουργηθεί πρώτα το κομμάτι του interface ( HTML ) από τους σχεδιαστές και στην συνέχεια οι προγραμματιστές να ενσωματώσουν πάνω σε αυτό τον

κώδικα java που προσφέρει το δυναμικό περιεχόμενο. Αυτός ο διαχωρισμός επιτρέπει στους σχεδιαστές να χρησιμοποιήσουν οποιοδήποτε εργαλείο σχεδίασης HTML, όπως Macromedia DreamWeaver ή Adobe GoLive, και στους προγραμματιστές να δουλέψουν χωριστά το προγραμματιστικό κομμάτι σε οποιοδήποτε SDK προτιμούν. Τελικά ενώνεται το προγραμματιστικό κομμάτι με το interface ενσωματώνοντας java tags στον HTML κώδικα. Τα tags αυτά μπορούν να επικοινωνούν με μεταβλητές, συναρτήσεις, και αντικείμενα του server που βρίσκονται σε χωριστά αρχεία java. Ανάλογα με αυτά και με το αίτημα toy client παράγεται τελικά διαφορετικός HTML κώδικας ώστε να παρουσιάζει τις πληροφορίες που ζητήθηκαν [30].

## 3.3. Ενοποίηση των τεχνολογιών Java Servlet και Java Server Pages: Η αρχιτεκτονική MVC (Model View Controller)

### 3.3.1. Η αρχιτεκτονική MVC

Το MVC είναι μια αρχιτεκτονική λογισμικού, που σήμερα θεωρείται πρότυπη, και χρησιμοποιείται στην τεχνολογία λογισμικού. Το συγκεκριμένο πρότυπο απομονώνει τις διεργασίες επεξεργασίας και ανάκτησης των πληροφοριών από την διεπαφή χρήστη, η οποία αναλαμβάνει την προβολή των πληροφοριών και την ανάγνωση της εισόδου, επιτρέποντας ανεξάρτητη ανάπτυξη, έλεγχο και συντήρηση στο καθένα.




Εικόνα 11. Το πρότυπο MVC. Ο controller λαμβάνει ένα event από το View(1). Στη συνέχεια ζητάει από το μοντέλο να κάνει κάποια αλλαγή ή να επιστρέψει κάποια δεδομένα(2,3). Τέλος επιλέγει την κατάλληλη προβολή (4).

Το μοντέλο (**model**) χρησιμοποιείται για την διαχείριση των πληροφοριών. Είναι ουσιαστικά η αναπαράσταση τον δεδομένων με τα οποία αλληλεπιδρά η εφαρμογή. Τα δεδομένα μπορούν να προέρχονται από διάφορες πηγές πληροφορίας, όπως το διαδίκτυο, μια άλλη εφαρμογή ή μια βάση δεδομένων.

Η προβολή (**view**) οπτικοποιεί τα δεδομένα που περιέχει το μοντέλο με τρόπο ώστε να είναι κατανοητά από τον χρήστη και να μπορεί αυτός να αλληλεπιδράσει μαζί τους. Σε μια εφαρμογή μπορεί να υπάρχουν πολλαπλές προβολές ώστε να μπορούν να προβληθούν τα δεδομένα με διαφορετικούς τρόπους εστιάζοντας κάθε φορά σε διαφορετική οπτική.

Ο **controller** λαμβάνει την εξωτερική είσοδο και δίνει εντολή στο μοντέλο να κάνει συγκεκριμένες ενέργειες (actions). Στη συνέχεια χρησιμοποιεί την κατάλληλη προβολή για να προβάλει την έξοδο του μοντέλου και περιμένει νέα είσοδο [32] [33].

### 3.3.2. Πλεονεκτήματα της αρχιτεκτονικής MVC

- Προτρέπει τον προγραμματιστή να διαχωρίσει την κύρια λειτουργικότητα από το user interface. Με αυτό τον τρόπο μπορεί να δουλεύεται το interface σαν διαφορετικό project από διαφορετική ομάδα.
- Επιτρέπει πολλαπλά user interfaces για τα ίδια δεδομένα, πράγμα πολύ σημαντικό σήμερα, αφού έχουν μπει στην ζωή μας διαφόρων ειδών τερματικά όπως κινητά, tablets κτλ, το καθένα από τα οποία απαιτεί διαφορετική προσέγγιση για το interface.
- Γίνονται ευκολότερες οι αλλαγές στο επίπεδο τον δεδομένων, αφού το μοντέλο είναι αυτόνομο και ανεξάρτητο από το user interface. Αν για παράδειγμα αλλάξει σε μια εφαρμογή η βάση δεδομένων, χρειάζεται απλά να αλλάξεις το μοντέλο. Το επίπεδο του controller και του view αν έχει εφαρμοστεί σωστά το MVC δεν νοιάζονται για το ποια βάση υπάρχει από πίσω.
- Ο ίδιος ο controller είναι ένα πλεονέκτημα. Ουσιαστικά ενώνει διάφορες προσφερόμενες από το μοντέλο λειτουργίες και επιλέγει την καταλληλότερη προβολή από τις προσφερόμενες προβολές για να ικανοποιήσει ένα αίτημα με τον πιο αποδοτικό τρόπο [34].

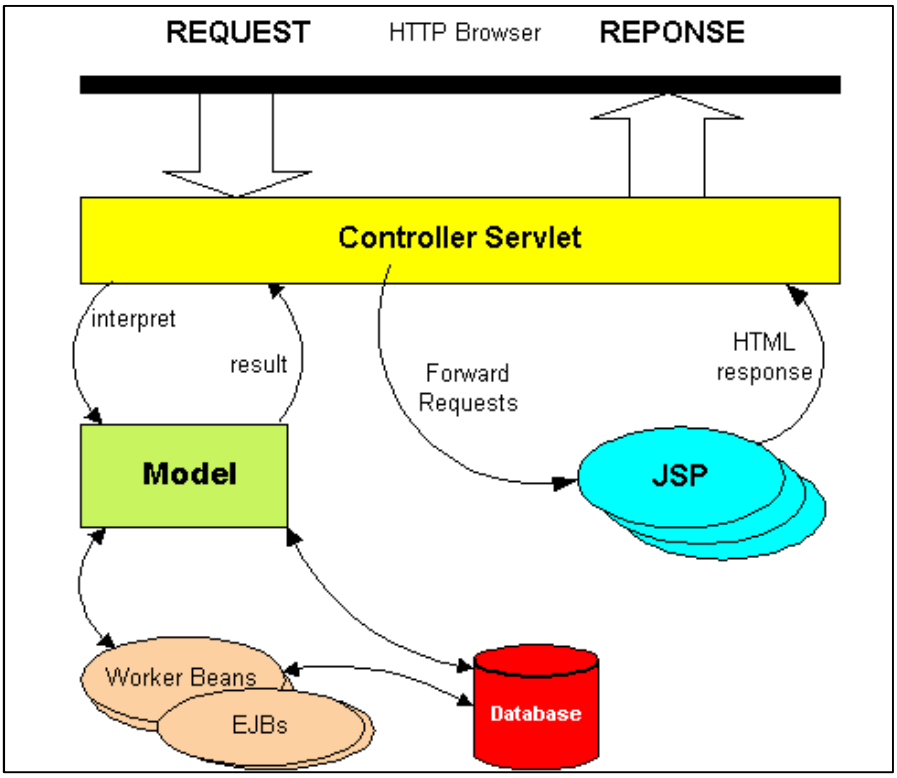
### 3.3.3. Εφαρμογή της αρχιτεκτονικής MVC με χρήση servlet και JSP

Παραπάνω είδαμε πως η χρήση των servlet μας επιτρέπει να έχουμε εύκολη διαχείριση των αιτημάτων και της ροής του προγράμματος αλλά δημιουργεί δυσκολία στην παραγωγή των απαντήσεων (κώδικας HTML). Την λύση μας την έδωσε η τεχνολογία JSP η οποία μας επέτρεψε να δημιουργούμε εύκολα απαντήσεις (HTML κώδικα) έχοντας ταυτόχρονα τα πλεονεκτήματα των servlets. Η ενσωμάτωση όμως κώδικα java στην HTML που επέβαλε το

JSP δημιουργεί προβλήματα συντήρησης όταν οι εφαρμογές γίνονται μεγάλες και ο κώδικας java αυξάνει. Μπορεί λοιπόν σε μικρές εφαρμογές η λύση που έδωσε το JSP να είναι βολική, αλλά όσο μεγαλώνει η εφαρμογή, τόσο μεγαλώνουν τα προβλήματα. Η Εικόνα 12 περιγράφει πως η αρχιτεκτονική MVC μπορεί να μας βοηθήσει να ενοποιήσουμε τις δύο παραπάνω τεχνολογίες, ώστε να ελαχιστοποιήσουμε τα μειονεκτήματα κάθε μιας.

Ο **controller** μπορεί να υλοποιηθεί σαν ένα servlet. Στην συνάρτηση init() αρχικοποιούμε ένα hashMap στο οποίο τα κλειδιά θα είναι τα ονόματα όλων των γεγονότων-λειτουργιών που κάνει η σελίδα μας και κάθε ένα θα αντιστοιχίζεται σε μία κλάση που θα χειρίζεται το γεγονός. Κάθε μία από αυτές τις κλάσεις πρέπει να υλοποιεί τις συναρτήσεις process() και forward() του servlet.

Οι παραπάνω κλάσεις αποτελούν τις ενέργειες (actions) του συστήματος και αποτελούν ουσιαστικά το μοντέλο (**model**) της αρχιτεκτονικής MVC. Κάθε κλάση περιέχει 2 μεταβλητές. Η πρώτη είναι ένα String το οποίο μέσω της forward() παραπέμπει τον έλεγχο σε ένα jsp ώστε να εμφανίσει την πληροφορία στον χρήστη. Η δεύτερη είναι ένα java bean που χρησιμοποιείται ώστε να μοντελοποιείται η πληροφορία που ανακτάται ή φορτώνεται στην βάση δεδομένων μέσα από την συνάρτηση process(). Η συνάρτηση αυτή επεξεργάζεται τα δεδομένα που παίρνει από την παράμετρο **HttpServletRequest request** και μεταβάλει το μοντέλο (ουσιαστικά την βάση). Στη συνέχεια περνάει τα αποτελέσματα της επεξεργασίας σε ένα jsp ( προβολή ) είτε μέσω του session, είτε μέσω της παραμέτρου **HttpServletRequest response** [35].




Εικόνα 12. Η εφαρμογή της αρχιτεκτονικής MVC με χρήση servlet και JSP για την υλοποίηση web εφαρμογών. Ο controller έχει υλοποιηθεί σαν servlet. Το μοντέλο αποτελείται από υλοποιήσεις των συναρτήσεων του servlet που κάνουν τις βασικές λειτουργίες επεξεργασίας. Το κομμάτι της προβολής υλοποιείται με χρήση JSP

Για λόγους συντήρησης και ευκολότερης μετάβασης από ένα σύστημα βάσης δεδομένων σε άλλο συνίσταται η χρήση ενός χαμηλότερου επιπέδου κλάσεων που αναλαμβάνουν να κάνουν τις ανακτήσεις και τις εγγραφές στην βάση δεδομένων. Οι κλάσεις αυτές είναι γνωστές στην βιβλιογραφία ως **Data Access Objects( DAO)**. Τα αντικείμενα αυτά παρέχουν ένα abstract interface για κάθε τύπο βάσης δεδομένων που υποστηρίζεται. Με αυτό τον τρόπο μπορούμε να αλλάξουμε σύστημα βάσης δεδομένων απλά γράφοντας μια κλάση που υλοποιεί το interface που ορίζει το DAO που χρησιμοποιεί η εφαρμογή μας, χωρίς να χρειάζεται οποιαδήποτε άλλη αλλαγή στην εφαρμογή μας [35].

Τέλος το κομμάτι του view υλοποιείται με χρήση αρχείων jsp. Όπως είπαμε παραπάνω η συνάρτηση forward() του κάθε action παραδίδει τον έλεγχο μετά την εκτέλεση της συνάρτησης process() σε ένα κατάλληλο αρχείο jsp. Το αρχείο αυτό αναλαμβάνει να παράγει τον κατάλληλο κώδικα HTML ώστε να εμφανίσει τυχόν αποτελέσματα και να δώσει στον χρήστη τις πιθανές επόμενες επιλογές ώστε να συνεχίσει την πλοήγηση στην εφαρμογή. Όλες οι παράμετροι που χρειάζονται βρίσκονται στο session ή στο αντικείμενο **HttpServletRequest response,** ανάλογα που έχουμε ορίσει το μοντέλο να τις τοποθετεί.

Τελικά, με χρήση της αρχιτεκτονικής MVC καταφέραμε να συνδυάσουμε τις τεχνολογίες servlet και JSP ώστε να υπερκεράσουμε τις αδυναμίες τις κάθε μίας. Χρησιμοποιήσαμε κάθε μία στο κομμάτι του MVC που είναι πιο ισχυρή, ώστε στο τέλος να έχουμε ένα μοντέλο το οποίο:

- Επιτρέπει την κατανομή ανεξάρτητων εργασιών σε κάθε μέλος της προγραμματιστικής ομάδας.
- Κάνει ευκολότερη την συντήρηση και την αποσφαλμάτωση της εφαρμογής.
- Δίνει την δυνατότητα επαναχρησιμοποίησης κομματιών κώδικα σε επόμενες εφαρμογές.
- Διευκολύνει την εισαγωγή στην εφαρμογή μας νέων τεχνολογιών που θα έρθουν στο μέλλον χωρίς να χρειάζονται ριζικές αλλαγές σε αυτή.

# 4. Η εφαρμογή που δημιουργήσαμε και η κύρια λειτουργικότητά της

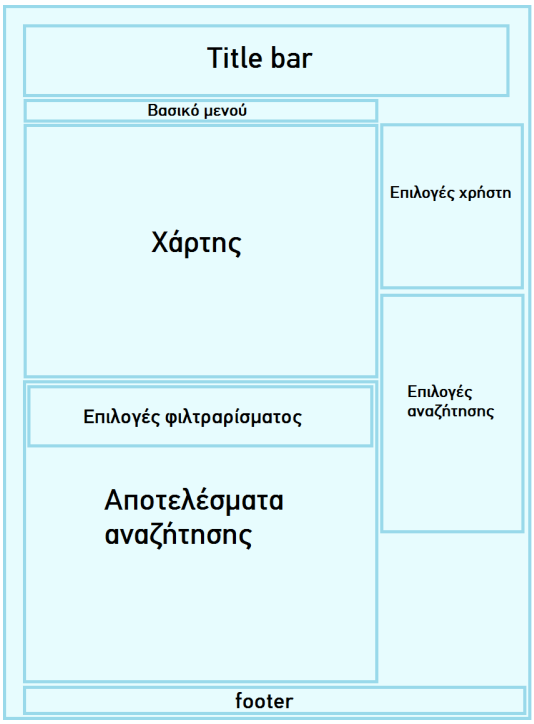
## 4.1. Γενική Ιδέα – Σκοπός

Ο κύριος σκοπός ήταν η δημιουργία μιας εφαρμογής που θα επέτρεπε την εύκολη εύρεση εστιατορίων εκμεταλλευόμενη την χωρική πληροφορία που μπορούσε να παραχθεί μέσω της υπηρεσίας Google Maps. Θέλαμε να εκμεταλλευτούμε την υπηρεσία της Google ώστε να διευκολύνουμε την αναζήτηση με γεωγραφικά κριτήρια. Επιπροσθέτως επιδιώξαμε να δώσουμε την δυνατότητα στον επισκέπτη να εκφράσει τη γνώμη του για τα εστιατόρια τα οποία παρουσιάζαμε, ώστε να δώσουμε περισσότερες και πιο ζωντανές πληροφορίες στους επόμενους επισκέπτες. Η συμμετοχή των επισκεπτών δίνει την δυνατότητα να ανανεώνεται η πληροφορία της εφαρμογής χωρίς κόστος, αφού αναλαμβάνει η κοινότητα να προσθέσει πληροφορίες. Επιπλέον, οι πληροφορίες γίνονται πιο άμεσες και έγκυρες στα μάτια των χρηστών.

Τα γεωγραφικά δεδομένα, σύμφωνα και με το κεφάλαιο 2 είναι αρκετά ακριβά και όχι εύκολα διαθέσιμα. Αυτή ήταν μια βασική δυσκολία που αντιμετωπίσαμε κατά την συλλογή των δεδομένων. Χρησιμοποιήσαμε την υπηρεσία geocoding του Google Maps για όσα εστιατόρια δεν είχαμε γεωγραφικά δεδομένα. Ωστόσο η υπηρεσία αυτή δεν είναι αρκετά αξιόπιστη, απαιτεί συγκεκριμένο format διευθύνσεων και δίνει πολλές λάθος αντιστοιχίσεις. Συμβιβαστήκαμε με αυτά τα λάθη, καθώς ο σκοπός της παρούσας εργασίας δεν ήταν η ακρίβεια των δεδομένων που παρέχεται μέσα από την εφαρμογή, αλλά η λειτουργικότητα των αναζητήσεων.

## 4.2. Βασική διάταξη της διεπαφής χρήστη

Για τη βασική διεπαφή χρήστη επιλέχθηκε η διάταξη που φαίνεται στην Εικόνα 13. Η διάταξη αυτή ακολουθείται στην αρχική σελίδα και σε όλα τα αποτελέσματα αναζήτησης.




Εικόνα 13. Διάταξη διεπαφής χρήστη για την αρχική σελίδα και τα αποτελέσματα αναζήτησης

- Στο **title bar** φαίνεται ο τίτλος της εφαρμογής μας ο οποίος αποτελεί σύνδεσμο και για την αρχική σελίδα
- Στο **βασικό μενού** περιέχονται 4 σύνδεσμοι. Ο πρώτος σε επαναφέρει στην αρχική σελίδα, ο δεύτερος σε πηγαίνει στην σελίδα με τις κατηγορίες των εστιατορίων, ο τρίτος σε πηγαίνει στην σελίδα με πληροφορίες σχετικές με την εφαρμογή, ενώ ο τέταρτος σύνδεσμος εμφανίζει τα στοιχεία επικοινωνίας με τους διαχειριστές της εφαρμογής.
- Οι επιλογές χρήστη εμφανίζουν τις διαθέσιμες στον χρήστη ενέργειες, όπως η σύνδεση/δημιουργία λογαριασμού, η εμφάνιση του προφίλ ή των προσωπικών μηνυμάτων αν ο χρήστης είναι συνδεδεμένος, καθώς και επιπρόσθετες επιλογές διαχείρισης αν ο χρήστης έχει δικαιώματα διαχειριστή. Περισσότερα στις ενότητες «4.4.1. Σύνδεση χρήστη, ανάκτηση χαμένου κωδικού και διαχείριση προφίλ», «4.4.3. Προσωπικά μηνύματα» και «4.5. Λειτουργίες που προσφέρονται μόνο σε χρήστες με δικαιώματα διαχειριστή».
- Οι **επιλογές αναζήτησης** εμφανίζουν τα δύο διαθέσιμα πεδία αναζήτησης, με βάση την περιοχή και το είδος κουζίνας, καθώς και την επιλογή αναζήτησης με βάση την περιοχή που έχει επιλεχθεί στον χάρτη. Περισσότερα στην ενότητα «4.3.1. Λειτουργίες αναζήτησης».

- Ο **χάρτης** εμφανίζει ένα χάρτη Google στον οποίο εμφανίζονται αποτελέσματα των αναζητήσεων όταν υπάρχουν διαθέσιμες γεωγραφικές πληροφορίες. Επίσης χρησιμοποιείται για την επιλογή περιοχής στην χωρική αναζήτηση.
- Τα **αποτελέσματα της αναζήτησης** εμφανίζουν τα εστιατόρια που βρέθηκαν σε μια αναζήτηση, παρέχοντας και την δυνατότητα ταξινόμησης των αποτελεσμάτων. Αν δεν έχει γίνει αναζήτηση το κομμάτι αυτό δεν εμφανίζεται καθόλου. Περισσότερα στην ενότητα «3.3.2. Εμφάνιση και φιλτράρισμα αποτελεσμάτων».
- Οι **επιλογές φιλτραρίσματος** εμφανίζονται όποτε πραγματοποιείται κάποια αναζήτηση και παρέχουν δυνατότητες προσθήκης επιπλέων περιορισμών στα αποτελέσματα της αναζήτησης. Περισσότερα στην ενότητα «3.3.2. Εμφάνιση και φιλτράρισμα αποτελεσμάτων».
-
- Το **footer** περιέχει πληροφορίες πνευματικής ιδιοκτησίας του template και της σελίδας.

Στην Εικόνα 14 βλέπουμε ένα παράδειγμα από την εφαρμογή που ακολουθεί την διάταξη της Εικόνα 13.


Εικόνα 14. Παράδειγμα της εφαρμογής που ακολουθεί την διάταξη της Εικόνα 13

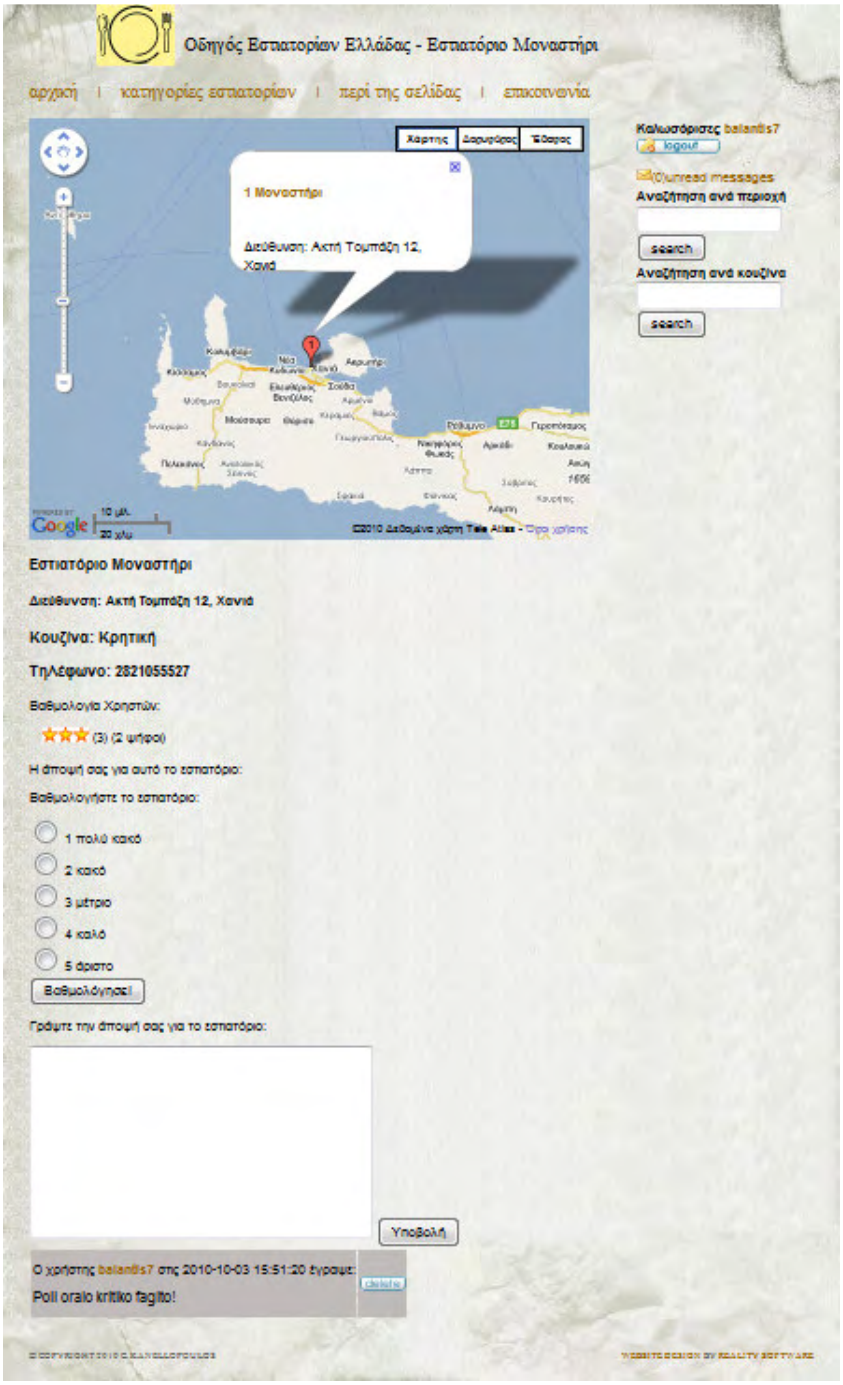
Η παραπάνω διάταξη διαφοροποιείται λίγο στην σελίδα εμφάνισης εστιατορίου. Στην Εικόνα 15 φαίνεται η διαφοροποιημένη διάταξη.


Εικόνα 15. Διάταξη διεπαφής χρήστη για την σελιδα λεπτομερειών εστιατορίου

Οι περιοχές title bar, Βασικό μενού, επιλογές χρήστη, επιλογές αναζήτησης, χάρτης και footer είναι ίδιες με παραπάνω.

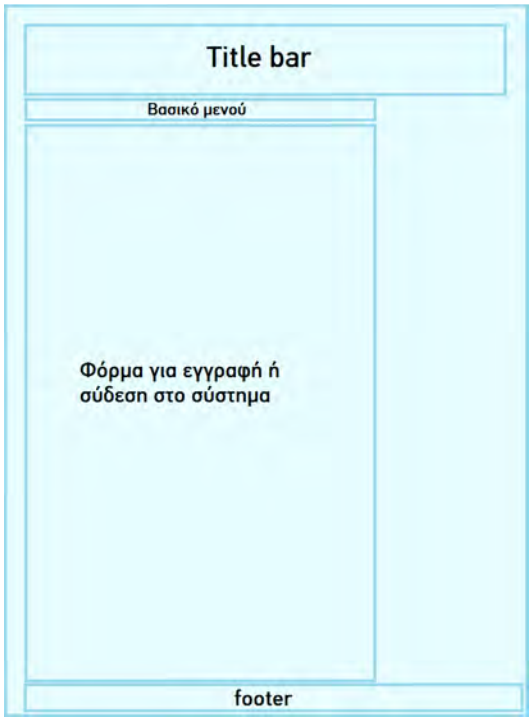
- **Οι πληροφορίες εστιατορίου** εμφανίζουν πληροφορίες για το εστιατόριο που έχει επιλεγεί, όπως διεύθυνση, τηλέφωνο, είδος κουζίνας καθώς και την βαθμολογία που έχει λάβει από τους χρήστες. Περισσότερα στην ενότητα «4.3.3. Σελίδα εστιατορίου και προφίλ χρηστών».
- Οι **επιλογές βαθμολόγησης και σχολιασμού** αν ο χρήστης είναι συνδεδεμένος εμφανίζουν επιλογή βαθμολόγησης του εστιατορίου και πεδίο εισαγωγής κειμένου για τον σχολιασμό του εστιατορίου. Περισσότερα στην ενότητα «3.4.2. Βαθμολόγηση και σχολιασμός εστιατορίου».
- Τέλος στα **σχόλια χρηστών** εμφανίζονται όλα τα σχόλια που έχουν γίνει για το συγκεκριμένο εστιατόριο.

