Τ.Ε.Ι ΠΕΛΟΠΟΝΝΗΣΟΥ

Τμήμα Μηχανικών Πληροφορικής Τ.Ε.
Σχολή Τεχνολογικών εφαρμογών (Έδρα Σπάρτη)

Τρόποι δημιουργίας και συνεχούς μετεξέλιξης των προφίλ διαδικτυακών χρηστών

Παρασκευάς Δημήτρης
Σπάρτη 2016

Περιεχόμενα

Σύμφωνα με απόφαση της Συνέλευσης του ΤΕΙ Πελοποννήσου οι φοιτητές που εκπονούν την πτυχιακή τους εργασία υποχρεούνται να συμπεριλαμβάνουν στις προκαταρκτικές σελίδες της εργασίας τους το παρακάτω κείμενο, υπογεγραμμένο από τους ίδιους.

«ΔΗΛΩΣΗ ΜΗ ΛΟΓΟΚΛΟΠΗΣ ΚΑΙ ΑΝΑΛΗΨΗΣ ΠΡΟΣΩΠΙΚΗΣ ΕΥΘΥΝΗΣ

Με πλήρη επίγνωση των συνεπειών του νόμου περί πνευματικών δικαιωμάτων, δηλώνω ενυπογράφως ότι είμαι αποκλειστικός συγγραφέας της παρούσας Πτυχιακής Εργασίας, για την ολοκλήρωση της οποίας κάθε βοήθεια είναι πλήρως αναγνωρισμένη και αναφέρεται λεπτομερώς στην εργασία αυτή. Έχω αναφέρει πλήρως και με σαφείς αναφορές, όλες τις πηγές χρήσης δεδομένων, απόψεων, θέσεων και προτάσεων, ιδεών και λεκτικών αναφορών, είτε κατά κυριολεξία είτε βάση επιστημονικής παράφρασης. Αναλαμβάνω την προσωπική και ατομική ευθύνη ότι σε περίπτωση αποτυχίας στην υλοποίηση των ανωτέρω δηλωθέντων στοιχείων, είμαι υπόλογος έναντι λογοκλοπής, γεγονός που σημαίνει αποτυχία στην Πτυχιακή μου Εργασία και κατά συνέπεια αποτυχία απόκτησης του Τίτλου Σπουδών, πέραν των λοιπών συνεπειών του νόμου περί πνευματικών δικαιωμάτων. Δηλώνω, συνεπώς, ότι αυτή η Πτυχιακή Εργασία προετοιμάστηκε και ολοκληρώθηκε από εμένα προσωπικά και αποκλειστικά και ότι, αναλαμβάνω πλήρως όλες τις συνέπειες του νόμου στην περίπτωση κατά την οποία αποδειχθεί, διαχρονικά, ότι η εργασία αυτή ή τμήμα της δε μου ανήκει διότι είναι προϊόν λογοκλοπής άλλης πνευματικής ιδιοκτησίας.

Όνομα και Επώνυμο Συγγραφέα (Με Κεφαλαία):
ΠΑΡΑΣΚΕΥΑΣ…ΔΗΜΗΤΡΗΣ……..
Υπογραφή (Ολογράφως, χωρίς μονογραφή):
ΠΑΡΑΣΚΕΥΑΣ…ΔΗΜΗΤΡΗΣ……..
Ημερομηνία (Ημέρα – Μήνας – Έτος):
……
»

Πρόλογος

Η παρούσα πτυχιακή εργασία πραγματοποιήθηκε στη Σχολή Τεχνολογικών εφαρμογών(έδρα: Σπάρτη) Τ.Ε.Ι Πελοποννήσου στο τμήμα Μηχανικών Πληροφορικής Τ.Ε. Στόχος της εργασίας είναι η μελέτη των διαδικτυακών προφίλ χρηστών και η υλοποίηση μιας ολοκληρωμένης εφαρμογής ηλεκτρονικού καταστήματος, η οποία θα κάνει σύσταση προτάσεων στους χρήστες, βάση της παρελθοντικής τους συμπεριφοράς.

Θα ήθελα να ευχαριστήσω θερμά τον καθηγητή κ. Γκατζιώλη Κλεάνθη για την εμπιστοσύνη που έδειξε στο πρόσωπο μου και για τη δυνατότητα που μου έδωσε να εκπονήσω αυτού του είδους πτυχιακή εργασία. Ήταν ένα αντικείμενο πρωτόγνωρο για μένα, που κέντρισε το ενδιαφέρον μου και μου άνοιξε νέους δρόμους μελέτης και έρευνας.

Τέλος θα ήθελα να ευχαριστήσω θερμά την οικογένεια μου για την ηθική τους συμπαράσταση κατά τη διάρκεια της έρευνας και συγγραφής της πτυχιακής αυτής εργασίας.

1. Εισαγωγή

Τα τελευταία χρόνια έχει προκύψει έντονο ενδιαφέρον για την παρακολούθηση και τη καταγραφή της συμπεριφοράς των χρηστών στις ιστοσελίδες. Ο στόχος είναι η κατανόηση των επιθυμιών των χρηστών με σκοπό, μέσω της αξιοποίησης αυτής της πληροφορίας, την προβολή των κατάλληλων διαφημίσεων. Η προβολή των διαφημίσεων προς ένα χρήστη θα είναι εξατομικευμένη, καθώς θα χρησιμοποιεί τις διαθέσιμες πληροφορίες για τον χρήστη αυτό.