Στην Εικόνα 16 βλέπουμε ένα παράδειγμα από την εφαρμογή που ακολουθεί την διάταξη της Εικόνα 15.




Εικόνα 16. Παράδειγμα της εφαρμογής που ακολουθεί την διάταξη της Εικόνα 15

Στην Εικόνα 17, φαίνεται η διάταξη της διεπαφής χρήστη για την σελίδα της δημιουργίας λογαριασμού ή σύνδεσης σε υπάρχων λογαριασμό.

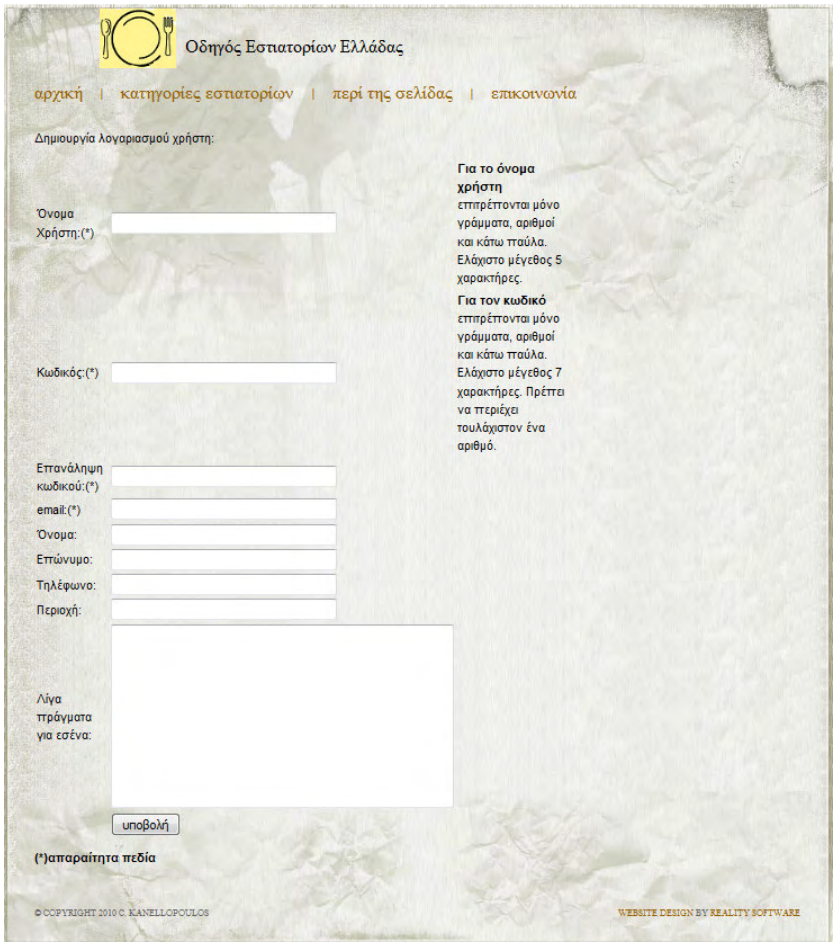



Εικόνα 17. Διάταξη διεπαφής χρήστη για την σελίδα εγγραφής ή εισαγωγής στο σύστημα

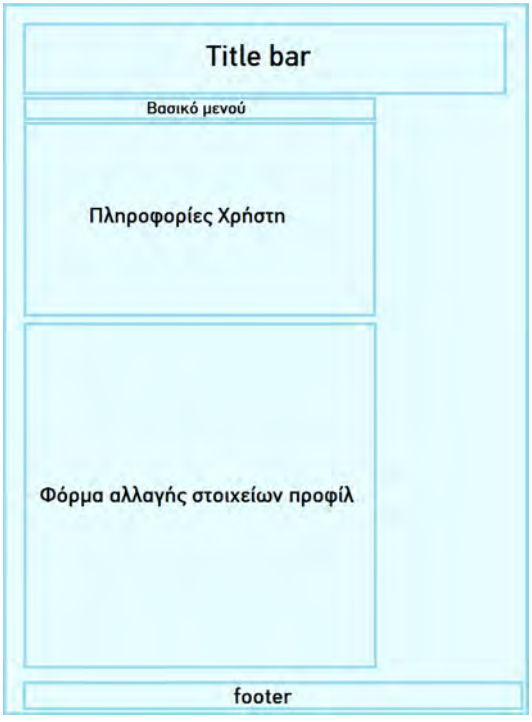
Στο πεδίο Φόρμα για εγγραφή ή σύνδεση στο σύστημα εμφανίζεται ανάλογα η φόρμα εγγραφής ή σύνδεσης. Περισσότερα στις ενότητες «4.3.4. Δημιουργία λογαριασμού χρήστη» και «4.4.1. Σύνδεση χρήστη, ανάκτηση χαμένου κωδικού και διαχείριση προφίλ».

Στην Εικόνα 18 βλέπουμε ένα παράδειγμα της εφαρμογής που ακολουθεί την διάταξη της Εικόνα 17. Στη συνέχεια, στην Εικόνα 19, φαίνεται η διάταξη της διεπαφής χρήστη για την σελίδα του προφίλ.

- Στην περιοχή πληροφορίες χρήστη εμφανίζονται τα στοιχεία του χρήστη που έχει επιλεγεί καθώς και επιλογή αποστολής προσωπικού μηνύματος και εμφάνισης όλων των σχολίων που έχει κάνει. Περισσότερα στην ενότητα «4.3.3. Σελίδα εστιατορίου και προφίλ χρηστών».
- Η περιοχή φόρμα **αλλαγής στοιχείων προφίλ** εμφανίζεται μόνο στην περίπτωση που ο επιλεγμένος χρήστης είναι ο ίδιος με τον χρήστη που είναι εκείνη την στιγμή συνδεδεμένος. Εμφανίζει μια φόρμα για την αλλαγή των στοιχείων που περιέχει το σύστημα. Περισσότερα στην ενότητα «4.4.1. Σύνδεση χρήστη, ανάκτηση χαμένου κωδικού και διαχείριση προφίλ».

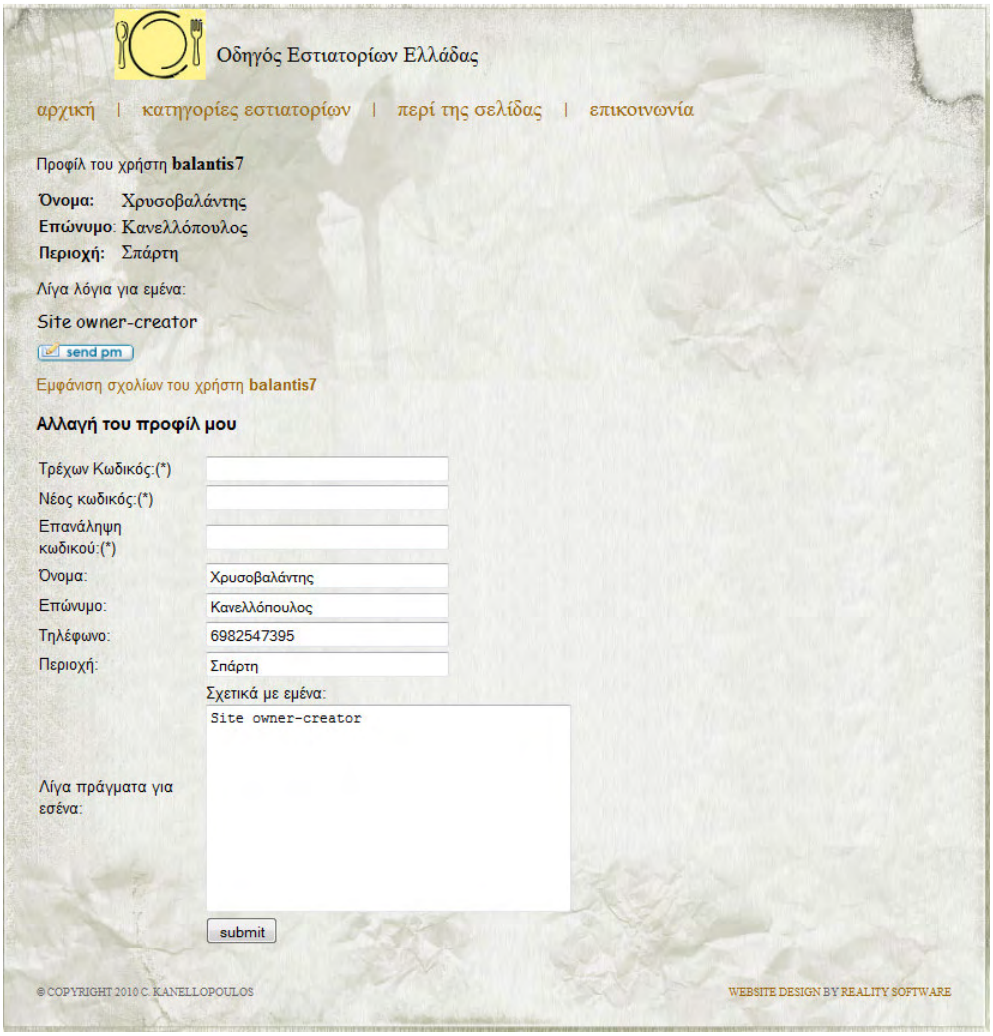



Εικόνα 18. Παράδειγμα της εφαρμογής που ακολουθεί την διάταξη της Εικόνα 17




Εικόνα 19. Διάταξη διεπαφής χρήστη για την σελίδα του προφίλ

Στην Εικόνα 20 βλέπουμε ένα παράδειγμα της εφαρμογής που ακολουθεί την διάταξη της Εικόνα 19.




Εικόνα 20. Παράδειγμα της εφαρμογής που ακολουθεί την διάταξη της Εικόνα 19

Στην Εικόνα 21 βλέπουμε την διάταξη της διεπαφής χρήστη που χρησιμοποιείται στις σελίδες των προσωπικών μηνυμάτων.


Εικόνα 21. Διάταξη διεπαφής χρήστη για τις σελίδες προσωπικών μηνυμάτων

- Στην περιοχή **μενού προσωπικών μηνυμάτων** εμφανίζονται οι διαθέσιμες κατηγορίες μηνυμάτων (εισερχόμενα - εξερχόμενα), και η επιλογή δημιουργίας νέου μηνύματος.
- Στον **πίνακα μηνυμάτων** εμφανίζονται τα μηνύματα που έχουν επιλεγεί από το μενού προσωπικών μηνυμάτων. Περισσότερα για τα παραπάνω στην ενότητα «4.4.3. Προσωπικά μηνύματα».

Στην Εικόνα 22 βλέπουμε παράδειγμα της εφαρμογής που ακολουθεί την διάταξη της Εικόνα 21.


Εικόνα 22. Παράδειγμα της εφαρμογής που ακολουθεί την διάταξη της Εικόνα 21

## 4.3. Λειτουργικότητα που παρέχεται σε μη εγγεγραμμένους χρήστες

Η αρχική σελίδα της εφαρμογής φαίνεται στην Εικόνα 23. Το κυρίαρχο στοιχείο της είναι ο χάρτης στον οποίο εμφανίζονται αποτελέσματα για τα οποία υπάρχουν γεωγραφικά δεδομένα. Επίσης χρησιμοποιείται για τον ορισμό περιοχής αναζήτησης από τον χρήστη. Δεξιά υπάρχει μια μπάρα η οποία περιέχει στο πάνω μέρος επιλογές για την σύνδεση του χρήστη και στο κάτω τις διαθέσιμες αναζητήσεις. Τέλος κάτω από το λογότυπο της σελίδας και επάνω από τον χάρτη, υπάρχει ένα μενού με σύνδεσμο στην αρχική σελίδα και μερικές ακόμα επιλογές που θα αναλύσουμε παρακάτω.


Εικόνα 23. Η αρχική σελίδα της εφαρμογής

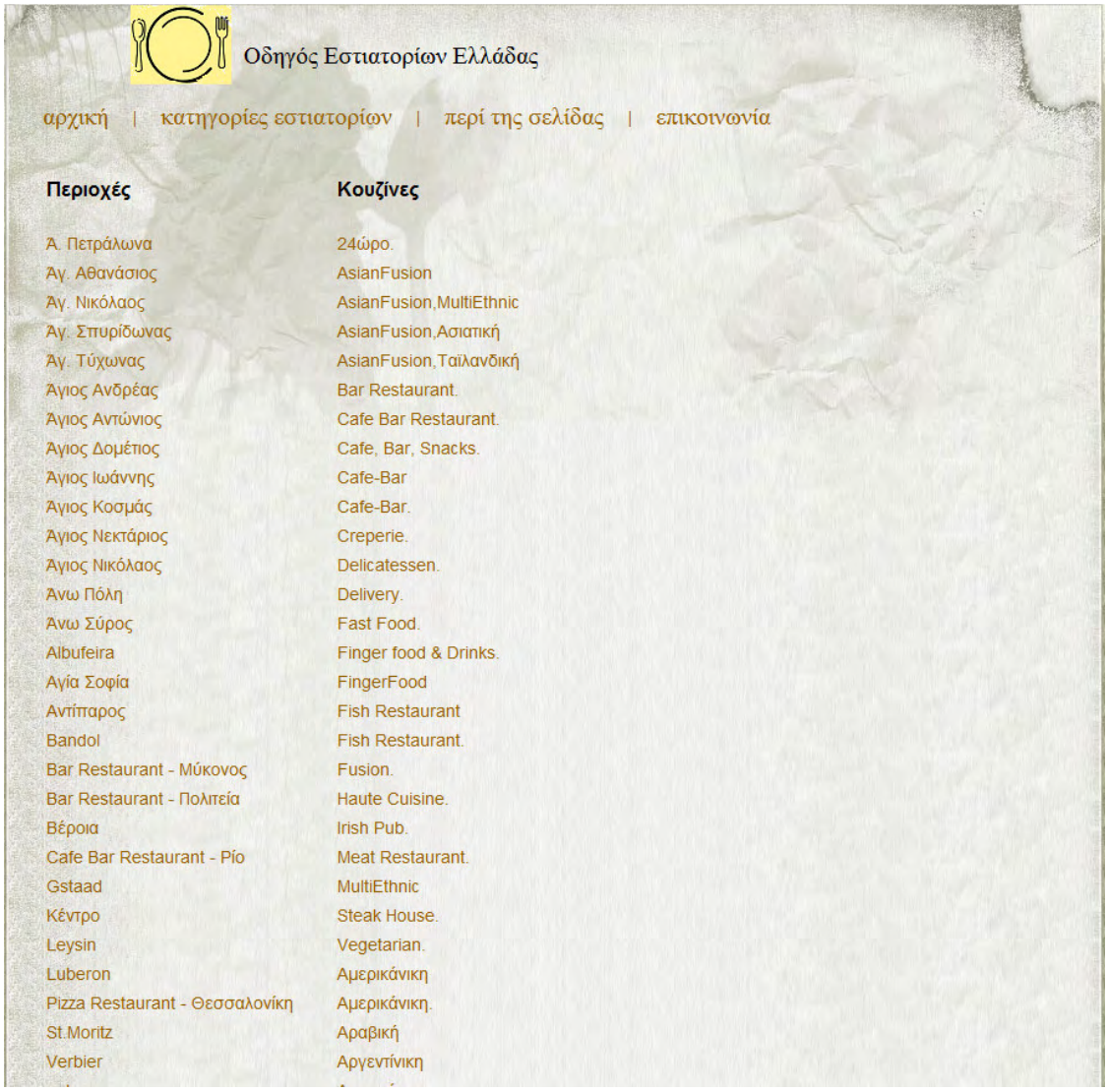
### 4.3.1. Λειτουργίες αναζήτησης

Παρέχονται πεδία αναζήτησης με βάση την περιοχή και με βάση την κουζίνα. Και οι δύο αυτές λειτουργίες αναζήτησης λειτουργούν με τον ίδιο ακριβώς τρόπο. Εισάγεις το επιθυμητό κλειδί στο ανάλογο πεδίο, ανάλογα τι αναζήτηση θέλεις να κάνεις, και πατάς το αντίστοιχο κουμπί search. Και τα δύο πεδία παρέχουν βοήθεια στην πληκτρολόγηση, όπως φαίνεται στην Εικόνα 24. Όταν ξεκινάμε να πληκτρολογούμε, μας εμφανίζονται πιθανά κλειδιά με βάση τι πληκτρολογούμε. Οι προτάσεις αυτές έχουν να κάνουν με τα δεδομένα που περιέχει η βάση για τη συγκεκριμένη κατηγορία αναζήτησης. Όσο συνεχίζουμε την πληκτρολόγηση μεταβάλλονται οι προτάσεις ανάλογα με αυτά που πληκτρολογούμε.

Η άλλη επιλογή αναζήτησης που μας δίνεται είναι με βάση τον χάρτη. Κεντράρουμε τον χάρτη να δείχνει την περιοχή στην οποία μας ενδιαφέρει να βρούμε εστιατόρια και πατάμε το κουμπί «search map!». Το αποτέλεσμα είναι να πάρουμε όσα εστιατόρια υπάρχουν στο σημείο που έχουμε κεντράρει.


Εικόνα 24. Η δυνατότητα βοήθειας στην πληκτρολόγηση στα πεδία αναζήτησης

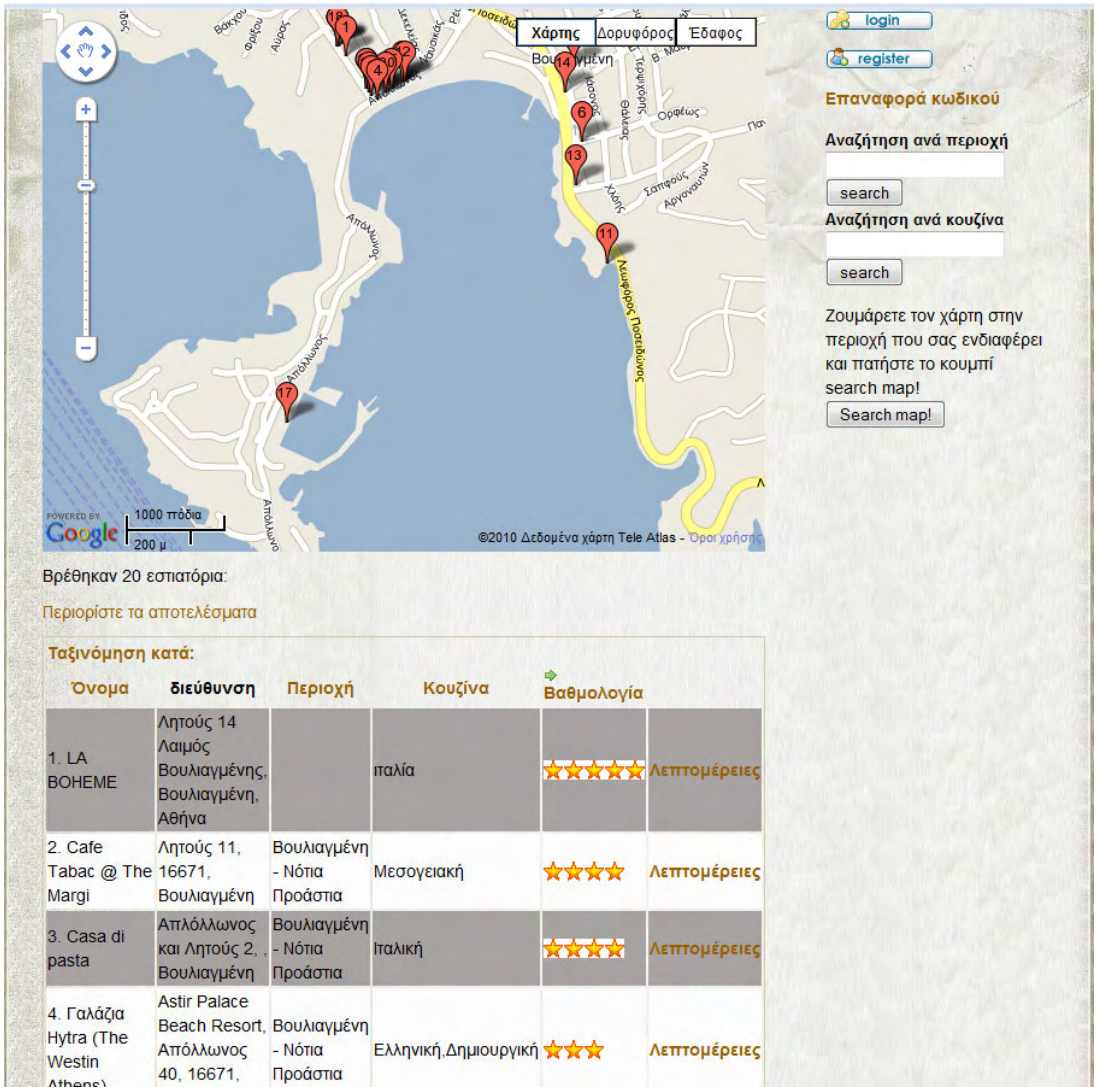
Η τρίτη επιλογή που έχουμε είναι να δούμε όλες τις διαθέσιμες κατηγορίες εστιατορίων και να επιλέξουμε μέσα από αυτές. Αυτό γίνεται επιλέγοντας τον σύνδεσμο «κατηγορίες εστιατορίων», που βρίσκεται στο μενού πάνω από τον χάρτη. Η σελίδα που θα εμφανιστεί θα είναι περίπου όπως αυτή της Εικόνα 25. Η σελίδα αυτή μας εμφανίζει 2 ειδών κατηγορίες εστιατορίων: σύμφωνα με όλες τις περιοχές στις οποίες υπάρχει καταχωρημένο εστιατόριο, και σύμφωνα με όλες τις κουζίνες για τις οποίες υπάρχει εστιατόριο. Πατώντας σε έναν από τους συνδέσμους μας εμφανίζονται όλα τα εστιατόρια τα οποία βρίσκονται στην συγκεκριμένη περιοχή ή έχουν την συγκεκριμένη κουζίνα.




Εικόνα 25. Η σελίδα "κατηγορίες εστιατορίων"

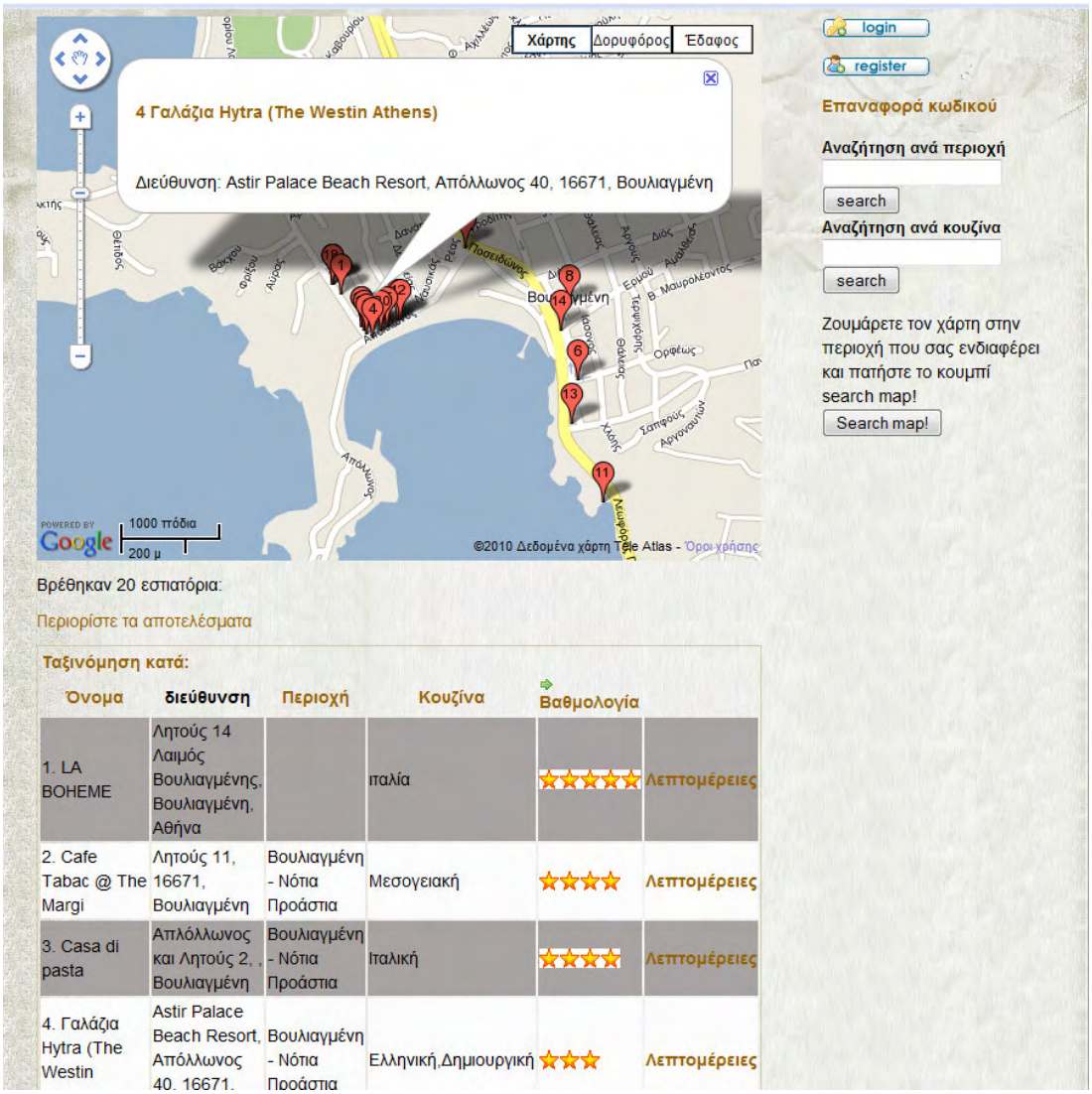
### 3.3.2. Εμφάνιση και φιλτράρισμα αποτελεσμάτων

Σε κάθε μία από τις περιπτώσεις αναζήτησης της προηγούμενης ενότητας τα αποτελέσματα εμφανίζονται με σχεδόν τον ίδιο τρόπο. Στην Εικόνα 26 φαίνονται τα αποτελέσματα με χρήση της αναζήτησης με βάση την περιοχή του χάρτη. Βλέπουμε πως τα αποτελέσματα εμφανίζονται πάνω στον χάρτη με αριθμημένα markers, και ως λίστα ακριβώς κάτω από αυτόν. Οι αριθμοί των markers αντιστοιχούν στους αριθμούς της λίστας, ώστε να εντοπίζεται ευκολότερα κάθε αποτέλεσμα και στον χάρτη.




Εικόνα 26. Αποτελέσματα αναζήτησης με βάση τον χάρτη όπως εμφανίζονται αμέσως μετά την αναζήτηση

Πατώντας πάνω σε κάποιο marker μας εμφανίζεται, όπως φαίνεται και στην Εικόνα 27, το info window που περιέχει τον τίτλο και την διεύθυνση του εστιατορίου. Ο τίτλος αποτελεί και σύνδεσμο προς την σελίδα με περισσότερες πληροφορίες και επιλογές για το συγκεκριμένο εστιατόριο.




Εικόνα 27. Το info window που εμφανίζεται όταν πατήσουμε πάνω σε κάποιο marker

Κάτω από τον χάρτη μας εμφανίζεται ο αριθμός των αποτελεσμάτων, ο σύνδεσμος περιορίστε τα αποτελέσματα, στον οποίο θα αναφερθούμε παρακάτω και στη συνέχεια ο πίνακας των αποτελεσμάτων. Τα αποτελέσματα αρχικά είναι ταξινομημένα με βάση την βαθμολογία που έχει πάρει κάθε ένα από τους χρήστες. Το πράσινο βέλος υποδεικνύει με βάση τι, είναι ταξινομημένα κάθε φορά. Μπορούμε να κάνουμε ταξινόμηση με βάση το όνομα, την περιοχή, την κουζίνα ή την βαθμολογία του εστιατορίου, πατώντας πάνω στον ανάλογο σύνδεσμο στην πρώτη γραμμή του πίνακα. Στην Εικόνα 28 βλέπουμε τα αποτελέσματα ταξινομημένα με βάση την κουζίνα.

Βρέθηκαν 22 εστιατόρια:

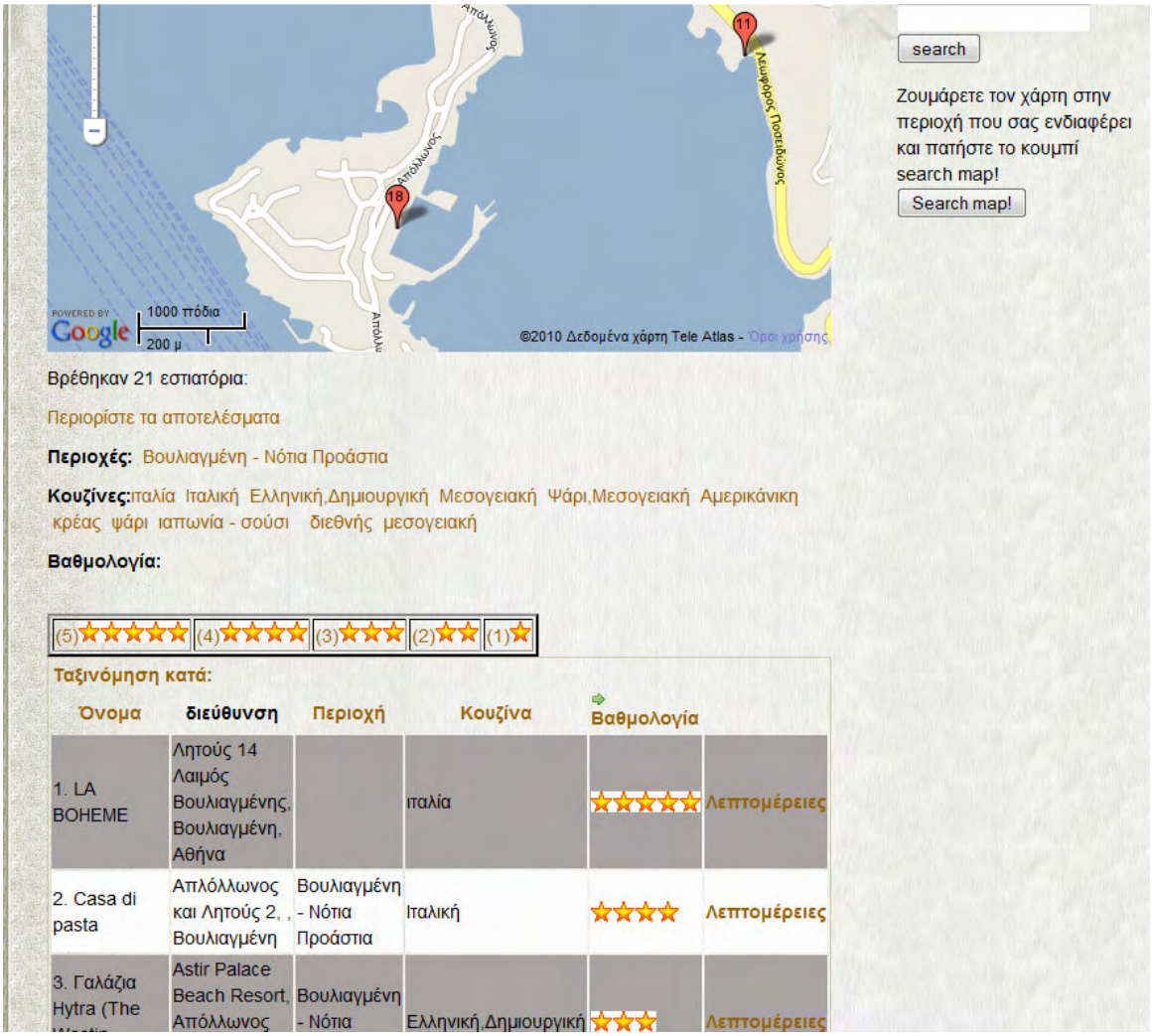
Περιορίστε τα αποτελέσματα

Ταξινόμηση κατά:

| Όνομα | διεύθυνση | Περιοχή | Κουζίνα | Βαθμολογία | |
|---|---|---|---|---|---|
| 1. BIER GARTEN | Ποσειδώνος 17 Πλ. Βουλιαγμένης, Βουλιαγμένη, Αθήνα | | | ★ | Λεπτομέρειες |
| 2. Silver Star | Λαιμός, Απόλλωνος και Λητούς 4, , Βουλιαγμένη | Βουλιαγμένη - Νότια Προάστια | Αμερικάνικη | ★ | Λεπτομέρειες |
| 3. MOORINGS | Μαρίνα Βουλιαγμένης, Βουλιαγμένη, Αθήνα | | διεθνής | ★ | Λεπτομέρειες |
| 4. Γαλάζια Hytra (The Westin Athens) | Astir Palace Beach Resort, Απόλλωνος 40, 16671, Βουλιαγμένη | Βουλιαγμένη - Νότια Προάστια | Ελληνική,Δημιουργική | ★★★ | Λεπτομέρειες |
| 5. MATSUHISA ATHENS | (ξεν. Astir Palace) Απόλλωνος 40 , Βουλιαγμένη, Αθήνα | | ιαπωνία - σούσι | ★ | Λεπτομέρειες |
| 6. LO STUZZICHINO | Ποσειδώνος 15 , Βουλιαγμένη, Αθήνα | | ιταλία | ★ | Λεπτομέρειες |
| 7. CASA DI PASTA | Απόλλωνος και Λητούς 2 , Βουλιαγμένη, Αθήνα | | ιταλία | ★ | Λεπτομέρειες |
| 8. LA BOHEME | Λητούς 14 Λαιμός Βουλιαγμένης, Βουλιαγμένη, Αθήνα | | ιταλία | ★ | Λεπτομέρειες |

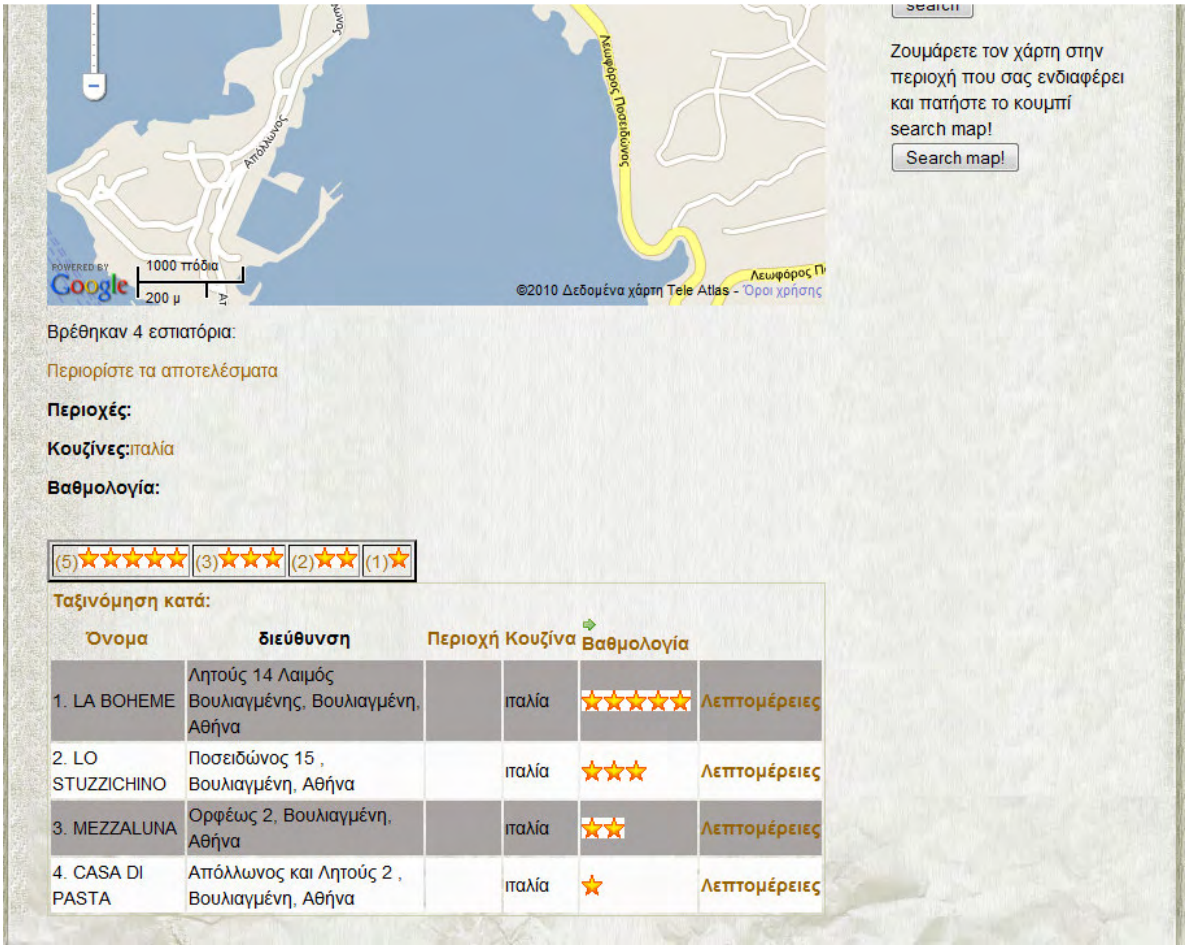
Εικόνα 28. Τα αποτελέσματα ταξινομημένα με βάση την κουζίνα

Κάτω από τον χάρτη, και πριν τα αποτελέσματα, βρίσκεται ο σύνδεσμος «περιορίστε τα αποτελέσματα». Πατώντας αυτόν τον σύνδεσμο, όπως φαίνεται και στην Εικόνα 29, μας εμφανίζονται μια σειρά από επιλογές («Περιοχές», «Κουζίνες» και «Βαθμολογία») για να περιορίσουμε τα αποτελέσματα που έχουμε. Στην προκειμένη περίπτωση έχουμε και τις 3 κατηγορίες περιορισμού διαθέσιμες καθώς κάναμε αναζήτηση με βάση τον χάρτη, και έχουν νόημα και οι 3 κατηγορίες. Στις αναζητήσεις με βάση την κουζίνα είναι διαθέσιμες μόνο οι κατηγορίες «Περιοχές» και «Βαθμολογία», ενώ στις αναζητήσεις με βάση την περιοχή είναι διαθέσιμες οι κατηγορίες «Κουζίνες» και «Βαθμολογία».




Εικόνα 29. Οι επιλογές για τον περιορισμό των αποτελεσμάτων

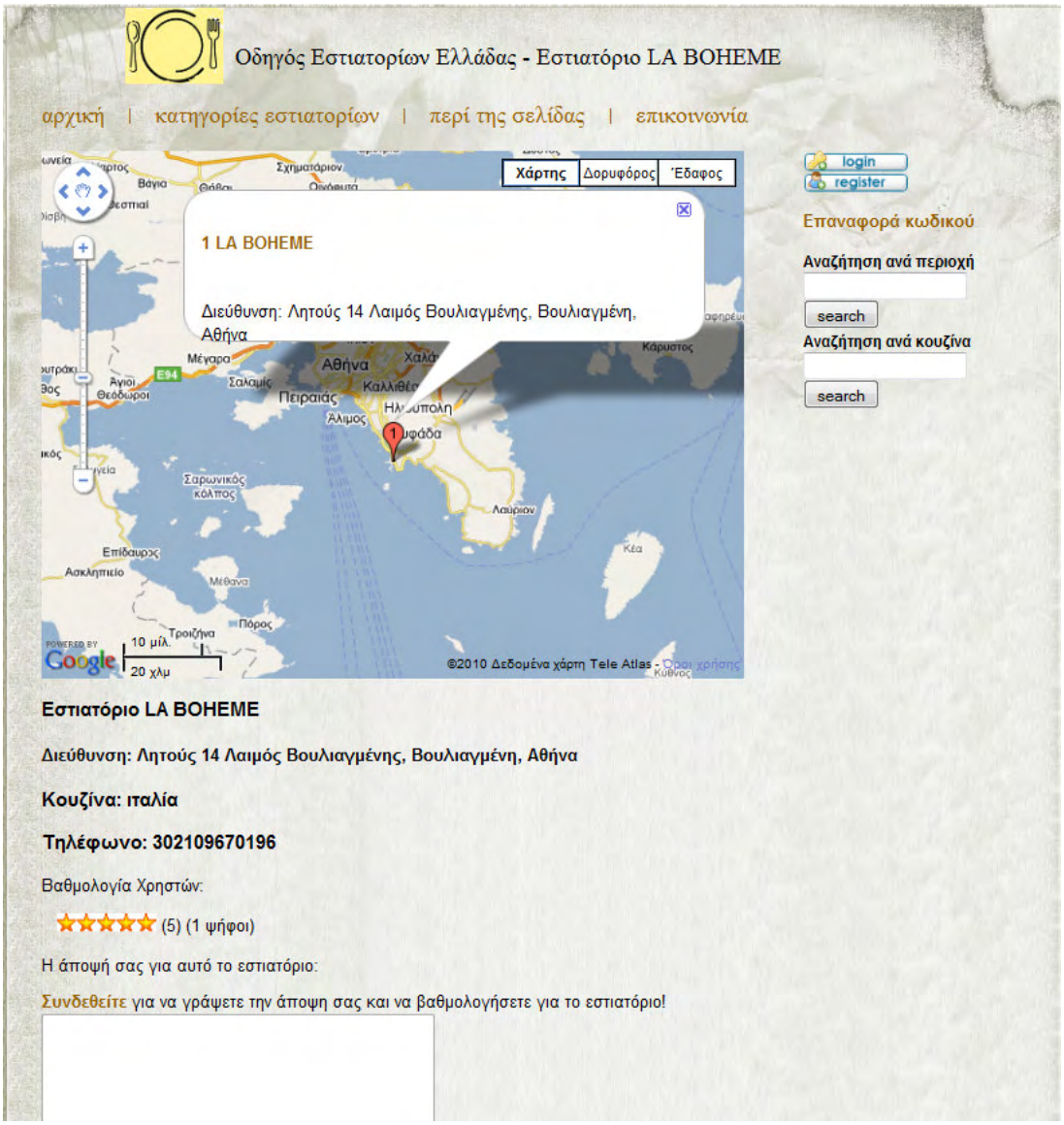
Πατώντας πάνω σε ένα σύνδεσμο κάποιας κατηγορίας, μας εμφανίζονται μόνο τα αποτελέσματα που έχουν το συγκεκριμένο χαρακτηριστικό. Στην Εικόνα 30 βλέπουμε τα αποτελέσματα μετά από την επιλογή της κουζίνας «Ιταλία». Βλέπουμε πως ανανεώνονται και τα αποτελέσματα στον χάρτη αλλά και οι επιλογές φιλτραρίσματος. Στην συνέχεια μπορούμε αν θέλουμε να φιλτράρουμε εκ νέου τα εναπομείναντα αποτελέσματα.




Εικόνα 30. Τα αποτελέσματα μετά από επιλογή της κουζίνας "Ιταλία" από το φιλτράρισμα

### 4.3.3. Σελίδα εστιατορίου και προφίλ χρηστών

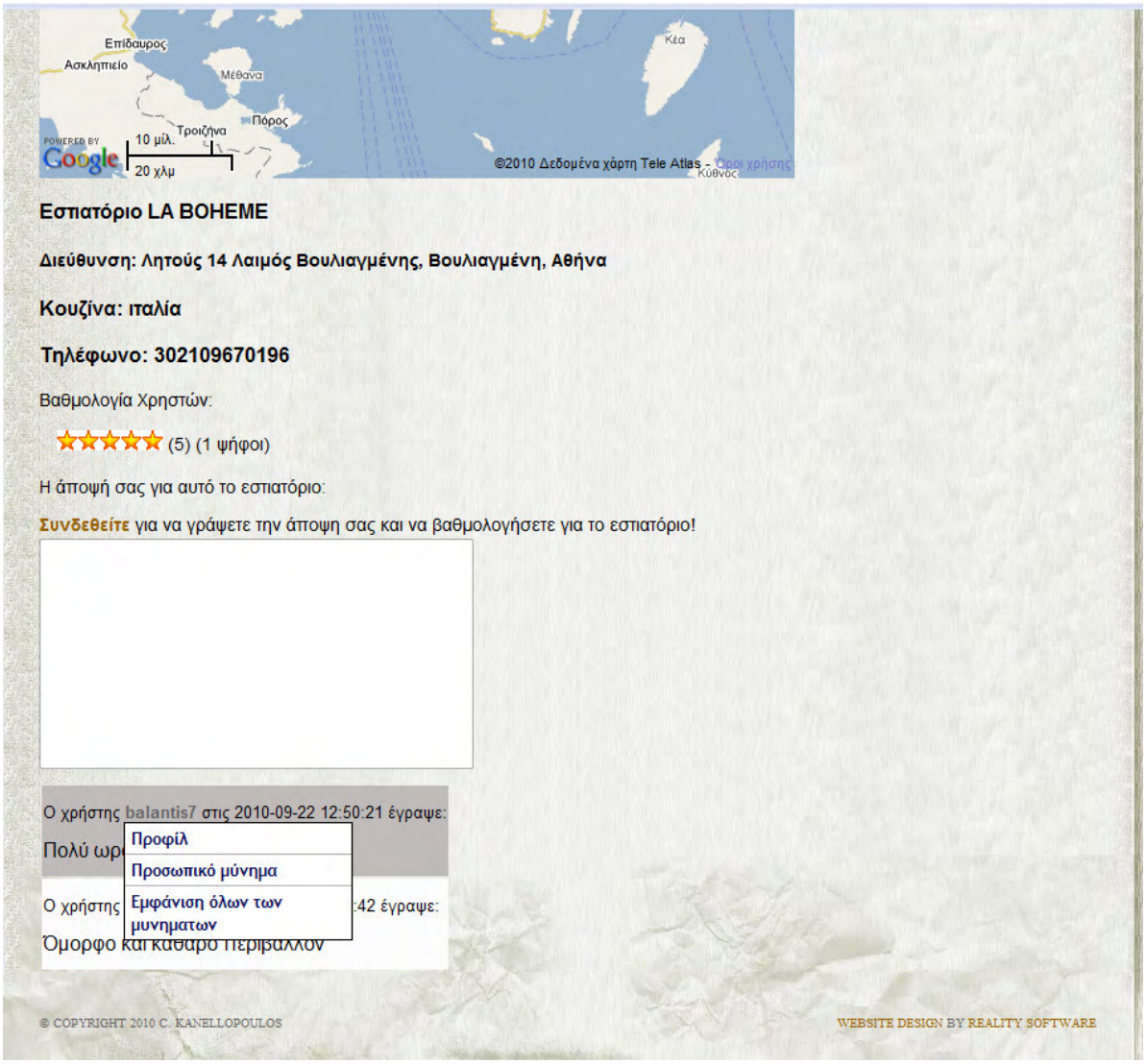
Στη συνέχεια αφού έχουμε καταλήξει στο εστιατόριο που μας ενδιαφέρει μπορούμε να πατήσουμε τον σύνδεσμο «Λεπτομέρειες» και να ανοίξουμε την σελίδα του εστιατορίου. Στην Εικόνα 31 βλέπουμε τη σελίδα του εστιατορίου «LA BOHEME». Βλέπουμε την θέση του εστιατορίου στον χάρτη (εφόσον υπάρχουν γεωγραφικά δεδομένα), την διεύθυνση, το τηλέφωνο, το είδος κουζίνας, την βαθμολογία που έχει πάρει από χρήστες καθώς και τον αριθμό τον ατόμων που έχει βαθμολογήσει.




Εικόνα 31. Σελίδα εστιατορίου "LA BOHEME"

Στην Εικόνα 32 βλέπουμε τα σχόλια του εστιατορίου «LA BOHEME». Μπορούμε να δούμε ποιος χρήστης έγραψε το σχόλιο και πότε ακριβώς. Οι μη εγγεγραμμένοι χρήστες δεν έχουν την δυνατότητα να εισάγουν σχόλιο ή βαθμολογία. Υπάρχει σύνδεσμος για να συνδεθούμε (εφόσον έχουμε λογαριασμό) ώστε να έχουμε τις παραπάνω δυνατότητες που θα παρουσιάσουμε σε επόμενη ενότητα. Αν βάλουμε τον δείκτη του ποντικιού πάνω σε κάποιο από τα ονόματα τον χρηστών μας εμφανίζεται το μενού που φαίνεται στην Εικόνα 32.

Πατώντας τον σύνδεσμο «προφίλ» βλέπουμε την σελίδα της Εικόνα 33. Σε αυτή την σελίδα βλέπουμε όλες τις πληροφορίες που είναι αποθηκευμένες για τον συγκεκριμένο χρήστη. Υπάρχουν και εδώ οι δυνατότητες που μας έβγαζε το μενού για τον χρήστη στην Εικόνα 32.



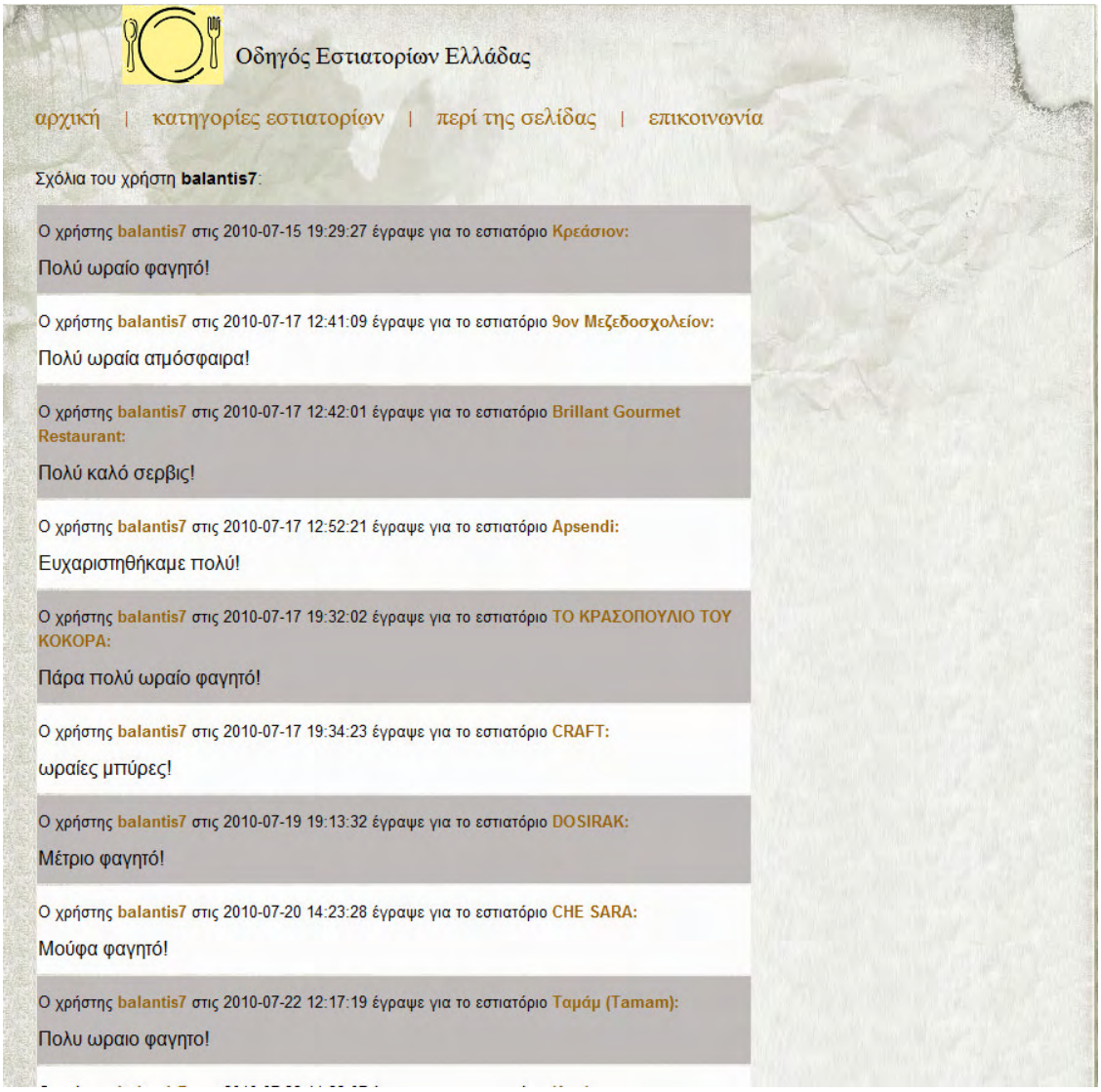

Εικόνα 32. Σχόλια του εστιατορίου "LA BOHEME",και επιλογές επικοινωνίας με κάποιο χρήστη


Εικόνα 33. Προφίλ χρήστη

Πατώντας το κουμπί «send pm» μας βγάζει στην σελίδα αποστολής προσωπικού μηνύματος που βλέπουμε στην Εικόνα 34. Όπως βλέπουμε η φόρμα αποστολής είναι απενεργοποιημένη αφού δεν είμαστε συνδεδεμένοι χρήστες και η λειτουργία αυτή δεν διατίθεται σε ανώνυμους χρήστες. Για την λειτουργία του προσωπικού μηνύματος θα μιλήσουμε στην ενότητα «4.4.3. Προσωπικά μηνύματα».

Αν από το προφίλ χρήστη της Εικόνα 33 πατήσουμε τον σύνδεσμο «Εμφάνιση σχολίων του χρήστη balantis7», μας επιστρέφεται ένας πίνακας με όλα τα σχόλια του συγκεκριμένου χρήστη, όπως αυτός της Εικόνα 35. Τα σχόλια παρατίθενται με χρονολογική σειρά και υπάρχει σύνδεσμος προς την σελίδα του εστιατορίου στο οποίο έγινε το κάθε σχόλιο.


Εικόνα 34. Σελίδα αποστολής προσωπικού μηνύματος με απενεργοποιημένη την φόρμα αποστολής αφού δεν είμαστε συνδεδεμένοι χρήστες




Εικόνα 35. Εμφάνιση όλων των σχολίων ενός χρήστη

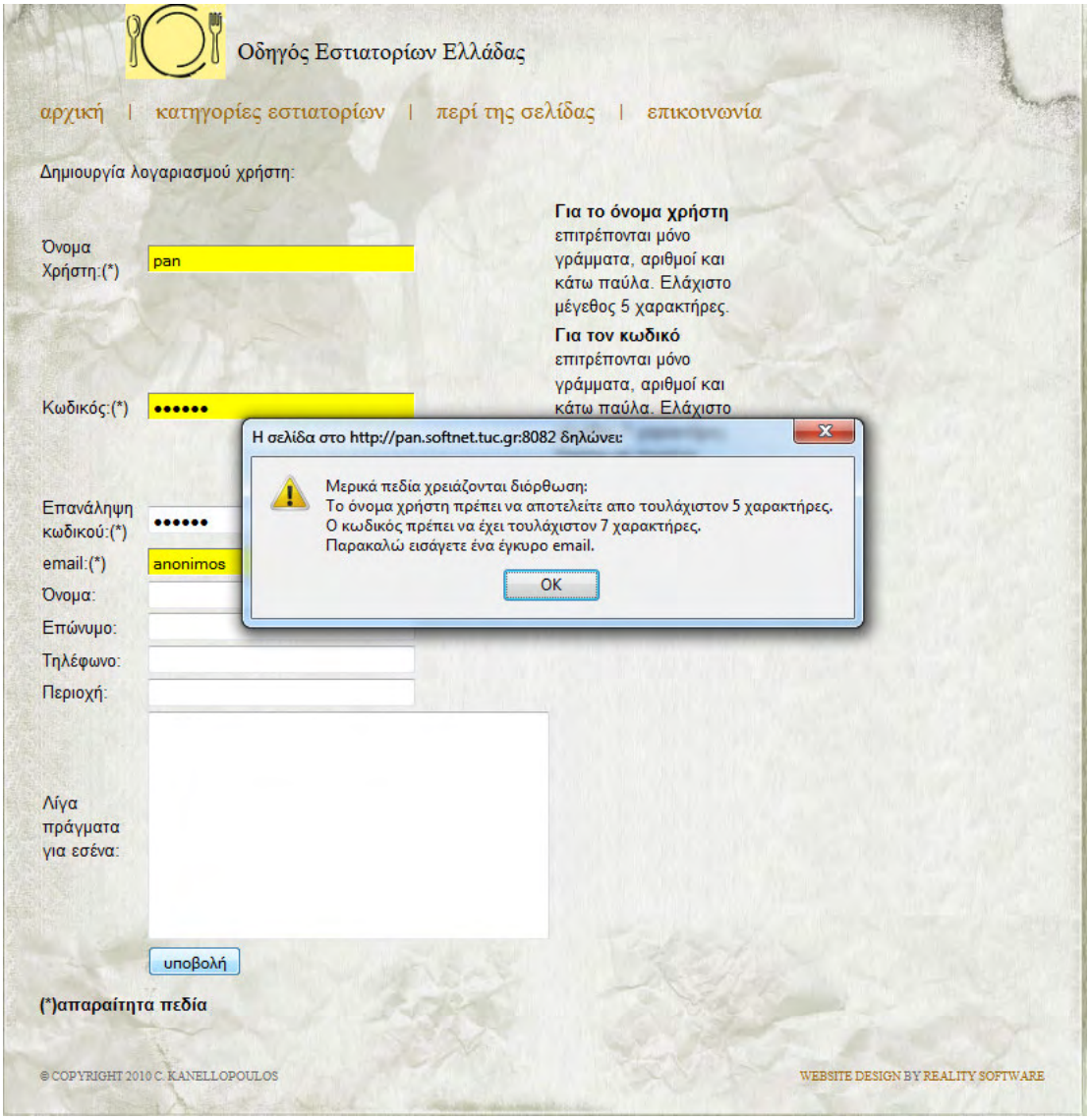
### 4.3.4. Δημιουργία λογαριασμού χρήστη

Η τελευταία λειτουργία που δεν έχουμε παρουσιάσει και μπορεί να κάνει ένας μη εγγεγραμμένος χρήστης είναι η δημιουργία λογαριασμού χρήστη. Στην αρχική σελίδα, από την δεξιά μπάρα επιλογών, πατάμε το κουμπί «register». Αυτό μας οδηγεί στην σελίδα δημιουργίας λογαριασμού η οποία φαίνεται στην Εικόνα 36. Τα πεδία όνομα χρήστη, κωδικός, επανάληψη κωδικού και email είναι απαραίτητο να συμπληρωθούν για την δημιουργία του λογαριασμού. Για το όνομα χρήστη πρέπει να δώσουμε ένα σύνολο τουλάχιστον 5 χαρακτήρων, οι οποίοι να είναι γράμματα, κάτω παύλα ή αριθμοί. Το πεδίο του κωδικού πρέπει να έχει μέγεθος τουλάχιστον 7 χαρακτήρες, οι οποίοι να αποτελούνται από γράμματα, κάτω παύλα και τουλάχιστον 1 αριθμό. Το πεδίο επανάληψη κωδικού πρέπει να περιέχει τους ίδιους χαρακτήρες με το πεδίο κωδικός. Τέλος το πεδίο email πρέπει οπωσδήποτε να περιέχει τον χαρακτήρα (@).


Εικόνα 36. Σελίδα δημιουργίας λογαριασμού χρήστη

Το σύστημα ελέγχει αυτόματα αν τηρούνται οι παραπάνω περιορισμοί πριν την υποβολή της φόρμας στον server. Αν δεν τηρείται κάποιος από αυτούς μας το επισημάνει και περιμένει να διορθώσουμε και να υποβάλλουμε την φόρμα ξανά. Στην Εικόνα 37 βλέπουμε το μήνυμα λάθους που μας δίνει το σύστημα για λανθασμένη εισαγωγή ονόματος χρήστη, κωδικού και email. Τα αντίστοιχα πεδία που είναι λάθος χρωματίζονται με κίτρινο.

Τα υπόλοιπα πεδία της φόρμας είναι προαιρετικά. Το μόνο πεδίο από αυτά που ελέγχεται είναι το πεδίο του τηλεφώνου το οποίο πρέπει να έχει μήκος 10 και να αποτελείται μόνο από νούμερα. Μετά την σωστή εισαγωγή των στοιχείων στην φόρμα, ελέγχεται στον server αν το όνομα χρήστη ή το email χρησιμοποιούνται, καθώς πρέπει να είναι μοναδικά, και δημιουργείται ο λογαριασμός. Στη συνέχεια το σύστημα μας καλωσορίζει, και μας ανοίγει την αρχική σελίδα ως συνδεδεμένος χρήστης.




Εικόνα 37. Λανθασμένη εισαγωγή ονόματος χρήστη, κωδικού και email και προειδοποιητικό μήνυμα από το σύστημα για αλλαγή των παραπάνω πεδίων

## 4.4. Λειτουργικότητα που παρέχεται μόνο σε εγγεγραμμένους χρήστες

### 4.4.1. Σύνδεση χρήστη, ανάκτηση χαμένου κωδικού και διαχείριση προφίλ

Η είσοδος του χρήστη στο σύστημα γίνεται από την σελίδα σύνδεσης, η οποία εμφανίζεται από την αρχική σελίδα, αν πατήσουμε το κουμπί «login». Στην Εικόνα 38 φαίνεται η σελίδα εισόδου. Συμπληρώνουμε στην φόρμα το όνομα χρήστη και τον κωδικό μας που δημιουργήσαμε όπως περιγράφεται στην Ενότητα «

4.3.4. Δημιουργία λογαριασμού χρήστη», και πατάμε υποβολή.


Εικόνα 38. Η σελίδα σύνδεσης χρήστη

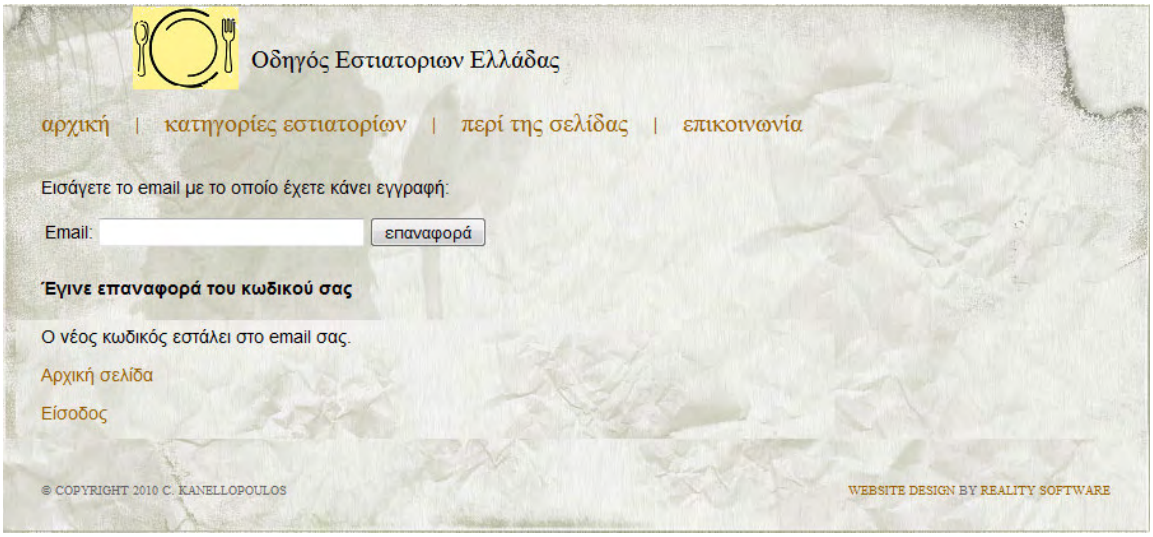
Εφόσον τα στοιχεία είναι σωστά το σύστημα μας συνδέει και μας καλωσορίζει όπως φαίνεται στην Εικόνα 39. Μετά από 3 δευτερόλεπτα μας προωθεί αυτόματα στην αρχική σελίδα. Αν δεν εισάγουμε σωστά στοιχεία βγαίνει πληροφοριακό μήνυμα ότι κάναμε λάθος και μας προτρέπει να εισάγουμε τα στοιχεία ξανά.


Εικόνα 39. Το μήνυμα καλωσορίσματος του χρήστη κατά την είσοδο

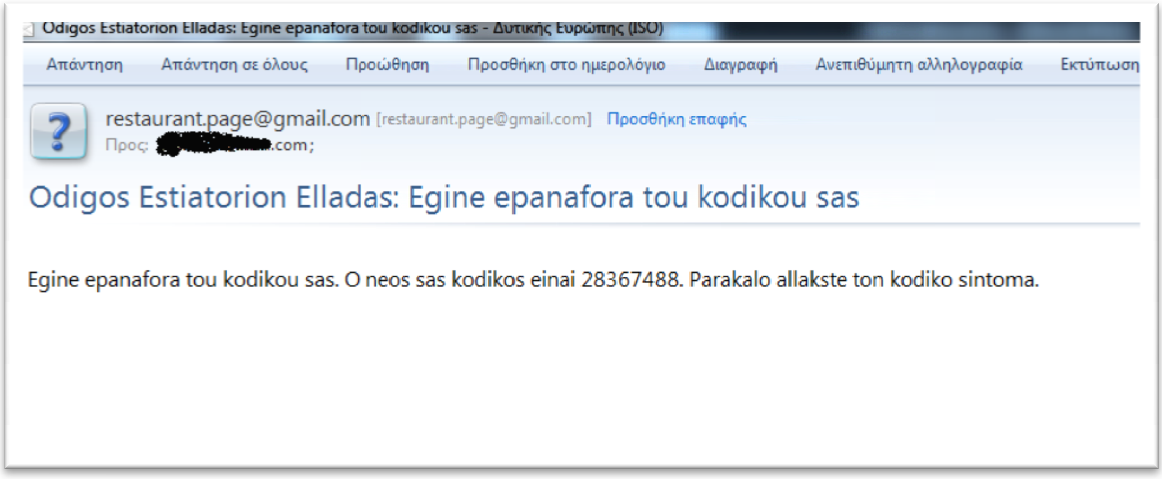
Παρέχεται η δυνατότητα αν έχουμε ξεχάσει τον κωδικό πρόσβασης να τον ανακτήσουμε. Αυτό γίνεται είτε μέσω του συνδέσμου «επαναφορά κωδικού» που υπάρχει στην αρχική σελίδα, είτε μέσω του ίδιου συνδέσμου που υπάρχει στην σελίδα σύνδεσης. Η σελίδα επαναφοράς κωδικού είναι αυτή που φαίνεται στην Εικόνα 40. Μας ζητάει να εισάγουμε το email με το οποίο είχαμε κάνει εγγραφή. Μετά την εισαγωγή, αφού πατήσουμε το κουμπί επαναφορά, το σύστημα δημιουργεί ένα νέο τυχαίο κωδικό για τον λογαριασμό που αντιστοιχεί στο email που δώσαμε και τον στέλνει σε αυτόν. Αν το email δεν αντιστοιχεί σε κανένα λογαριασμό μας βγάζει μήνυμα λάθους και μας προτρέπει να δοκιμάσουμε ξανά.


Εικόνα 40. Η σελίδα επαναφοράς κωδικού πρόσβασης

Στην Εικόνα 41 φαίνεται το μήνυμα του συστήματος ότι έγινε η επαναφορά και στην Εικόνα 42 φαίνεται τα email που στέλνει το σύστημα με τον νέο τυχαίο κωδικό πρόσβασης που δημιούργησε.




Εικόνα 41. Το μήνυμα του συστήματος μετά την επαναφορά του κωδικού

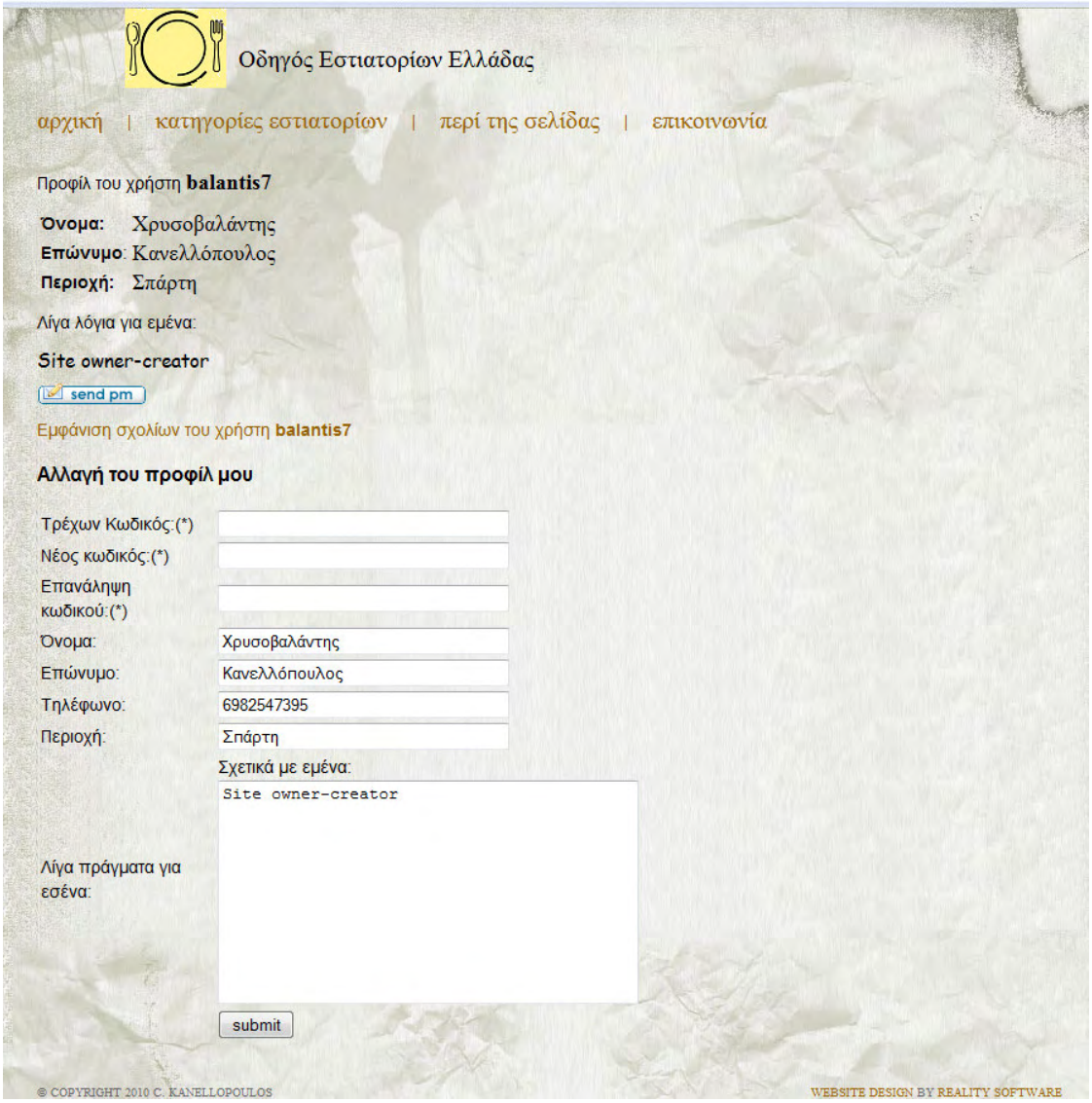



Εικόνα 42. Το email που στέλνει το σύστημα με τον νέο κωδικό

Ο κωδικός πρόσβασης, όπως και όλα τα στοιχεία που εισάγαμε κατά την εγγραφή μας εκτός από το email και το όνομα χρήστη, μπορούν να αλλάξουν από την σελίδα του προφίλ μας. Για να το κάνουμε αυτό πρέπει να συνδεθούμε στο σύστημα όπως περιγράψαμε παραπάνω και στην αρχική σελίδα, η οποία μετά την επιτυχή σύνδεση φαίνεται όπως στην Εικόνα 43, να πατήσουμε πάνω στο όνομα χρήστη μας, που φαίνεται στο δεξί μενού στο πάνω μέρος.

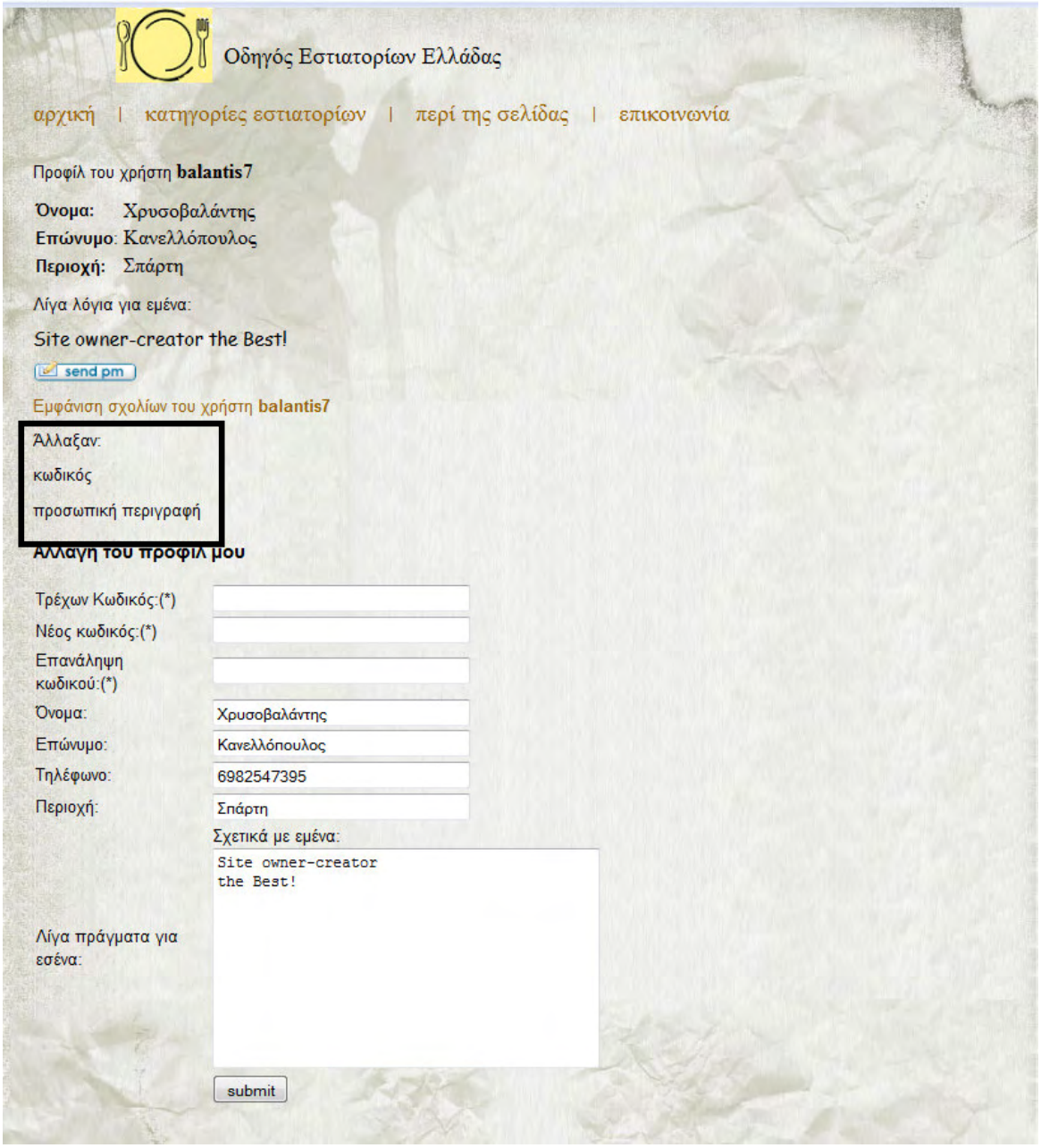

Εικόνα 43. Η αρχική σελίδα όπως φαίνεται μετά την επιτυχή σύνδεση χρήστη

Η σελίδα του προφίλ μας που θα εμφανιστεί θα μοιάζει με την Εικόνα 44. Η σελίδα αυτή είναι ίδια με αυτή της Εικόνα 33, μόνο που επειδή το προφίλ που εμφανίζεται είναι το προφίλ του χρήστη που είναι συνδεδεμένος εμφανίζει στο κάτω μέρος την φόρμα αλλαγής στοιχείων του προφίλ. Οποιοδήποτε άλλο προφίλ εκτός του δικού μας το βλέπουμε όπως δείχνει η Εικόνα 33. Με την φόρμα λοιπόν που εμφανίζεται στο προφίλ μας, μας δίνεται η δυνατότητα να αλλάξουμε όλα τα στοιχεία μας εκτός από το email και το όνομα χρήστη.




Εικόνα 44. Η σελίδα προφίλ συνδεδεμένου χρήστη

Για λόγους ασφαλείας, για να πραγματοποιηθεί οποιαδήποτε αλλαγή χρειάζεται να έχουμε εισαγάγει τον τρέχοντα κωδικό του λογαριασμού μας στο πρώτο πεδίο της φόρμας. Στη συνέχεια συμπληρώνουμε οποιοδήποτε από τα παρακάτω πεδία θέλουμε να αλλάξουμε. Ότι δεν πειράξουμε δεν θα αλλαχθεί. Για αλλαγή του κωδικού πρόσβασης συγκεκριμένα, πρέπει να εισάγουμε 2 φορές τον νέο κωδικό στα πεδία νέος κωδικός, και επανάληψη κωδικού. Μετά την υποβολή της φόρμας, και εφόσον ο τρέχον κωδικός είναι σωστός, το σύστημα μας πληροφορεί ποια πεδία του προφίλ τροποποιήθηκαν με μήνυμα πάνω από την φόρμα αλλαγής, όπως φαίνεται στην Εικόνα 45.

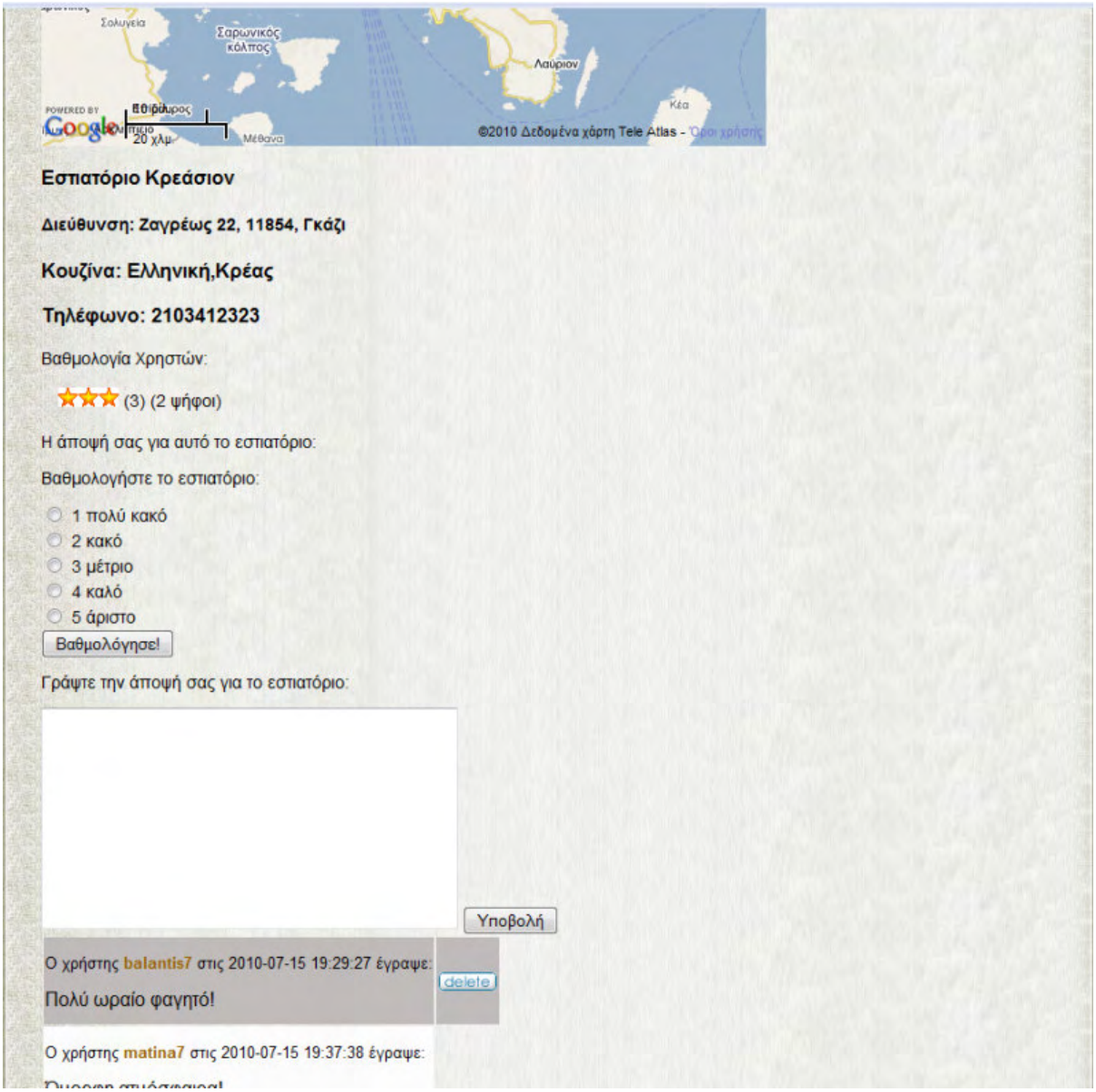



Εικόνα 45. Μήνυμα συστήματος για την επιτυχή αλλαγή κωδικού πρόσβασης και προσωπικής περιγραφής

### 3.4.2. Βαθμολόγηση και σχολιασμός εστιατορίου

Στους εγγεγραμμένους χρήστες δίνεται η δυνατότητα σχολιασμού και βαθμολόγησης οποιουδήποτε εστιατορίου. Για την εμφάνιση κάποιου εστιατορίου ακολουθούμε κάποια από τις διαδικασίες αναζήτησης που περιγράφονται στην Ενότητα «4.3.1. Λειτουργίες αναζήτησης». Στην σελίδα εστιατορίου, που για συνδεδεμένο χρήστη μοιάζει όπως η Εικόνα 46, μας εμφανίζεται μια λίστα βαθμολόγησης για να βαθμολογήσουμε το εστιατόριο και μία φόρμα κειμένου για να γράψουμε το σχόλιό μας. Η βαθμολόγηση δεν συνεπάγεται πως πρέπει να κάνουμε και σχολιασμό, και το αντίθετο. Μετά την βαθμολόγηση βλέπουμε πως αυξάνεται ο αριθμός των βαθμολογήσεων του εστιατορίου και ο βαθμός μας επηρεάζει τον μέσο όρο

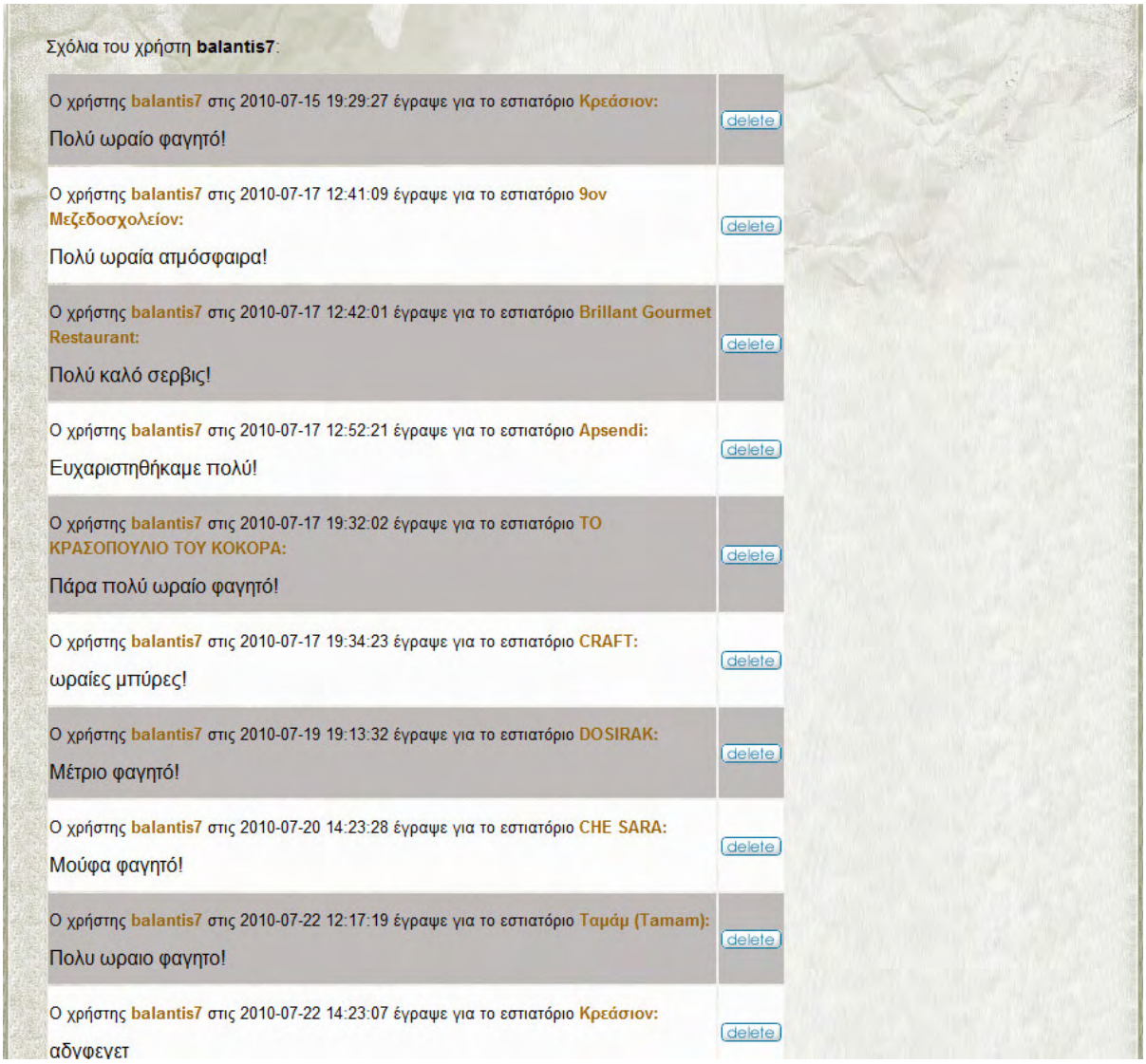
βαθμολογίας για αυτό το εστιατόριο. Αν πάλι κάνουμε σχόλιο, το σχόλιό μας εμφανίζεται στο τέλος της λίστας των σχολίων, καθώς αυτά εμφανίζονται με αύξουσα χρονολογική σειρά.




Εικόνα 46. Σελίδα εστιατορίου όπως εμφανίζεται σε συνδεδεμένο χρήστη

Επίσης μπορούμε ανά πάσα στιγμή να διαγράψουμε οποιοδήποτε σχόλιο έχουμε κάνει με τον λογαριασμό χρήστη που είμαστε συνδεδεμένοι. Αυτό γίνεται απλά πατώντας το κουμπί «delete» που βρίσκεται δίπλα σε όσα σχόλια έχουμε κάνει. Για να δούμε όλα τα σχόλιά μας, ανοίγουμε το προφίλ μας όπως περιγράφεται στην προηγούμενη ενότητα και πατάμε τον σύνδεσμο «εμφάνιση σχολίων». Τότε παίρνουμε μια λίστα με όλα τα σχόλιά μας όπως φαίνεται στην Εικόνα 47. Από εκεί μπορούμε απευθείας να διαγράψουμε οποιοδήποτε σχόλιο θέλουμε ή να ανοίξουμε την σελίδα του εστιατορίου και να κάνουμε από εκεί την διαγραφή. Η Εικόνα 47 είναι ίδια με την Εικόνα 35, μόνο που στην τελευταία δεν είναι διαθέσιμη η επιλογή της διαγραφής καθώς δεν είναι εγγεγραμμένος ο χρήστης. Βέβαια αν πατήσουμε «εμφάνιση

σχολίων» για οποιονδήποτε άλλο χρήστη εκτός του λογαριασμού μας, θα πάρουμε σαν αποτέλεσμα την Εικόνα 35.



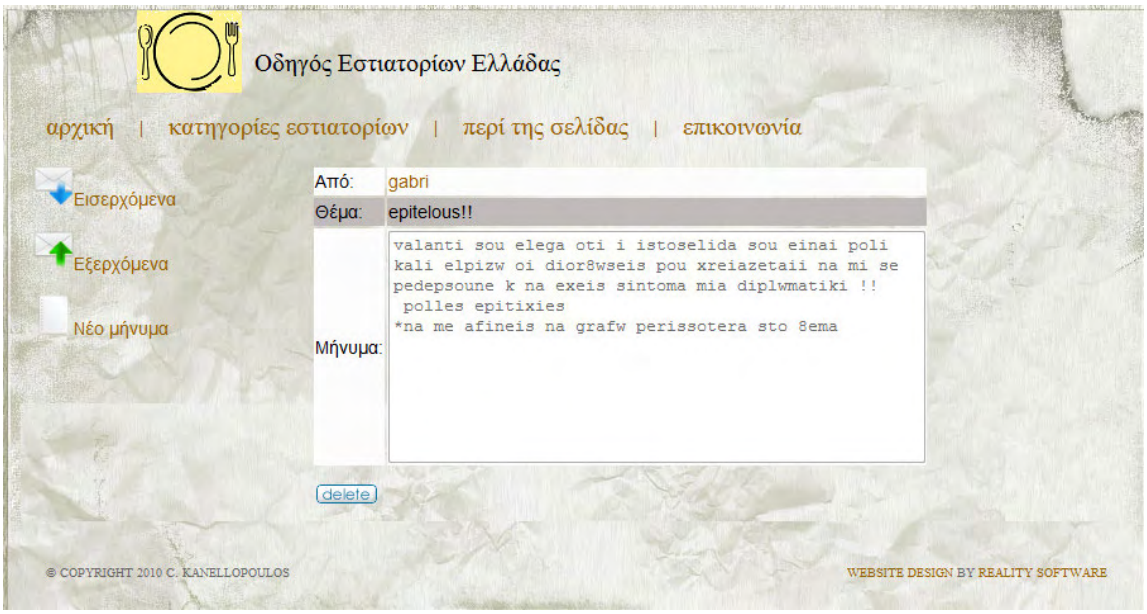

Εικόνα 47. Όλα τα σχόλια του χρήστη που είναι εκείνη την στιγμή συνδεδεμένος

### 4.4.3. Προσωπικά μηνύματα

Η τελευταία ενότητα για τις λειτουργίες που προσφέρονται στους εγγεγραμμένους χρήστες έχει να κάνει με την δυνατότητα αποστολής και λήψης προσωπικών μηνυμάτων μεταξύ των χρηστών. Κάθε χρήστης όταν συνδεθεί μπορεί να δει πόσα μη αναγνωσμένα μηνύματα έχει στο δεξί μενού της αρχικής σελίδας όπως φαίνεται στην Εικόνα 43, κάτω από το κουμπί «logout». Πατώντας πάνω σε αυτό τον σύνδεσμο μας βγάζει στην σελίδα των προσωπικών μηνυμάτων, έχοντας ενεργή την καρτέλα των εισερχομένων, όπως φαίνεται στην Εικόνα 48. Τα μηνύματα είναι ταξινομημένα με φθίνουσα χρονολογική σειρά, τα μη αναγνωσμένα εμφανίζονται με κλειστό φακελάκι ενώ τα αναγνωσμένα με ανοιχτό. Μας δίνεται η δυνατότητα

να διαβάσουμε κάποιο μήνυμα πατώντας τον σύνδεσμο «άνοιγμα» στο μήνυμα που επιθυμούμε, ή να διαγράψουμε κάποιο πατώντας στον σύνδεσμο «διαγραφή». Αν ανοίξουμε ένα μήνυμα θα πάρουμε σαν αποτέλεσμα κάτι περίπου σαν την Εικόνα 49.


Εικόνα 48. Η καρτέλα των εισερχόμενων μηνυμάτων




Εικόνα 49. Ανάγνωση ενός εισερχόμενου μηνύματος

Εκτός από τα εισερχόμενα μπορούμε να δούμε και τα μηνύματα τα οποία έχουμε στείλει πατώντας στη καρτέλα εξερχόμενα. Η σελίδα αυτή είναι όπως στην Εικόνα 50.


Εικόνα 50. Η καρτέλα των εξερχομένων μηνυμάτων

Τέλος μπορούμε να στείλουμε μήνυμα σε κάποιο άλλο μέλος πατώντας στην τρίτη καρτέλα, «Νέο μήνυμα». Η σελίδα που μας εμφανίζεται είναι αυτή της Εικόνα 51. Πρέπει απαραίτητα να εισάγουμε στο πρώτο πεδίο της φόρμας το όνομα χρήστη στον οποίο θέλουμε να στείλουμε το μήνυμα. Τα άλλα δύο πεδία είναι προαιρετικά. Αν εισάγουμε όνομα χρήστη που δεν υπάρχει το σύστημα μας βγάζει μήνυμα λάθους και μας προτρέπει να ξαναδοκιμάσουμε, αλλιώς στέλνει το μήνυμα και μας πληροφορεί πως το μήνυμα εστάλει.


Εικόνα 51. Σελίδα αποστολής προσωπικού μηνύματος

## 4.5. Λειτουργίες που προσφέρονται μόνο σε χρήστες με δικαιώματα διαχειριστή

Ο διαχειριστής είναι μια ειδική κατηγορία χρήστη. Σκοπό έχει την επίβλεψη της λειτουργίας της κοινότητας, και την τήρηση μιας σειράς κανόνων καλής συμπεριφοράς από όλους τους χρήστες-μέλη. Έτσι σε αυτήν την κατηγορία χρήστη δόθηκαν κάποιες επιπλέον δυνατότητες επέμβασης σε σχέση με τους υπόλοιπους χρήστες. Οι δυνατότητες αυτές είναι οι ακόλουθες:

### 4.5.1. Ανανέωση γεωγραφικών δεδομένων

Στην Εικόνα 52 βλέπουμε την αρχική σελίδα όπως εμφανίζεται μετά την είσοδο στο σύστημα με λογαριασμό που έχει δικαιώματα διαχειριστή. Παρατηρούμε πως στο δεξί μενού επιλογών εμφανίζεται ένα σύνολο επιλογών διαχειριστή. Η πρώτη από αυτές είναι η επιλογή «ανανέωση γεωγραφικών δεδομένων». Η λειτουργία αυτή σαρώνει όλα τα εστιατόρια που περιέχει η βάση δεδομένων της σελίδας και σε όσα δεν υπάρχουν γεωγραφικά δεδομένα χρησιμοποιεί την υπηρεσία geocoding τις Google, για τον εντοπισμό της θέσης του εστιατορίου με βάση την διεύθυνση που είναι καταχωρημένη στην βάση. Η συγκεκριμένη λειτουργία είναι χρονοβόρα και δεν πρέπει να πραγματοποιείται πάνω από μία φορά την ημέρα. Ο λόγος είναι πως η υπηρεσία geocoding διατίθεται με περιορισμό όσον αφορά τις συνολικές αιτήσεις ανά ημέρα αλλά και όσον αφορά τον ρυθμό αποστολής των αιτήσεων από κάθε IP. Λόγο του δεύτερου περιορισμού, και για να μειώσουμε τον ρυθμό αποστολής των αιτήσεων κατά την εκτέλεση της λειτουργίας, το σύστημα αφήνει ένα χρονικό διάστημα μερικών δευτερολέπτων μεταξύ των αιτήσεων. Έτσι η λειτουργία αυτή μπορεί να χρειαστεί μερικές δεκάδες λεπτά για να ολοκληρωθεί. Μετά το τέλος της μας εμφανίζεται ο αριθμός των εστιατορίων που άλλαξαν οι γεωγραφικές τους πληροφορίες.


Εικόνα 52. Η αρχική σελίδα όπως εμφανίζεται μετά την είσοδο με λογαριασμό με δικαιώματα διαχειριστή

### 4.5.2. Προσθήκη εστιατορίου

Η λειτουργία αυτή είναι προσβάσιμη από την αρχική σελίδα, από το δεξί μενού επιλογών. Με την λειτουργία αυτή δίνεται η δυνατότητα να προστεθεί ένα καινούργιο εστιατόριο στην βάση δεδομένων, και να είναι διαθέσιμο σε όλους τους χρήστες. Πατώντας τον σύνδεσμο «προσθήκη εστιατορίου» μας εμφανίζεται η σελίδα που φαίνεται στην Εικόνα 53. Συμπληρώνουμε την φόρμα με τα κατάλληλα δεδομένα και μετακινούμε τον marker στον χάρτη στο σημείο που βρίσκεται το εστιατόριο. Η μετακίνηση γίνεται με drag n drop. Αν δεν μετακινήσουμε καθόλου τον marker τότε τα γεωγραφικά δεδομένα του εστιατορίου θα παραμείνουν μηδενικά. Μετά την υποβολή της φόρμας αποθηκεύεται το εστιατόριο και μας ανοίγει η σελίδα του εστιατορίου.


Εικόνα 53. Η σελίδα προσθήκης νέου εστιατορίου

### 4.5.3. Διαχείριση χρηστών

Στους διαχειριστές έχει δοθεί η δυνατότητα να διαγράφουν λογαριασμούς χρηστών ή να τους δίνουν δικαιώματα διαχειριστή. Αυτό γίνεται από τον σύνδεσμο «εμφάνιση όλων των χρηστών» που βρίσκεται στο δεξιό μενού της αρχικής σελίδας. Η σελίδα που μας εμφανίζεται αν ανοίξουμε τον παραπάνω σύνδεσμο είναι αυτή της Εικόνα 54. Εκεί βλέπουμε όλους τους εγγεγραμμένους χρήστες. Μπορούμε να διαγράψουμε όποιον λογαριασμό θέλουμε ή να του δώσουμε δικαιώματα διαχειριστή. Τα username είναι σύνδεσμοι προς το προφίλ κάθε χρήστη.

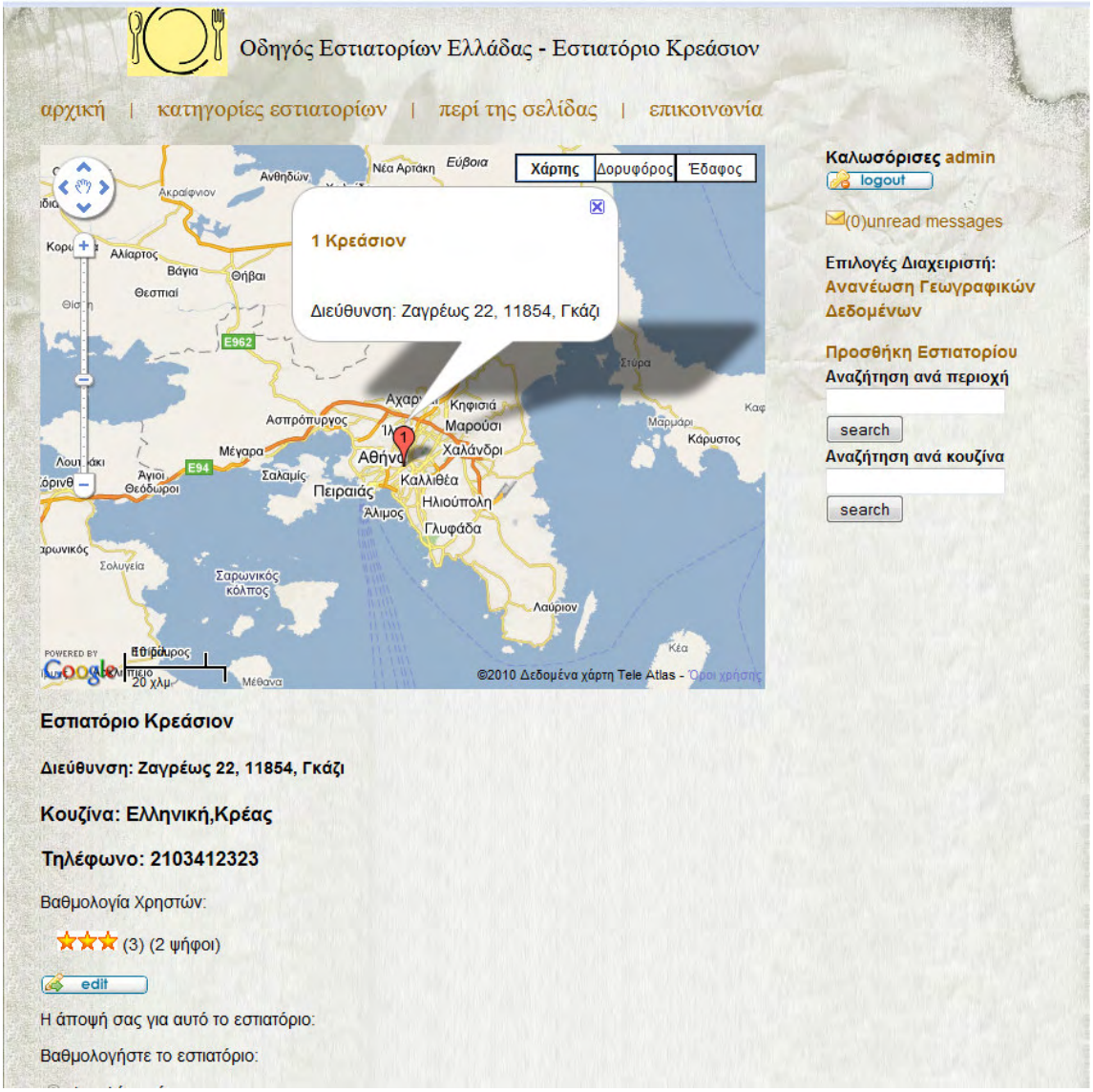

Εικόνα 54. Σελίδα διαχείρισης χρηστών

### 4.5.4. Αλλαγή στοιχείων εστιατορίου

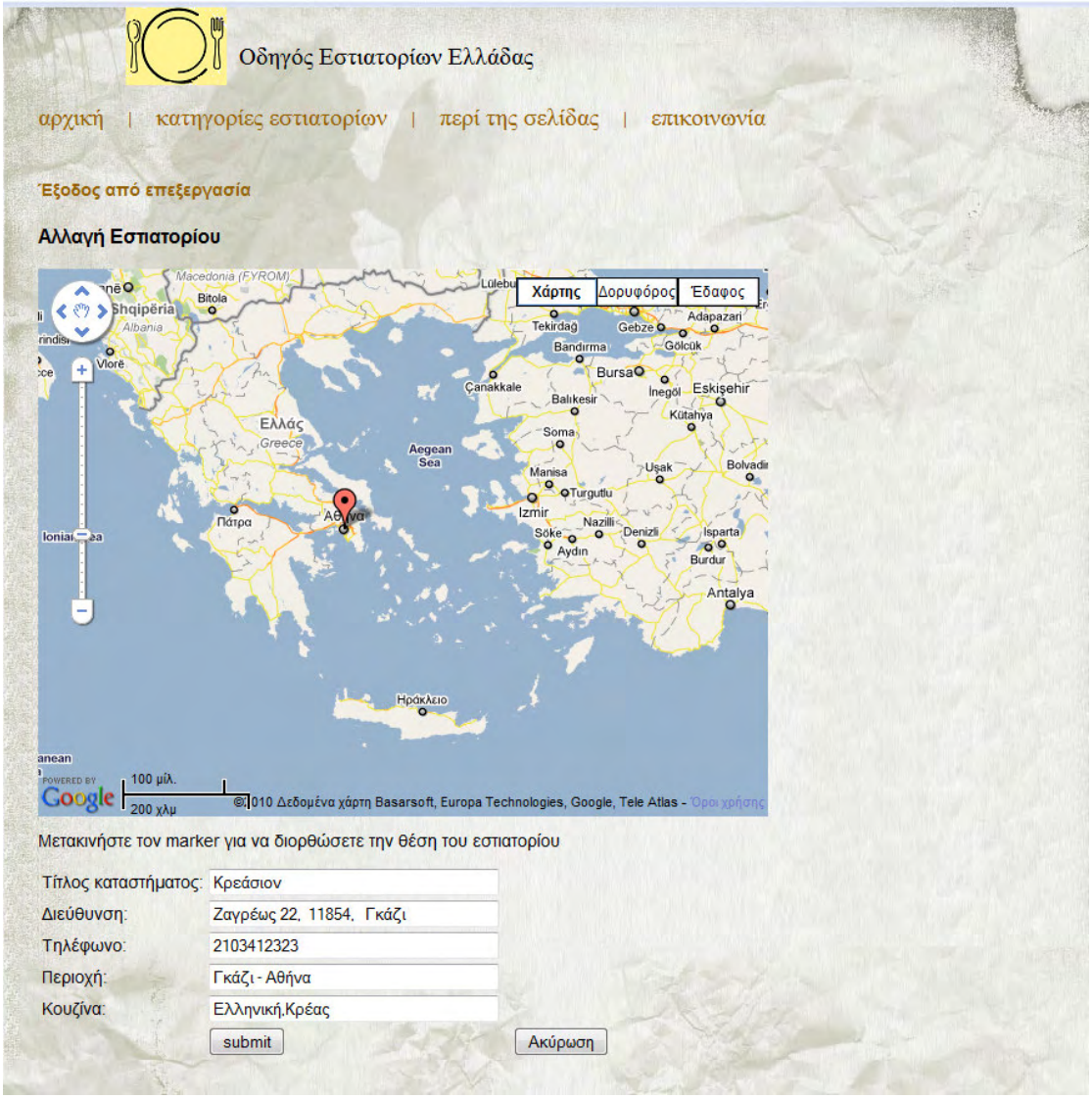
Ο διαχειριστής έχει την δυνατότητα να τροποποιεί τα δεδομένα των εστιατορίων. Σε κάθε σελίδα εστιατορίου, αν είναι συνδεμένος κάποιος ως διαχειριστής, του εμφανίζεται το κουμπί «edit», όπως φαίνεται στην Εικόνα 55. Αν πατήσουμε το κουμπί αυτό μας εμφανίζεται η σελίδα της Εικόνα 56. Μπορούμε να αλλάξουμε όλα τα στοιχεία της φόρμας αλλά και να μετακινήσουμε τον marker στον χάρτη για να αλλάξουμε την θέση του εστιατορίου. Μετά την υποβολή της φόρμας το σύστημα μας ενημερώνει ποια πεδία έχουν αλλάξει με ανάλογο μήνυμα πριν από την φόρμα.

### 4.5.5. Δυνατότητες διαγραφής σχολίων

Μια τελευταία δυνατότητα που δίνεται σε χρήστες με δικαιώματα διαχειριστή είναι η διαγραφή όλων των σχολίων ανεξαρτήτως αν τα έχουν κάνει οι ίδιοι. Η δυνατότητα αυτή δίνεται για την τήρηση κανόνων ευπρεπούς έκφρασης οι οποίοι μπορεί να μην τηρούνται από πολλούς χρήστες, αλλά και για την διόρθωση ανακριβειών που μπορεί να διαπιστώνει ο διαχειριστής στα σχόλια των υπολοίπων χρηστών.




Εικόνα 55. Εμφάνιση κουμπιού "edit" στους διαχειριστές

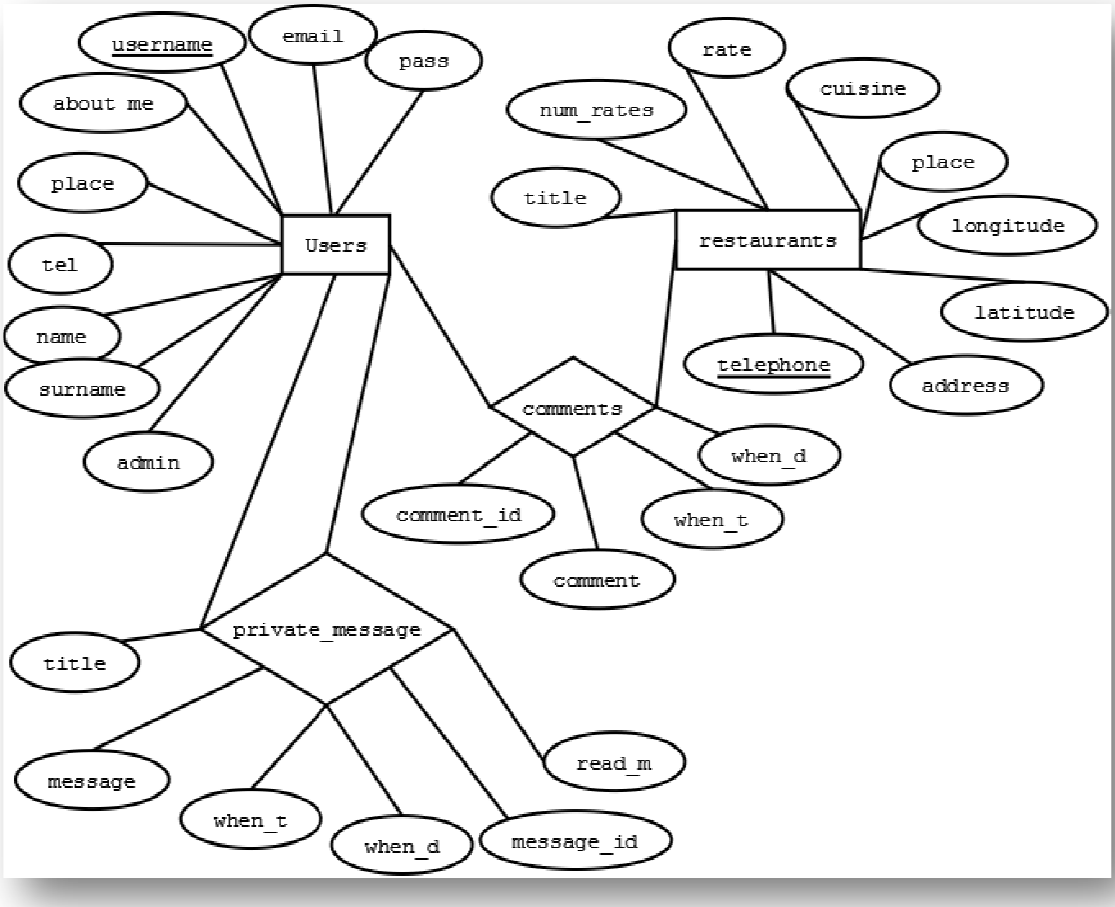



Εικόνα 56. Επεξεργασία δεδομένων εστιατορίου

# 5. Λεπτομέρειες Υλοποίησης της Εφαρμογής μας

Η εφαρμογή αναπτύχθηκε με χρήση του NetBeans IDE 6.8. Σαν web-server δοκιμών χρησιμοποιήθηκε ο Apache TomCat 6.0.29. Το σύστημα διαχείρισης βάσης δεδομένων που χρησιμοποιήθηκε ήταν το MySQL 5.1.47. Το λειτουργικό σύστημα που υποστήριζε όλο το προηγούμενο λογισμικό ήταν τα Windows 7 Ultimate 32 bit.

## 5.1. Το διάγραμμα οντοτήτων-συσχετίσεων (entity-relationship) και το σχεσιακό σχήμα της βάσης δεδομένων της εφαρμογής




Εικόνα 57. Το διάγραμμα οντοτήτων - συσχετίσεων της βάσης δεδομένων της εφαρμογής

Στην Εικόνα 57 βλέπουμε το διάγραμμα οντοτήτων – συσχετίσεων της βάσης δεδομένων της εφαρμογής. Η οντότητα Users μοντελοποιεί τους εγγεγραμμένους χρήστες στην εφαρμογή. Οι εγγεγραμμένοι χρήστες μπορούν να στέλνουν μεταξύ τους προσωπικά μηνύματα. Η λειτουργία αυτή αναπαρίσταται από την σχέση private message. Η οντότητα restaurants αντιπροσωπεύει όλα τα εστιατόρια που είναι καταχωρημένα στην εφαρμογή. Οι χρήστες έχουν

την δυνατότητα να σχολιάζουν τα εστιατόρια. Η λειτουργία αυτή αναπαρίσταται από την σχέση comments.

Από το παραπάνω διάγραμμα προκύπτει το ακόλουθο σχεσιακό σχήμα:

**Users**(username,email,pass,name,surname,tel,place,about_me,admin)

- Username: το όνομα χρήστη που δίνεται κατά την εγγραφή
- Email: το email του χρήστη που δίνει κατά την εγγραφή
- Pass: ο κωδικός (κρυπτογραφημένος) που χρησιμοποιείται για την είσοδο στο σύστημα
- Name: το όνομα του χρήστη
- Surname: το επώνυμο του χρήστη
- Tel: το τηλέφωνο του χρήστη
- Place: η περιοχή που διαμένει ο χρήστης
- About_me: περιγραφή που γράφει ο χρήστης για τον εαυτό του
- Admin: λογική μεταβλητή που είναι 1 αν ο χρήστης έχει δικαιώματα διαχειριστή, αλλιώς είναι 0.

**Restaurants**(telephone,title,address,place,cuisine,latitude,longitude,rate,num_rates)

- Telephone: το τηλέφωνο του εστιατορίου
- Title: το όνομα του εστιατορίου
- Address: η διεύθυνση του εστιατορίου
- Place: η περιοχή που βρίσκεται το εστιατόριο
- Cuisine: το είδος κουζίνας που σερβίρεται στο εστιατόριο
- Latitude: το γεωγραφικό πλάτος στο οποίο βρίσκεται το εστιατόριο
- Longitude: το γεωγραφικό μήκος στο οποίο βρίσκεται το εστιατόριο
- Rate: ο μέσος όρος τον βαθμολογιών που έχει λάβει το εστιατόριο από χρήστες
- Num_rates: ο συνολικός αριθμός χρηστών που έχουν αξιολογήσει το εστιατόριο

**Comments**(comment_id,username(FK),telephone(FK),when_t,when_d,comment)

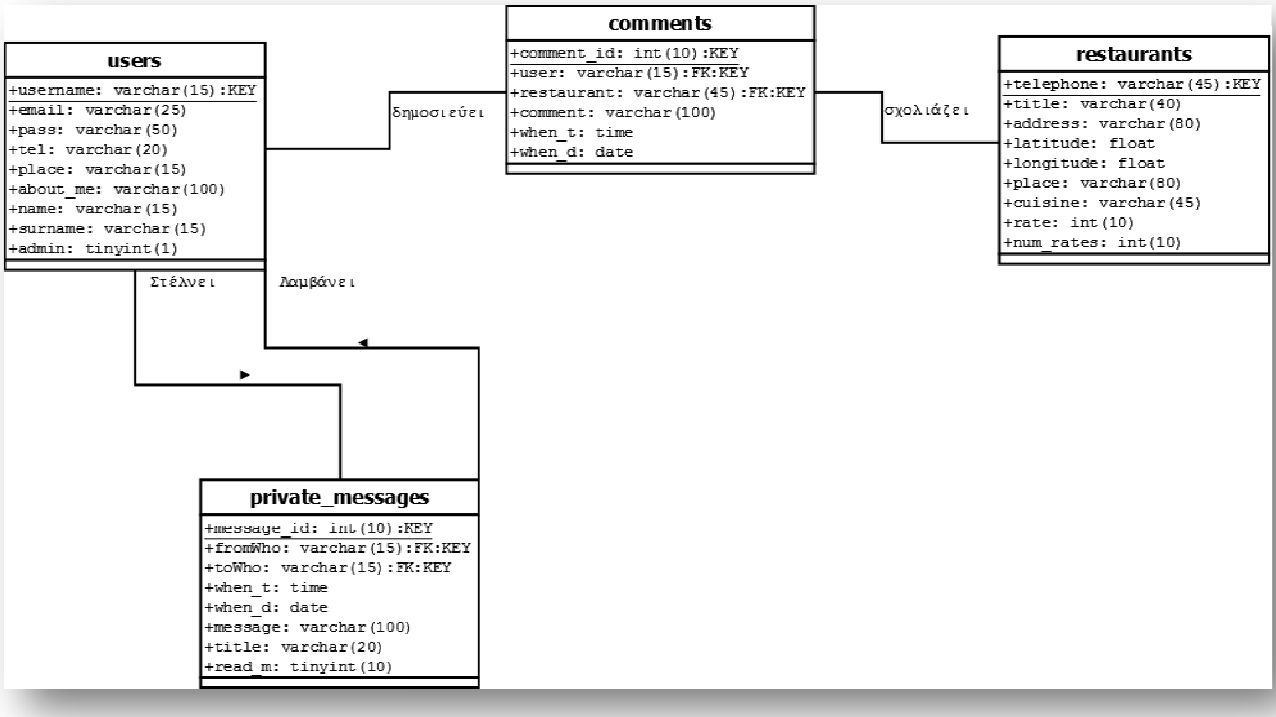
- Comment_id: μοναδικός αριθμός που προσδιορίζει κάθε σχόλιο
- Username: το όνομα χρήστη του χρήστη που έχει κάνει το σχόλιο
- Telephone: το τηλέφωνο του εστιατορίου στο οποίο αναφέρεται το σχόλιο
- When_t: η ώρα που έγινε το σχόλιο
- When_d: η ημερομηνία που έγινε το σχόλιο
- Comment: το κείμενο του σχολίου

**Private_message**(message_id,from(FK),to(FK),title,message,when_t,when_d,read_m)

- Message_id: μοναδικός αριθμός που προσδιορίζει κάθε μήνυμα
- From: το όνομα χρήστη του χρήστη που έστειλε το μήνυμα
- To: το όνομα χρήστη του χρήστη στον οποίο αποσταλεί το μήνυμα
- Title: ο τίτλος του μηνύματος
- Message: το κείμενο του μηνύματος
- When_t: η ώρα που εστάλει το μήνυμα
- When_d: η ημερομηνία που εστάλει το μήνυμα

- Read_m: λογική μεταβλητή που είναι 1 αν το μήνυμα έχει αναγνωσθεί, αλλιώς είναι 0

Το UML διάγραμμα που προκύπτει είναι αυτό της Εικόνα 58.




Εικόνα 58.Το διάγραμμα UML της βάσης δεδομένων

## 5.2. Η αρχιτεκτονική που χρησιμοποιήθηκε

Για την υλοποίηση της σελίδας που παρουσιάσαμε στο προηγούμενο κεφάλαιο, χρησιμοποιήθηκε η αρχιτεκτονική MVC, όπως παρουσιάστηκε στην Ενότητα «3.3.3. Εφαρμογή της αρχιτεκτονικής MVC με χρήση servlet και JSP». Στην παραπάνω αρχιτεκτονική, όπως φαίνεται και στην Εικόνα 12, κυρίαρχο ρόλο παίζει ένα servlet, ο Controller. Η υλοποίηση για τον Controller που παρουσιάζεται παρακάτω δόθηκε ως υλικό εργαστηρίου στα πλαίσια του μαθήματος ΕΚΠ403 «Ανάπτυξη Εφαρμογών Πληροφοριακών Συστημάτων στο Διαδίκτυο» του τμήματος ΗΜΜΥ.

Στην αρχιτεκτονική που χρησιμοποιήθηκε, η οντότητα του μοντέλου που ορίζεται στο MVC, μοντελοποιήθηκε με ένα σύνολο από Data Access Objects [35], τα οποία θα αναλυθούν παρακάτω, και από ένα interface με το όνομα Action, το οποίο υλοποιείται για κάθε δυνατή λειτουργία της εφαρμογής. Το interface Action φαίνεται στην Εικόνα 59. Η συνάρτηση execute στην γραμμή 47, είναι αυτή που ουσιαστικά εκτελεί την κάθε λειτουργία. Για κάθε λειτουργία της εφαρμογής δημιουργείται μια κλάση η οποία υλοποιεί το interface Action. Παρακάτω, σε επόμενη ενότητα, θα περιγραφούν εκτενέστερα οι σημαντικότερες λειτουργίες.

```
22  import javax.servlet.*;
23  import javax.servlet.http.*;
24  import java.io.*;
25
26  /**
27   * Interface for Action objects
28   *
29   * <p>This interface is used to provide a generic interface to Action objects,
30   * which are used to implement a request action. Action object are actually
31   * commnad objects</p>
32   *
33   * <p> The Action interface defines three core methods: execute, getView and
34   * getModel</p>
35   *
36   * <p>The execute method, when implemented, will perform any necessary business
37   * logic needed to carry out the request</p>
38   *
39   * <p>The getView and getModel methods are used to return the page and data
40   * necessary to present the results of the action</p>
41   */
42
43  public interface Action {
44
45
46      /** Execute business logic */
47      public boolean execute(HttpServletRequest req, HttpServletResponse res)
48        throws ServletException, IOException;
49
50      /** Return the page name (and path) to display the view */
51      public String getView();
52
53      /** Return a JavaBean containing the model (data) */
54      public Object getModel();
55  }
```

Εικόνα 59. Κώδικας για το interface Action

Όπως είπαμε και παραπάνω ο Controller υλοποιείται σαν ένα servlet. Στην Εικόνα 60 βλέπουμε την δήλωση της κλάσης που υλοποιεί τον controller. Βλέπουμε πως η κλάση αυτή κληρονομεί την κλάση HttpServlet. Οι κύριες μεταβλητές της κλάσης είναι η μεταβλητή Factory, η οποία χρησιμοποιείται για να μας παράγει οποιοδήποτε DAO χρειαστεί και θα αναλυθεί περαιτέρω σε επόμενη ενότητα, και η μεταβλητή actions η οποία είναι ένα hash map. Ουσιαστικά περιέχει αντιστοιχίες συμβολοσειρών.

```
1  package controller;
2
3  /*
4   * To change this template, choose Tools | Templates
5   * and open the template in the editor.
6   */
7
8
9  import DAO.DAOFactory;
10 import java.io.IOException;
11 import java.util.HashMap;
12 import javax.servlet.*;
13 import javax.servlet.http.*;
14 /**
15  *
16  * @author ganest
17  */
18 public class Controller extends HttpServlet {
19
20     /** The HashMap events is used to hold the action/event definitions: */
21   public static DAOFactory Factory;
22   protected HashMap actions = new HashMap();
23   public static HashMap path = new HashMap();
```

Εικόνα 60. Οι μεταβλητές της κλάσης του Controller

Οι αντιστοιχίες αυτές αρχικοποιούνται στην συνάρτηση init του servlet, η οποία φαίνεται στην Εικόνα 61. Οι αντιστοιχίες αυτές αντιστοιχίζουν ένα action, που αποτελεί μια λειτουργία της εφαρμογής, με την κλάση η οποία υλοποιεί-εκτελεί αυτό το action. Η πρώτη παράμετρος, το action, περιέχεται στο HttpRequest που φτάνει στον controller. Όπως θα δούμε στην συνέχεια ο controller αντιστοιχίζει το action αυτό στην κλάση που το υλοποιεί.

```
32      public void init() throws ServletException {
33        super.init();
34
35        /**
36         * The Event.properties file is read to determine the actions/events that
37         * can be processed.
38         *
39         * The format of the file is: event/action = handler full class name.
40         * For example:
41         *
42         * loadFirstPage=tuc.ced.music.ecesite.actions.ShowMainAction
43         */
44
45         //insert some actions for testing...
46
47         Factory = DAOFactory.getDAOFactory();
48
49         actions.put("getRestaurantsTEL", "actions.getRestaurantsTEL");
50         actions.put("getRestaurantsPLACE", "actions.getRestaurantsPLACE");
51         actions.put("getRestaurantsCUISINE", "actions.getRestaurantsCUISINE");
52         actions.put("sort_n", "actions.sort_n");
53         actions.put("sort_c", "actions.sort_c");
54         actions.put("sort_p", "actions.sort_p");
55         actions.put("sort_r", "actions.sort_r");
56         actions.put("clear", "actions.clear");
57         actions.put("showRestaurant", "actions.showRestaurant");
58         actions.put("signup", "actions.signup");
59         actions.put("login", "actions.login");
60         actions.put("comment", "actions.comment_act");
61         actions.put("forward", "actions.forward");
62         actions.put("logout", "actions.logout");
63         actions.put("search_map", "actions.search_map");
64         actions.put("deleteComment", "actions.delete_comment");
65         actions.put("updateLocations", "actions.updateLocations");
66         actions.put("rate", "actions.rate");
67         actions.put("retrieve_pass", "actions.retrieve_pass");
68         actions.put("show_profile", "actions.show_profile");
           actions.put("EditProf", "actions.EditProf");
```

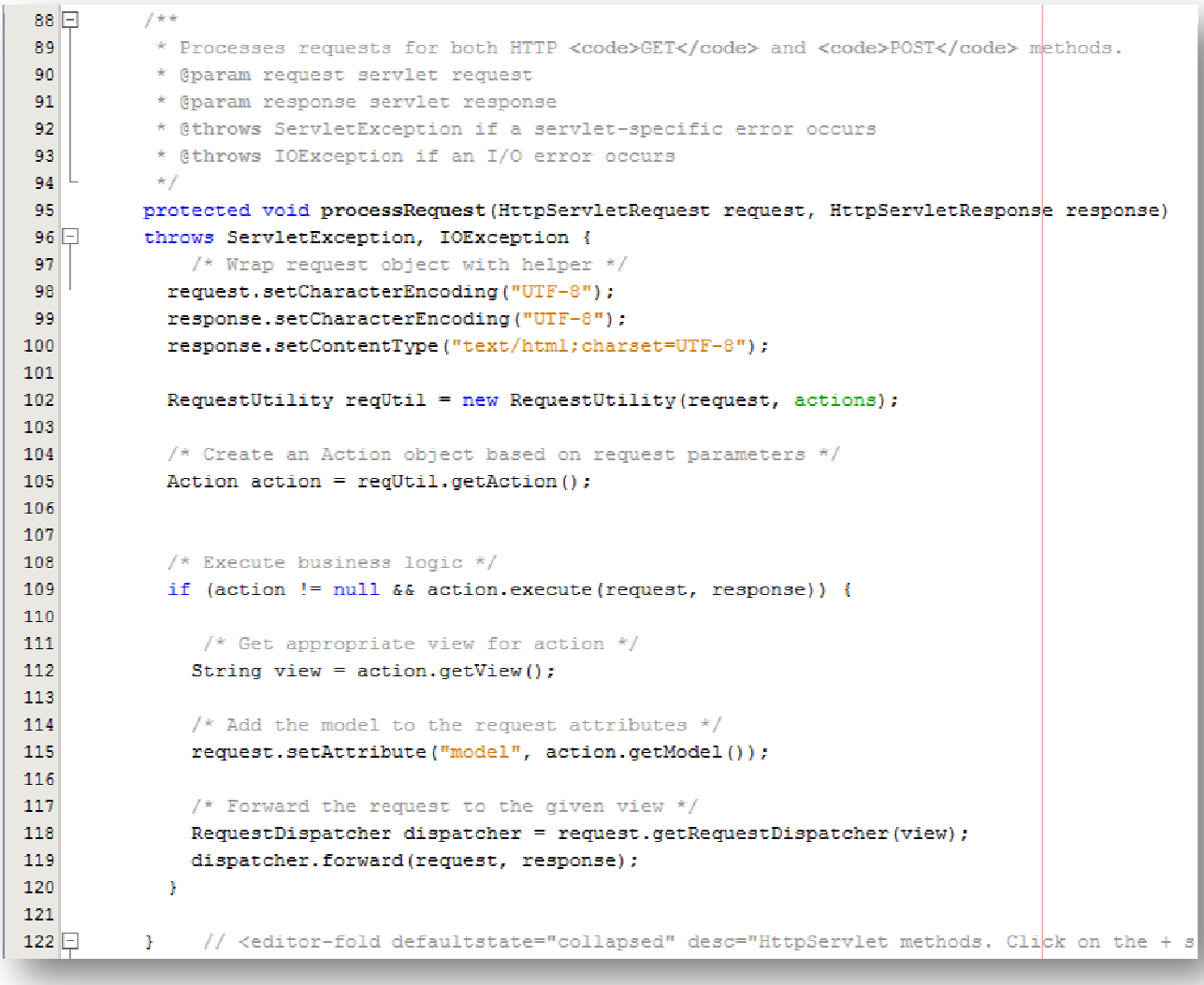
Εικόνα 61. Η συνάρτηση init του servlet

Αφού λοιπόν ο controller αρχικοποιηθεί από την init, μπορεί πλέον να εξυπηρετεί αιτήσεις. Οι συναρτήσεις που εξυπηρετούν τις αιτήσεις σε κάθε servlet είναι οι doGet και doPost, ανάλογα με το αίτημα που φτάνει. Οι υλοποίηση των παραπάνω συναρτήσεων φαίνεται στην Εικόνα 62.

Και οι δύο αυτές συναρτήσεις καλούν την συνάρτηση processRequest η οποία και στις δύο περιπτώσεις εξυπηρετεί το αίτημα. Η υλοποίηση της συνάρτησης αυτής φαίνεται στην Εικόνα 63. Στην Γραμμή 102 βλέπουμε πως δημιουργεί ένα αντικείμενο requestUtility το οποίο ουσιαστικά αντιστοιχίζει το αίτημα με το κατάλληλο αντικείμενο Action που θα το εξυπηρετήσει.

```
    @Override
131 protected void doGet(HttpServletRequest request, HttpServletResponse response)
132 throws ServletException, IOException {
133     processRequest(request, response);
134 }
135
136 /**
137  * Handles the HTTP <code>POST</code> method.
138  * @param request servlet request
139  * @param response servlet response
140  * @throws ServletException if a servlet-specific error occurs
141  * @throws IOException if an I/O error occurs
142  */
    @Override
144 protected void doPost(HttpServletRequest request, HttpServletResponse response)
145 throws ServletException, IOException {
146     processRequest(request, response);
147 }
148
149 /**
150  * Returns a short description of the servlet.
151  * @return a String containing servlet description
152  */
    @Override
154 public String getServletInfo() {
155     return "Short description";
156 }// </editor-fold>
157
158 }
159
```

Εικόνα 62. Οι συναρτήσεις doGet και doPost του controller



```
88      /**
89       * Processes requests for both HTTP <code>GET</code> and <code>POST</code> methods.
90       * @param request servlet request
91       * @param response servlet response
92       * @throws ServletException if a servlet-specific error occurs
93       * @throws IOException if an I/O error occurs
94       */
95      protected void processRequest(HttpServletRequest request, HttpServletResponse response)
96      throws ServletException, IOException {
97          /* Wrap request object with helper */
98        request.setCharacterEncoding("UTF-8");
99        response.setCharacterEncoding("UTF-8");
100       response.setContentType("text/html;charset=UTF-8");
101
102       RequestUtility reqUtil = new RequestUtility(request, actions);
103
104       /* Create an Action object based on request parameters */
105       Action action = reqUtil.getAction();
106
107
108       /* Execute business logic */
109       if (action != null && action.execute(request, response)) {
110
111          /* Get appropriate view for action */
112         String view = action.getView();
113
114         /* Add the model to the request attributes */
115         request.setAttribute("model", action.getModel());
116
117         /* Forward the request to the given view */
118         RequestDispatcher dispatcher = request.getRequestDispatcher(view);
119         dispatcher.forward(request, response);
120       }
121
122     }    // <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the + s
```

Εικόνα 63. Η συνάρτηση processRequest η οποία εξυπηρετεί τις αιτήσεις GET και POST

Η υλοποίηση της κλάσης RequestUtility φαίνεται στην Εικόνα 64. Η κλάση περιέχει δύο μεταβλητές. Η πρώτη είναι ένα αντικείμενο HttpServletRequest και η δεύτερη ένα hashMap. Και οι δύο μεταβλητές αρχικοποιούνται από τον constructor κατά την δημιουργία του αντικειμένου. Όπως βλέπουμε στην Γραμμή 102 της Εικόνα 63, στην πρώτη μεταβλητή αντιστοιχίζεται το request που μας έχει σταλεί από τον client του χρήστη και στην δεύτερη το hashMap με τις αντιστοιχίες ενεργειών-κλάσεων που έχει ο controller. Μετά την δημιουργία του αντικειμένου καλείται η συνάρτηση getAction. Η συνάρτηση αυτή, όπως βλέπουμε στην Γραμμή 58 της Εικόνα 64, ανακτά την παράμετρο action από το request και στη Γραμμή 62 διαβάζει από το hashMap σε ποια κλάση αντιστοιχεί. Στη συνέχεια στις γραμμές 65 ή 72 ανάλογα, αρχικοποιεί και επιστρέφει το αντικείμενο action, μέσω της συνάρτησης createAction της κλάσης ActionFactory. Η κλάση αυτή φαίνεται στην Εικόνα 65. Η συνάρτηση createAction στην Γραμμή 50, ανακτά το class αρχείο του οποίου το όνομα είναι αυτό που πήραμε από το HashMap και στη συνέχεια, στην Γραμμή 51, δημιουργεί ένα thread που εκτελεί αυτή την κλάση. Ουσιαστικά η κλάση αυτή είναι εκείνη που κάνε την κύρια επεξεργασία του αιτήματος.

Έτσι στην Γραμμή 105 της Εικόνα 63 επιστρέφεται ένα action του οποίου η συνάρτηση execute, η οποία κάνει όλη την επεξεργασία που έχει ορίσει ο προγραμματιστής πως πρέπει να γίνει για να εξυπηρετηθεί το αίτημα, εκτελείται μέσα στο if στην Γραμμή 109.

Όλα τα actions περιέχουν δύο μεταβλητές. Η μία είναι ένα java bean που χρησιμοποιείται για να περάσει δεδομένα επεξεργασίας στο request και ονομάζεται model, και η άλλη είναι μία συμβολοσειρά με το όνομα view, που υποδεικνύει το αρχείο jsp που πρέπει να προβληθεί στον client ώστε να απαντηθεί το αίτημα. Μετά την εκτέλεση της execute οι δύο αυτές μεταβλητές έχουν πάρει τις σωστές τιμές. Στην Γραμμή 115 περνάει η τιμή του model στο request ώστε να είναι διαθέσιμη στο jsp στο οποίο στις Γραμμές 118 και 119 περνάει ο έλεγχος. Το jsp αυτό υποδεικνύεται από το view και παράγει το τελικό html αρχείο το οποίο θα αποσταλεί στον client που έκανε το αίτημα.

```
39  public class RequestUtility {
40
41      /** Local copy of request object. */
42      HttpServletRequest request;
43      HashMap actions;
44      /**
45       * Constructor. Used to set local request object.
46       */
47      public RequestUtility(HttpServletRequest req, HashMap actions)
48                          throws ServletException, IOException {
49         this.request = req;
50         this.actions = actions;
51      }
52
53      /**
54       * Use factory to create action based on request parms
55       */
56      public Action getAction() {
57
58         String action = (String)request.getParameter("action");
59
60         if (action != null)
61         {
62           String actionClass = (String)actions.get(action);
63
64           if ( actionClass != null)
65             return ActionFactory.createAction(actionClass);
66         }
67         else
68         {
69           String actionClass = (String)actions.get("loadFirstPage");
70
71           if ( actionClass != null)
72             return ActionFactory.createAction(actionClass);
73
74         }
75         return null;
76      }
```

Εικόνα 64. Η κλάση RequestUtility

```
38  public abstract class ActionFactory {
39
40      /**
41       * Instantiate and return the appropriate
42       * Action object
43       */
44      public static Action createAction(String actionClassName) {
45
46          Class actionObj = null;
47          Action action = null;
48
49          try {
50              Class c = getClass(actionClassName);
51              action = (Action) c.newInstance();
52          }
53           catch(ClassNotFoundException cnf) {
54             System.out.println( "Couldn't find class " + actionClassName);
55           }
56           catch(InstantiationException ie) {
57             System.out.println( "Couldn't instantiate an object of type " + actionClassName);
58           }
59           catch(IllegalAccessException ia) {
60             System.out.println("Couldn't access class " + actionClassName);
61           }
62           return action;
63      }
64
65
66      private static Class getClass(String classname)
67          throws ClassNotFoundException {
68
69          ClassLoader classLoader = Thread.currentThread().getContextClassLoader();
70
71          if(classLoader == null)
72              classLoader = ActionFactory.class.getClassLoader();
73
74          return (classLoader.loadClass(classname));
75      }
```

Εικόνα 65. Η κλάση ActionFactory

## 5.3. Data Access Objects – Java Beans

Για την επικοινωνία της εφαρμογής με την βάση δεδομένων επιλέξαμε να φτιάξουμε ένα ενδιάμεσο επίπεδο μεταξύ της εφαρμογής και της βάσης, γνωστό στην βιβλιογραφία ως Data Access Objects (DAO). Η χρήση αυτού του ενδιάμεσου επιπέδου κρίνεται επιβεβλημένη γιατί δίνει την δυνατότητα η εφαρμογή να είναι ανεξάρτητη από το σύστημα διαχείρισης της βάσης δεδομένων που χρησιμοποιείται. Σε περίπτωση που χρειαστούν αλλαγές στο σύστημα της βάσης αρκεί απλά να αλλάξουμε το επίπεδο του DAO.

Όπως βλέπουμε στην Εικόνα 60, ο controller έχει μια static μεταβλητή Factory, τύπου DAOFactory. Η κλάση αυτή φαίνεται στην Εικόνα 66. Ουσιαστικά είναι ένα interface με συναρτήσεις που μας επιστρέφουν όλα τα DAO's που είναι διαθέσιμα για την βάση της εφαρμογής. Επιλέξαμε να κάνουμε ένα DAO για κάθε πίνακα που διαθέτει η βάση. Για κάθε ένα από αυτά δημιουργήσαμε ένα interface, που περιέχει όλες τις συναρτήσεις που χρειάζεται η εφαρμογή. Ενδεικτικά, στην Εικόνα 67 βλέπουμε το interface για το restaurantDAO. Αντίστοιχα interface έχουμε κάνει για όλα τα DAO. Η λογική πίσω από την δημιουργία των interfaces είναι ο ορισμός όλων των συναρτήσεων που χρειάζεται η εφαρμογή για να λειτουργήσει. Η υλοποίηση των συναρτήσεων διαφέρει ανάλογα με το σύστημα της βάσης δεδομένων που χρησιμοποιείται. Στην περίπτωσή μας χρησιμοποιήσαμε MySQL, έτσι υλοποιήσαμε το παραπάνω interface ώστε να δουλεύει με MySQL. Αν χρειαστεί να αλλάξει το σύστημα της βάσης, αρκεί απλά να γίνει υλοποίηση του interface για το νέο σύστημα, χωρίς να γίνει οποιαδήποτε άλλη αλλαγή στην εφαρμογή.

```
6   package DAO;
7
8   /**
9    *
10   * @author balantis7
11   */
12  public abstract class DAOFactory
13  {
14      public abstract restaurantDAO getRestaurantDAO();
15      public abstract userDAO getUserDAO();
16      public abstract commentDAO getCommentDAO();
17      public abstract pmDAO getPMDAO();
18
19      public static DAOFactory getDAOFactory()
20      {
21
22            return new mysqlDAOFactory();
23
24      }
25
26  }
27
```

Εικόνα 66. Η abstract κλάση DAOFactory

```java
package DAO;
import java.util.Vector;
import model.restaurant_data;

/**
 *
 * @author balantis7
 */
public interface restaurantDAO {
    public boolean setRestaurant(restaurant_data restaurant);
    public restaurant_data getRestaurantTEL(String telephone);
    public Vector getRestaurantPLACE(String place);
    public Vector getRestaurantCUISINE(String place);
    public Vector getAllRestaurants();
    public boolean deleteRestaurant(String telephone);
    public boolean UpdateGeo(String telephone, float lat, float log);
    public boolean UpdateTelephone(String telephone, String newphone);
    public boolean UpdateName(String telephone, String name);
    public boolean UpdateAddress(String telephone, String address);
    public boolean UpdatePlace(String telephone, String place);
    public boolean UpdateCuisine(String telephone, String cuisine);
    public boolean UpdateRate(String telephone, int rate, int num_rates);
    public Vector getSearchPlaces(String cuisine,String place);
    public Vector getSearchRates(String cuisine,String place);
    public Vector getCuisines();
    public Vector getPlaces();

}
```

Εικόνα 67. Το interface του restaurantDAO

Για την σύνδεση στην βάση χρησιμοποιήσαμε JDBC. Η σύνδεση δημιουργείται από την συνάρτηση getDAOFactory στην Εικόνα 66. Η υλοποίηση της συνάρτησης δεν γίνεται απευθείας σε αυτή την κλάση για τους ίδιους λόγους που περιγράψαμε παραπάνω. Στην Εικόνα 68 βλέπουμε την κλάση η οποία κάνει την σύνδεση στην βάση με χρήση JDBC και μας επιστρέφει το DAO που επιθυμούμε.

```
15  public class mysqlDAOFactory extends DAOFactory {
16      public static final String DRIVER= "com.mysql.jdbc.Driver";
17      public static final String DBURL= "jdbc:mysql://pan.softnet.tuc.gr:3306/restaurants_page";
18      public static final String username="[redacted]";
19      public static final String password="[redacted]";
20
21      public static Connection createConnection()
22      {
23          Connection conn=null;
24          try {
25              // Use DRIVER and DBURL to create a connection // Recommend connection pool implementation/usage
26              Class.forName(DRIVER);
27              try {
28
29                  conn = DriverManager.getConnection(DBURL, username,password);
30              } catch (SQLException ex) {
31                  Logger.getLogger(mysqlDAOFactory.class.getName()).log(Level.SEVERE, null, ex);
32              }
33          } catch (ClassNotFoundException ex) {
34              Logger.getLogger(mysqlDAOFactory.class.getName()).log(Level.SEVERE, null, ex);
35          }
36          return conn;
37      }
38
39
40      public restaurantDAO getRestaurantDAO(){
41          return new mysqlRestaurantDAO();
42      }
43      public userDAO getUserDAO()
44      {
45          return new mysqlUserDAO();
46      }
47      public commentDAO getCommentDAO()
48      {
49          return new mysqlCommentDAO();
50      }
51      public pmDAO getPMDAO()
52      {
```

Εικόνα 68. Η κλάση mysqlDAOFactory που υλοποιεί το interface που ορίζει η κλάση DAOFactory, για συνεργασία με εξυπηρετητή MySQL

Για την μεταφορά των δεδομένων από την βάση στην εφαρμογή και την αποθήκευσή τους στην μνήμη ώστε να είναι διαθέσιμα για επεξεργασία ή προβολή από τα JSP, χρησιμοποιήσαμε Java Beans [37]. Τα Java Beans είναι αντικείμενα που περιέχουν μια συλλογή διαφόρων αντικειμένων, τα οποία είναι προσβάσιμα για ανάγνωση ή εγγραφή μέσω ενός συνόλου συναρτήσεων. Οι συναρτήσεις που είναι για ανάγνωση ονομάζονται getters και αυτές που είναι για εγγραφή setters. Στην Εικόνα 69 βλέπουμε ενδεικτικά ένα μέρος του κώδικα για το java bean που χρησιμοποιείται για την αναπαράσταση των δεδομένων ενός εστιατορίου.

Έτσι κάθε DAO διαβάζει από την βάση τα δεδομένα που ζητούνται, και επιστρέφει ένα διάνυσμα από java beans στα οποία έχει φορτώσει τα δεδομένα που διάβασε. Αν πάλι χρειάζεται να εγγραφούν δεδομένα στην βάση, τα δεδομένα αυτά περνούν στο ανάλογο DAO σαν ένα διάνυσμα από java beans τα οποία περιέχουν τα δεδομένα προς εγγραφή. Στην Εικόνα 70, βλέπουμε ενδεικτικά την συνάρτηση getRestaurantPLACE του restaurantDAO η οποία μας

επιστέφει ένα Vector από java beans σαν αυτό της Εικόνα 69, τα οποία περιέχουν τις εγγραφές οι οποίες βρίσκονται στην περιοχή που ορίζει η παράμετρος place.

```
12  public class restaurant_data extends Object {
13      float latitude;
14      float longitude;
15      String address;
16      String title;
17      String phone;
18      String place;
19      String cuisine;
20      int rate;
21      int rate_num;
22
23      public restaurant_data()
24      {
25          latitude=0;
26          longitude=0;
27          address="null";
28          title="null";
29          phone="0";
30          place="null";
31          cuisine="null";
32          rate=0;
33          rate_num=0;
34      }
35      public void setTitle(String t)
36      {
37          title=t;
38      }
39      public void setAddress(String a)
40      {
41          address=a;
42      }
43      public void setLatitude(float l)
44      {
45          latitude=l;
46      }
```

Εικόνα 69. Java Bean για την αναπαράσταση των δεδομένων της βάσης για τα εστιατόρια

```
74      public Vector getRestaurantPLACE(String place)
75      {
76          Vector restaurants=new Vector();
77
78          Connection conn=new mysqlDAOFactory().createConnection();
79          try
80          {
81              Statement st = conn.createStatement();
82              ResultSet prod=st.executeQuery("SELECT * FROM restaurants WHERE place LIKE '%"
83                      +place+"%' or address LIKE '%"+place+"%' order by rate desc");
84              while(prod.next())
85              {
86                 restaurant_data restaurant = new restaurant_data();
87                 restaurant.setTitle(prod.getString(1));
88                 restaurant.setAddress(prod.getString(2));
89                 restaurant.setPhone(prod.getString(3));
90                 restaurant.setLatitude(prod.getFloat(4));
91                 restaurant.setLongitude(prod.getFloat(5));
92                 restaurant.setPlace(prod.getString(6));
93                 restaurant.setCuisine(prod.getString(7));
94                 restaurant.setRate(prod.getInt(8));
95                 restaurant.setRate_num(prod.getInt(9));
96                 restaurants.add(restaurant);
97
98              }
99              st.close();
100             return restaurants;
101         }
102         catch (SQLException e)
103         {
104             e.printStackTrace();
105             return null;
106         }
107     }
```

Εικόνα 70. Μια συνάρτηση του mysqlRestaurantDAO η οποία επιστρέφει όλες τις εγγραφές που βρίσκονται στην περιοχή που ορίζει η παράμετρος place

## 5.4. Υλοποίηση των ενεργειών (actions)

Είδαμε στην Ενότητα «5.2. Η αρχιτεκτονική που χρησιμοποιήθηκε» πως σε κάθε αίτημα προς τον server υπήρχε μια παράμετρος με το όνομα action και ανάλογα με αυτή την παράμετρο ο controller επέλεγε την κατάλληλη κλάση που θα χειριστεί το αίτημα, και δημιουργούσε ένα αντικείμενο Action από την κλάση αυτή.

Έτσι για κάθε ενέργεια που θέλουμε να υποστηρίζει η εφαρμογή, έπρεπε να φτιάξουμε μια υλοποίηση του interface Action που βλέπουμε στην Εικόνα 59. Οι συναρτήσεις getView και getModel είναι απλές και επιστρέφουν το jsp που θέλουμε να αναλάβει τον έλεγχο η πρώτη, και τα java beans που χρησιμοποιήθηκαν η δεύτερη. Η συνάρτηση που αναλαμβάνει να κάνει τις απαραίτητες ενέργειες στην βάση μέσω κάποιου DAO, και να επεξεργαστεί τα δεδομένα που είτε θα γραφτούν είτε αναγνώστηκαν από τα DAO, είναι η execute.

Ενδεικτικά, στην Εικόνα 71 βλέπουμε την υλοποίηση της συνάρτησης execute για την λειτουργία του login. Σε αυτή την περίπτωση έχει υποβληθεί μία φόρμα από τον χρήστη με το όνομα χρήστη και τον κωδικό του. Στις Γραμμές 32 και 33 και 34 διαβάζονται αυτά τα στοιχεία τα οποία περιέχονται στο αίτημα (req) που εστάλει από τον client. Στις Γραμμές 36 και 37 κρυπτογραφείται ο κωδικός που μας δόθηκε, αφού στην βάση οι κωδικοί είναι κρυπτογραφημένοι για λόγους ασφαλείας, και στην Γραμμή 40 ανακτάται από την βάση με χρήση του αντίστοιχου DAO η εγγραφή χρήστη με τα δεδομένα στοιχεία. Στη Γραμμή 41 ελέγχεται αν επιστράφηκε κάποια εγγραφή από την βάση, άρα τα στοιχεία που μας έδωσε ο χρήστης είναι σωστά, και στην Γραμμή 43 εισάγει το java bean με την εγγραφή του χρήστη στο session, στο attribute “login”, το οποίο όταν υπάρχει υποδηλώνει πως ο χρήστης είναι συνδεδεμένος. Στις Γραμμές 44-55 γίνεται πάλι πρόσβαση στην βάση για να ελέγξουμε αν υπάρχουν νέα μηνύματα για τον λογαριασμό χρήστη και αν θέσουμε το ανάλογο attribute στο session. Αν το αντικείμενο που μας επέστρεψε η βάση είναι κενό, όπως βλέπουμε στην Γραμμή 61, βάζουμε ένα attribute στο session που υποδηλώνει πως τα στοιχεία που έδωσε ο χρήστης είναι λάθος. Τέλος στην Γραμμή 64 δηλώνουμε το jsp που θα αναλάβει να δημιουργήσει το αρχείο html που θα αποσταλεί στον χρήστη ως απάντηση στο αίτημά του.

```
30      public boolean execute(HttpServletRequest req, HttpServletResponse res)
31              throws ServletException, IOException {
32        u=new user();
33        String action = req.getParameter("action");
34       String username=req.getParameter("username");
35       String pass=req.getParameter("password");
36       //validate encrypted pass
37       encrypt Encrypt=new encrypt();
38       String passw=Encrypt.Doencrypt(pass);
39
40        userDAO d =  Controller.Factory.getUserDAO();
41        u=d.getUser(username, passw);
42        if(!u.getUsername().equals(""))
43        {
44            req.getSession().setAttribute("login", u);
45            pmDAO pm =  Controller.Factory.getPMDAO();
46            Vector messages,unread;
47            messages=pm.getUserMessages(u.getUsername());
48            Message m;
49            unread=new Vector();
50            for(int i=0;i<messages.size();i++)
51            {
52                m=(Message)messages.elementAt(i);
53                if(m.getRead()==0)
54                {
55                   unread.add(m);
56                }
57            }
58            req.getSession().setAttribute("unread_messages", unread.size());
59        }
60        else
61        {
62            req.getSession().setAttribute("user_not_found", "user not found");
63        }
64
65          view = "./login.jsp";
66        return true;
      }
```

Εικόνα 71. Υλοποίηση της συνάρτησης execute για την ενέργεια login

## 5.5. Προβολή των δεδομένων στον χρήστη μέσω των jsp

Αφού περαστούν τα απαραίτητα δεδομένα από τον server στο session, το αρχείο jsp που έχει επιλεγεί αναλαμβάνει να οπτικοποιήσει τα δεδομένα μέσω κώδικα html και javascript ώστε να τα δει ο χρήστης μέσα από τον browser του. Όπως είδαμε στην ενότητα «3.2. Java Server Pages (JSP)», τα αρχεία jsp αποτελούνται από κώδικα html και javascript στον οποίο έχει ενσωματωθεί και κώδικας java μέσα σε ειδικά tags. Ο server πριν προωθήσει το jsp στον client τρέχει τα κομμάτια κώδικα java. Ότι βρίσκεται εκτός των tags της java απλά αντιγράφεται στο αρχείο εξόδου και αγνοείται. Το τελικό αποτέλεσμα είναι ένα αρχείο που περιέχει μόνο html και javascript. Κάθε φορά, ανάλογα με τα δεδομένα που έχουν περαστεί στο session ή σε άλλες μεταβλητές από τα actions, προκύπτει διαφορετικός κώδικας html ως τελικό αποτέλεσμα. Άρα από ένα αρχείο jsp μπορούμε να παράξουμε πολύ διαφορετικά αποτελέσματα ανάλογα με τις παραμέτρους που έχουν φορτωθεί πιο πριν.

Ενδεικτικά στην Εικόνα 72 βλέπουμε το κομμάτι του κώδικα που παράγει τα σχόλια στην σελίδα του εστιατορίου. Την σελίδα αυτή την βλέπουμε στην Εικόνα 32. Ο κώδικας που είναι σε γαλάζιο φόντο είναι κώδικας java μέσα σε tags. Τα κομμάτια αυτά εκτελούνται στον server πριν την αποστολή του jsp στον client. Στην Γραμμή 337 ελέγχεται αν περιέχει το session το attribute comments το οποίο περιέχει τα σχόλια που θέλουμε να εμφανίζουμε. Στην γραμμή 339 διαβάζουμε από το session τα σχόλια και τα περνάμε σε μία μεταβλητή java. Στην συνέχεια κλείνει το tag της java και πάμε σε κώδικα html. Τα αποτελέσματα θα εμφανιστούν μέσα σε ένα πίνακα, με δύο στήλες. Θέλουμε για λόγους ευκρίνειας οι γραμμές του πίνακα να έχουν εναλλάξ διαφορετικό χρώμα φόντου. Στην Γραμμή 341 δηλώνεται ο πίνακας. Στη συνέχεια ανοίγουμε μία δομή επανάληψης με κώδικα java ώστε να δημιουργήσουμε τις γραμμές του πίνακα. Σε κάθε γραμμή θα περιέχεται ένα σχόλιο. Επειδή θέλουμε να έχουμε διαφορετικά χρώματα φόντου εναλλάξ στις γραμμές του πίνακα, έχουμε τις Γραμμές 348-351 και 363-366 που δημιουργούν μια γραμμή πίνακα αλλά με διαφορετικό φόντο. Για να επιτύχουμε την εναλλαγή ελέγχουμε κάθε φορά αν ο αριθμός της επανάληψης είναι άρτιος ή περιττός. Αν είναι άρτιος θα παραχθεί ο κώδικας των Γραμμών 348-353, αλλιώς αυτός των Γραμμών 366-371. Κάθε φορά ελέγχουμε αν υπάρχει συνδεδεμένος χρήστης (Γραμμές 354 και 372) και αν ο χρήστης αυτός είναι ο ίδιος με αυτόν που έχει γράψει το σχόλιο που επεξεργαζόμαστε εκείνη την στιγμή (Γραμμές 357 και 375). Αν ισχύουν και τα δύο δημιουργούμε μια νέα στήλη με το κουμπί διαγραφή.

Τέλος να αναφέρουμε πως περνάμε παραμέτρους προς τον controller. Σε κάθε ενέργεια είναι απαραίτητο να περάσουμε την παράμετρο action ώστε να ξέρει ο controller πώς να χειριστεί το αίτημα. Στην Γραμμή 349 της Εικόνα 72 βλέπουμε τον κώδικα για την δημιουργία συνδέσμου που μας προβάλλει το προφίλ χρήστη. Σε κάθε σύνδεσμο έχουμε το όνομα του servlet του controller (στην περίπτωσή μας Controller) και στην συνέχεια ερωτηματικό και την λίστα με τις παραμέτρους χωρισμένες με &. Η παράμετρος που περνάμε πάντα είναι η παράμετρος action. Στην συνέχεια μπορούμε να έχουμε οποιεσδήποτε άλλες παραμέτρους. Στο παράδειγμά μας περνάμε την παράμετρο action με τιμή “show_profile” και την παράμετρο user η οποία παίρνει τιμή από την συνάρτηση sx.getUserID().

```
337   if(session.getAttribute("comments")!=null)
338       {
339          comments=(Vector)session.getAttribute("comments");
340       }%>
341   <table>
342  <%
343       for(int i=0;i<comments.size(); i++)
344       {
345       sx=(comment)comments.elementAt(i);%>
346       <% if(i%2==0)
347           {%>
348       <tr bgcolor=#BFBBBB> <td>
349               <p><font size="2">Ο χρήστης <a href="Controller?action=show_profile&user=<%=sx.getUserID()%>"
350                                             id="designanchor<%=i+1%>"><b><%=sx.getUserID()%></b></a>
351                                             στις <%=sx.getDate()%> <%=sx.getTime()%> έγραψε:</font>
352               <p><font size="3"><%=sx.getComment()%></font>
353           </td>
354           <%if(session.getAttribute("login")!=null)
355           {
356               user u=(user)session.getAttribute("login");
357               if(u.getUsername().equals(sx.getUserID()) || u.getSuperuser()==1)
358                   {%>
359                   <td> <a href="Controller?action=deleteComment&comment_id=<%=sx.getCommentID()%>">
360                           <img src="icons/pmdelete.png" alt="log out" width="50" height="18" BORDER=0 /></a></td>
361       <%          }
362                }
363        }
364       else
365       {%>
366           <tr bgcolor=#FCFCFC> <td>
367                   <p><font size="2">Ο χρήστης <a href="Controller?action=show_profile&user=<%=sx.getUserID()%>"
368                                                 id="designanchor<%=i+1%>"><b><%=sx.getUserID()%></b></a>
369                                                 στις <%=sx.getDate()%> <%=sx.getTime()%> έγραψε:</font>
370                   <p><font size="3"><%=sx.getComment()%></font>
371           </td>
372           <%if(session.getAttribute("login")!=null)
373           {
374               user u=(user)session.getAttribute("login");
375               if(u.getUsername().equals(sx.getUserID()) || u.getSuperuser()==1)
376                   {%>
377                   <td> <a href="Controller?action=deleteComment&comment_id=<%=sx.getCommentID()%>">
378                          <img src="icons/pmdelete.png" alt="log out" width="50" height="18" BORDER=0 /></a></td>
379       <%          }
380                }
381     }%>
382      </tr>
383  <%  }%>
384    </table>
```

Εικόνα 72. Ο κώδικας του jsp που παράγει τα σχόλια στην σελίδα εστιατορίου

# 6. Σύγκριση της Εφαρμογής μας με παρόμοιες υπάρχουσες εφαρμογές

## 6.1. Εφαρμογές παρόμοιες με την δική μας

Στο διαδίκτυο υπάρχουν αρκετές εφαρμογές παρόμοιες με την εφαρμογή που δημιουργήσαμε. Παρακάτω θα παρουσιάσουμε συνοπτικά τις πιο γνωστές από αυτές καθώς και τις κυριότερες δυνατότητες που παρέχουν.

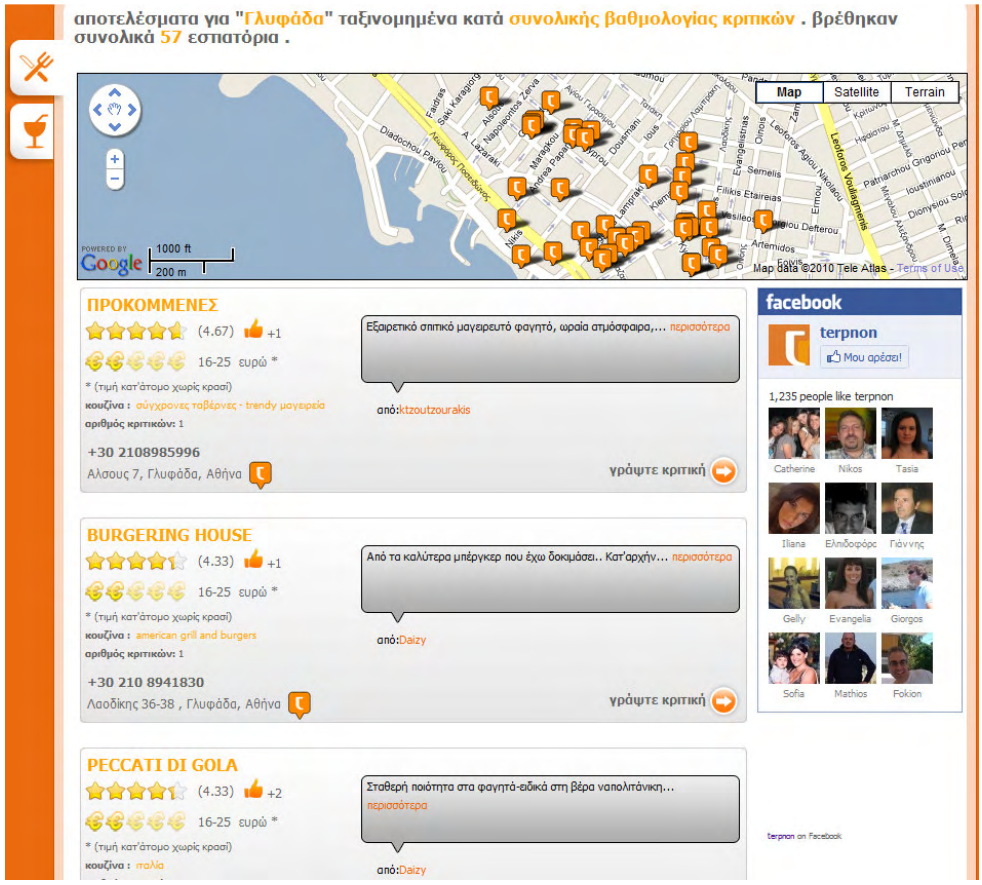
### 6.1.1 Εφαρμογή terpnon

Η εφαρμογή αυτή είναι διαθέσιμη στην διεύθυνση www.terpnon.com. Έχει την δυνατότητα αναζήτησης με βάση την περιοχή, το όνομα του εστιατορίου και την βαθμολογία που έχει πάρει. Τα αποτελέσματα της αναζήτησης, όπως βλέπουμε στην Εικόνα 73, είναι ταξινομημένα με βάση την βαθμολογία που έχουν πάρει και δεν δίνεται άλλος διαθέσιμος τρόπος ταξινόμησης. Επίσης τα αποτελέσματα εμφανίζονται και στον χάρτη, από τον οποίο μπορούμε να πάρουμε πληροφορία κάνοντας κλικ στο marker που μας ενδιαφέρει. Στην

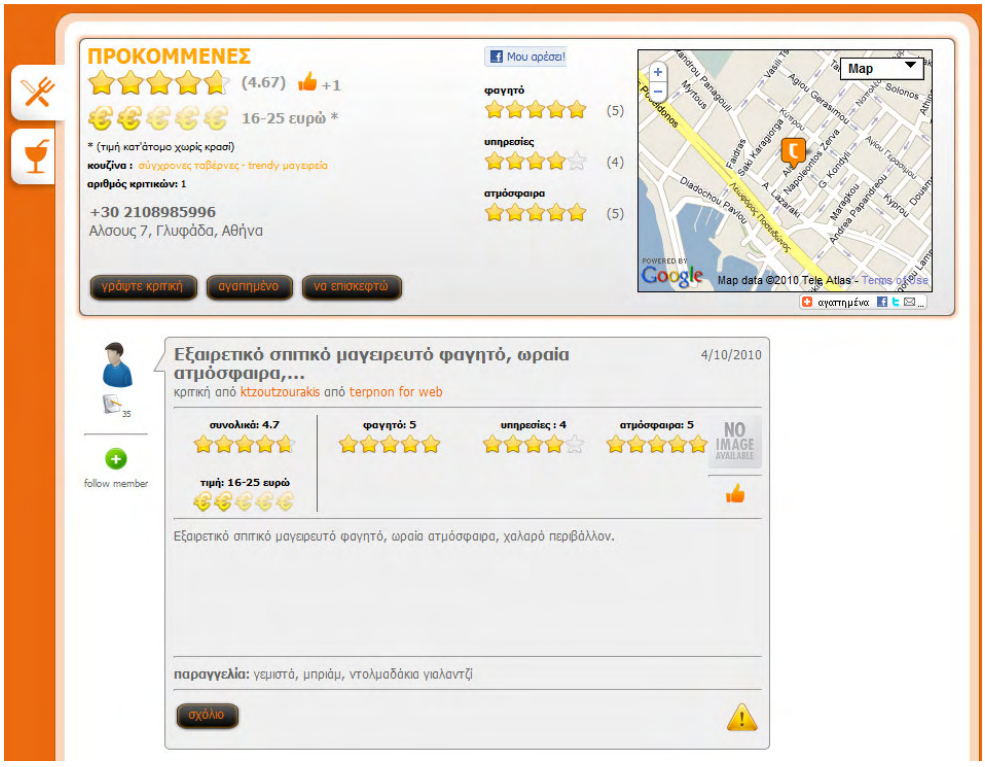
Εικόνα 74 βλέπουμε την σελίδα εστιατορίου της εφαρμογής. Οι εγγεγραμμένοι χρήστες έχουν την δυνατότητα να γράψουν σχόλιο, να βαθμολογήσουν το εστιατόριο ανά κατηγορία παροχής υπηρεσίας που προσφέρει και να το προσθέσουν στα αγαπημένα τους. Γενικά στην συγκεκριμένη εφαρμογή δίνεται έμφαση στις κριτικές των χρηστών και όχι τόσο στην γεωγραφική πληροφορία.

### 6.1.2. Εφαρμογή Ταβερνοχώρος

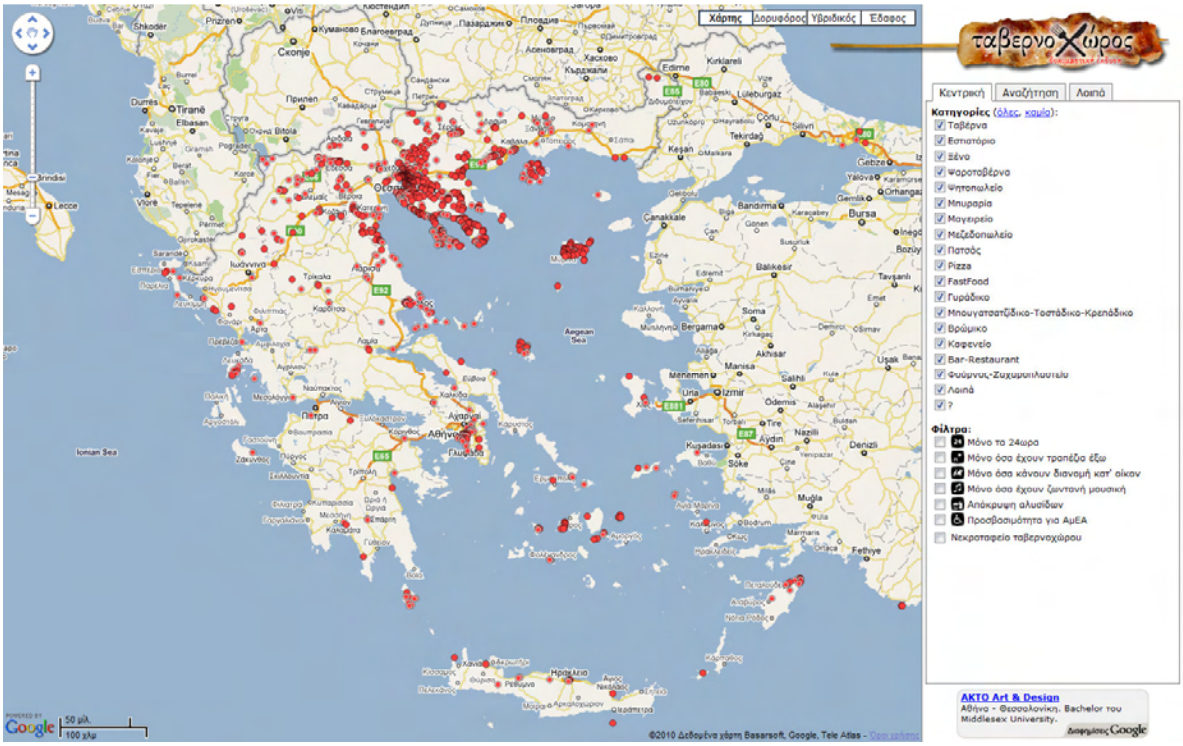
Η εφαρμογή αυτή είναι διαθέσιμη στην διεύθυνση www.tavernoxoros.gr/ και έχει διαφορετική φιλοσοφία από την προηγούμενη. Εστιάζει αποκλειστικά στον χάρτη και σου βγάζει με το που ανοίγεις την σελίδα όλα τα διαθέσιμα εστιατόρια φορτωμένα στον χάρτη ο οποίος καταλαμβάνει το μεγαλύτερο μέρος της σελίδας, όπως βλέπουμε και στην Εικόνα 75. Μπορείς να φιλτράρεις τα αποτελέσματα όπως φιλτράρεις τις πληροφορίες που εμφανίζονται στον χάρτη στην εφαρμογή Google Earth. Διαθέτει και αναζήτηση η οποία δεν είναι τόσο εύχρηστη όσο σε άλλες εφαρμογές καθώς τα αποτελέσματα εμφανίζονται στην μπάρα δεξιά, δεν υπάρχει καμία δυνατότητα φιλτραρίσματος ή ταξινόμησής τους και επίσης δεν ανανεώνονται οι πληροφορίες του χάρτη, όπως βλέπουμε στην Εικόνα 76. Υπάρχει η δυνατότητα σχολιασμού για κάθε εστιατόριο χωρίς να χρειάζεται να εγγραφείς στην σελίδα. Τέλος δεν διαθέτει σύστημα βαθμολόγησης.



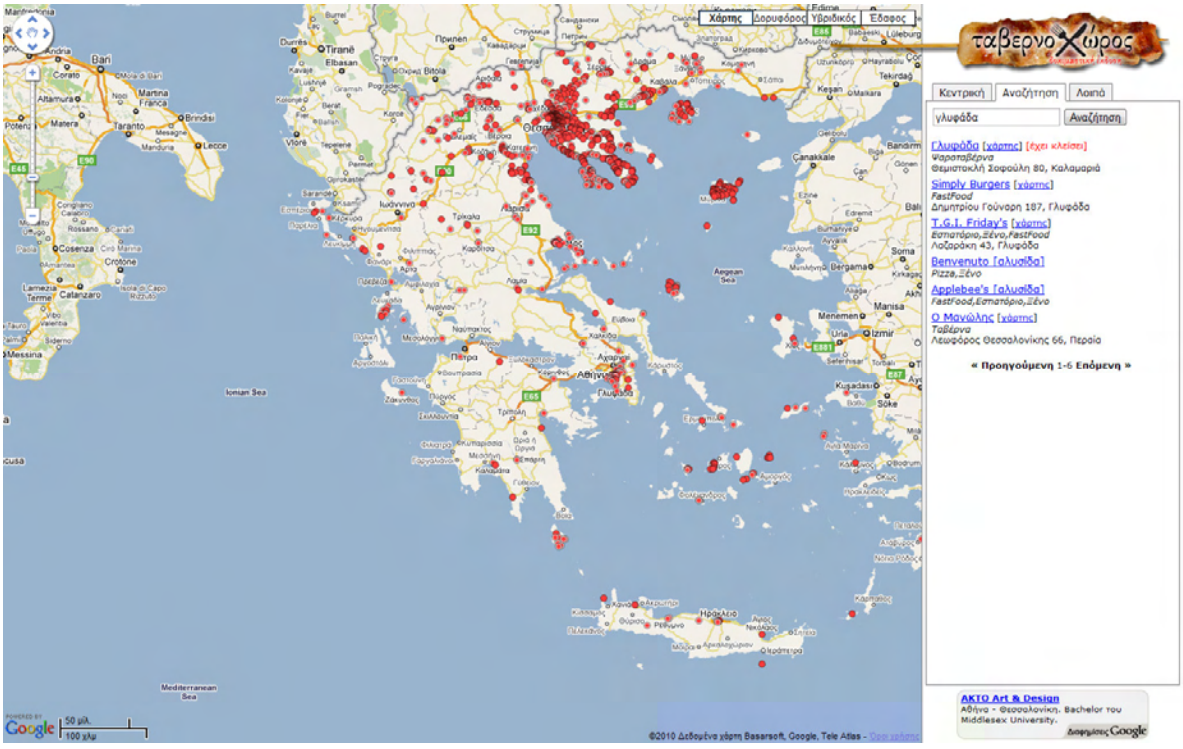

Εικόνα 73. Αποτελέσματα αναζήτησης στην εφαρμογή terpnon




Εικόνα 74. Σελίδα εστιατορίου στην εφαρμογή terpnon



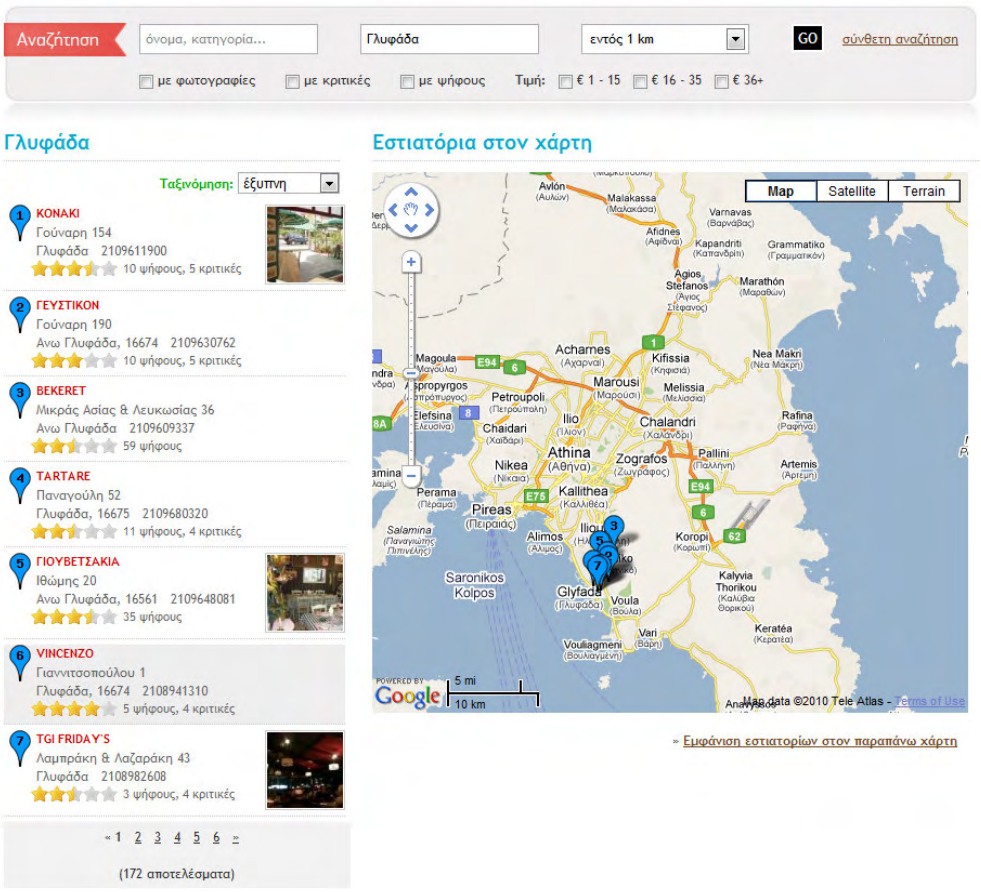

Εικόνα 75. Η εφαρμογή ταβερνοχώρος




Εικόνα 76. Αποτελέσματα αναζήτησης στην εφαρμογή ταβερνοχώρος

### 6.1.3. Η εφαρμογή gong

Η εφαρμογή αυτή είναι διαθέσιμη στην διεύθυνση www.gong.gr. Διαθέτει δυνατότητες αναζήτησης με βάση την περιοχή και με βάση το όνομα εστιατορίου. Επίσης δύνονται επιλογές φιλτραρίσματος ώστε να εμφανίζονται μόνο αποτελέσματα που έχουν κριτική ή φωτογραφία και που έχουν συγκεκριμένο μέσο κόστος κατά άτομο. Τα αποτελέσματα της αναζήτησης, όπως βλέπουμε και στην Εικόνα 77, εμφανίζονται ταυτόχρονα στον χάρτη και σε λίστα. Επίσης υπάρχει η δυνατότητα ταξινόμησης με βάση την βαθμολογία που έχουν πάρει, και μια ακόμα επιλογή ταξινόμησης που την ονομάζουν έξυπνη ταξινόμηση. Οι εγγεγραμμένοι χρήστες έχουν δυνατότητα σχολιασμού βαθμολόγησης και αποστολής φωτογραφίας του εστιατορίου.




Εικόνα 77. Αποτελέσματα αναζήτησης στην εφαρμογή gong

## 6.2. Πλεονεκτήματα της εφαρμογής μας σε σχέση με τις υπάρχουσες εφαρμογές

Η βασική καινοτομία που εισάγαμε στην εφαρμογή μας σε σχέση με τις παραπάνω εφαρμογές ήταν η χρήση του χάρτη ως μέσου ορισμού της περιοχής αναζήτησης από τον χρήστη. Έτσι ο χρήστης μπορεί να ορίσει με πιο φιλικό για αυτόν τρόπο την περιοχή στην οποία θέλει να βρίσκονται τα αποτελέσματα της αναζήτησης του. Επίσης δίνεται η δυνατότητα να πάρουμε αποτελέσματα για περισσότερες από μια γειτονικές περιοχές με μία μόνο αναζήτηση την στιγμή που οι περισσότερες παρόμοιες εφαρμογές δεν μπορούν να δώσουν συνολικά αυτά τα αποτελέσματα.

Ακόμα σημαντικό στην εφαρμογή μας είναι τα φίλτρα αναζήτησης τα οποία δημιουργούνται δυναμικά, προσαρμοζόμενα ανάλογα με τα αποτελέσματα τις αρχικής αναζήτησης. Σε όσες από τις εφαρμογές παρουσιάσαμε παραπάνω διατίθεται λειτουργία φιλτραρίσματος, τα φίλτρα που παρέχουν είναι γενικά, και δεν προσαρμόζονται με βάση τα αποτελέσματα της αρχικής αναζήτησης.

Τέλος στην εφαρμογή μας δίνομαι την δυνατότητα αποστολής και λήψης προσωπικών μηνυμάτων μεταξύ των χρηστών. Τη δυνατότητα αυτή δεν την παρέχει καμία από τις παραπάνω εφαρμογές. Δημιουργήσαμε αυτή την δυνατότητα με σκοπό να ενδυναμώσουμε την επικοινωνία μεταξύ των χρηστών ώστε να μπορέσουμε να ενισχύσουμε το social κομμάτι της εφαρμογής και την έννοια της κοινότητας γύρω από αυτή.

# Συμπεράσματα

Στην παρούσα διπλωματική είδαμε τις δυνατότητες και την πρόσθετη πληροφορία που μπορεί να μας δώσει η χρήση του χάρτη σε εφαρμογές που έχει νόημα να παρέχεις γεωγραφικές πληροφορίες. Η διευκόλυνση του χρήστη στην ανάγνωση των πληροφοριών και στην αναζήτηση με βάση γεωγραφικά χαρακτηριστικά είναι χαρακτηριστική στην εφαρμογή μας. Η χρήση του χάρτη θα μπορούσε να χρησιμοποιηθεί σε ακόμα μεγαλύτερο βάθος για να παρέχει ακόμα περισσότερες πληροφορίες, όπως για παράδειγμα πλοήγηση σε κάποια από τα σημεία ενδιαφέροντος που παρουσιάζονται σε αυτόν.

Ακόμα είδαμε τις επιπρόσθετες δυνατότητες που μπορεί να δώσει η χρήση της java στην ανάπτυξη διαδικτυακών εφαρμογών. Οι τεχνολογίες servlet και jsp που χρησιμοποιήθηκαν είναι δωρεάν και παρέχουν όλη την δύναμη της γλώσσας java σε μορφή κατάλληλη για την ανάπτυξη δυναμικών διαδικτυακών εφαρμογών.

Τέλος η αρχιτεκτονική Model View Controller που χρησιμοποιήσαμε έδεσε αρμονικά τις δύο παραπάνω τεχνολογίες και μας έδωσε την δυνατότητα να εκμεταλλευτούμε τα πλεονεκτήματα κάθε μίας εξαλείφοντας ταυτόχρονα τα μειονεκτήματά τους. Επιπροσθέτως η διαίρεση του κώδικα σε ανεξάρτητες λειτουργίες δίνει την δυνατότητα η εφαρμογή μας να αναβαθμίζεται εύκολα με νέες τεχνολογίες και γενικά κάνει πιο εύκολη και με όσο το δυνατόν μικρότερο κόστος την συντήρησή της.

## Αναφορές


[1] Web mapping: http://en.wikipedia.org/wiki/Web_mapping

[2] Google Maps: http://en.wikipedia.org/wiki/Google_Maps

[3] Google Maps developing history: http://tinyurl.com/qyatth

[4] Google Maps for Enterprise: http://tinyurl.com/396wkdj

[5] Neighborhood Search Capability: http://tinyurl.com/yokks5

[6] Driving Directions Support Added to the Google Maps API: http://tinyurl.com/2hyqaa

[7] Introducing... Street View! : http://tinyurl.com/2qckt9

[8] It's a click & drag situation: http://tinyurl.com/2a3fmm

[9] Microformats in Google Maps: http://tinyurl.com/ypz8hy

[10] More of the world for you to explore: http://tinyurl.com/2ed3zc

[11] Explore new Terrain**:** http://tinyurl.com/26qww3

[12] It's your world. Map it: http://tinyurl.com/3bkpf5

[13] Last summer, somewhere in the Adirondacks... : http://tinyurl.com/2waeznj

[14] News flash: Maps now open to Flash developers: http://tinyurl.com/599bhl

[15] Pound the pavement: http://tinyurl.com/6zv53b

[16] More streets in more places: http://tinyurl.com/5dwsvd

[17] Google to buy GeoEye satellite imagery: http://tinyurl.com/6doyp9

[18] Geocoding... in Reverse! : http://tinyurl.com/6bpoqh

[19] Google Maps API for Flash + AIR: How did we ever breathe before?: http://tinyurl.com/5uesr3

[20] Class GMap2, Google Maps JavaScript API V2 Reference: http://tinyurl.com/3837ea9

[21] Class GLatLng, Google Maps JavaScript API V2 Reference: http://tinyurl.com/374kpde

[22] Συνάρτηση setCenter, Google Maps JavaScript API V2 Reference: http://tinyurl.com/2wlaadu

[23] Class GControl, Google Maps JavaScript API V2 Reference: http://tinyurl.com/33sops7

[24] Class GMarker, Google Maps JavaScript API V2 Reference: http://tinyurl.com/29qg3k8

[25] Class GMarkerOptions, Google Maps JavaScript API V2 Reference: http://tinyurl.com/2u8tnvr

[26] Namespace GEvent, Google Maps JavaScript API V2 Reference: http://tinyurl.com/36az7cu

[27] Geocoding Responses, The Google Geocoding API: http://tinyurl.com/3yejem5

[28] Java servlet: http://en.wikipedia.org/wiki/Java_Servlet

[29] J2EE web server or container: http://tinyurl.com/2vsuleg

[30] JSP Intro and Overview: http://tinyurl.com/2wkt8sn

[31] JavaServer Pages: http://en.wikipedia.org/wiki/JavaServer_Pages

[32] Model – View – Controller: http://tinyurl.com/2czr3jp

[33] John Deacon Model-View-Controller (MVC) Architecture: http://www.jdl.co.uk/briefings/MVC.pdf

[34] MVC design pattern brings about better organization and code reuse: http://tinyurl.com/2wzh3nf

[35] Professional JSP 2nd Edition, Wrox Press Inc, chapter 6: Combining Servlets, JSP, and JavaBeans: http://tinyurl.com/ya7deyo

[36] Data access object: http://en.wikipedia.org/wiki/Data_access_object

[37] Java Bean: http://en.wikipedia.org/wiki/JavaBean