Ως εκ τούτου, προέκυψε η ανάγκη της δημιουργίας διαδικτυακών προφίλ για τους χρήστες μιας εφαρμογής. Η πληροφορία για τα προφίλ των χρηστών αποθηκεύονται σε βάσεις δεδομένων. Οι βάσεις δεδομένων προσφέρουν τη δυνατότητα της αποθήκευσης/ανάκτησης και τροποποίησης πληροφοριών. Οι βάσεις δεδομένων προσφέρουν πολλά πλεονεκτήματα στις εφαρμογές που τις χρησιμοποιούν, με κυριότερο τη δυνατότητα διαχείρισης μεγάλου όγκου πληροφοριών με έναν αποδοτικό τρόπο.

Η πιο δημοφιλής εφαρμογή των διαδικτυακών προφίλ είναι τα συστήματα σύστασης προτάσεων. Οι recommenders χρησιμοποιούν την πληροφορία της συμπεριφοράς των χρηστών και δημιουργούν ένα μοντέλο, βάση του οποίου κάνουν εξατομικευμένες συστάσεις στους χρήστες της εφαρμογής.

2. Διαδικτυακά Προφίλ Χρηστών

Τα διαδικτυακά προφίλ χρηστών αποτελούν μια οπτική απεικόνιση των προσωπικών δεδομένων που σχετίζονται με ένα συγκεκριμένο χρήστη. Επομένως το προφίλ ενός χρήστη αναφέρεται στη ρητή ψηφιακή αναπαράσταση της ταυτότητας του ίδιου. Επιπρόσθετα τα προφίλ χρηστών χρησιμοποιούνται για την αποθήκευση της περιγραφής των χαρακτηριστικών των χρηστών. Η πληροφορία αυτή μπορεί να αξιοποιηθεί από τις διαδικτυακές εφαρμογές με σκοπό την παροχή ενός εξατομικευμένου περιβάλλοντος χρήσης και πλοήγησης προς τους χρήστες.

Τα προφίλ χρηστών διακρίνονται σε δύο κατηγορίες:

- Τα στατικά προφίλ
- Τα δυναμικά προφίλ

Στα στατικά προφίλ χρηστών, η πληροφορία για τους χρήστες μιας εφαρμογής εξαρτάται συνήθως αποκλειστικά από τις πληροφορίες που θα δώσουν οι ίδιοι οι χρήστες στην εφαρμογή. Επομένως η εφαρμογή δεν χρησιμοποιεί καμία τεχνική για την ανάκτηση επιπρόσθετων στοιχείων για το χρήστη μέσω της συμπεριφοράς αυτού εντός στο σύστημα. Το πιο κλασσικό παράδειγμα είναι οι εφαρμογές που η μόνη πληροφορία που έχουν για ένα χρήστη είναι αυτή που δίνεται κατά τη διάρκεια της εγγραφής του χρήστη στην εφαρμογή. Μια πιο δυναμική περίπτωση στατικών προφίλ είναι οι εφαρμογές που επιτρέπουν στο χρήστη είτε να τροποποιεί υπάρχοντες πληροφορίες για το προφίλ του είτε να εισάγουν νέες. Για παράδειγμα, μια εφαρμογή, η οποία προσφέρει στο ευρύ κοινό τη δυνατότητα προβολής online ταινιών μέσω του διαδικτύου μπορεί να επιτρέπει στους χρήστες να δηλώσουν τις προτιμήσεις τους σε είδη ταινιών. Την πληροφορία αυτή μπορεί να χρησιμοποιηθεί από την εφαρμογή για να προτείνει στο χρήστη ταινίες που πιθανόν να τον ενδιαφέρουν.

Σε αντίθεση με τα στατικά προφίλ χρηστών, τα δυναμικά προφίλ χρησιμοποιούν διάφορες τεχνικές για την εξαγωγή δεδομένων για ένα χρήστη. Την πληροφορία αυτή θα είναι σε θέση να χρησιμοποιήσει η εφαρμογή για να προσφέρει υπηρεσίες στο χρήστη, βάση της ταυτότητας συμπεριφοράς που έχει δημιουργήσει γι’ αυτόν.

Η πληροφορία για τις προτιμήσεις και τις προτιμήσεις ενός χρήστη αποτελεί ένα πολύ σημαντικό όπλο στα χέρια του «marketing». Γνωρίζοντας τις καταναλωτικές επιθυμίες ενός χρήστη, τα συστήματα είναι σε θέση να διαφημίζουν προϊόντα τα οποία με αρκετά μεγάλη πιθανότητα θα ενδιαφέρουν το χρήστη. Με τον τρόπο αυτό, τα συστήματα επιτυγχάνουν να προτείνουν/ενημερώνουν τους χρήστες τους για νέα προϊόντα/υπηρεσίες, τα οποία θεωρούνται ενδιαφέροντα προς αυτούς. Αυτό έχει ως αποτέλεσμα την αγορά προϊόντων και υπηρεσιών από τους χρήστες, των οποίων πιθανόν να αγνοούσαν την ύπαρξη τους αν δεν τους γινόταν πρόταση από το σύστημα εκμεταλλευόμενο την αντίστοιχη πληροφόρηση.

3. Συστήματα Προτάσεων

Η πιο σημαντική εφαρμογή των προφίλ χρηστών είναι η σύσταση προτάσεων. Τα συστήματα σύστασης προτάσεων συγκεντρώνουν την online πληροφορία που είναι διαθέσιμη στο προφίλ κάθε χρήστη. Εν συνεχεία αξιοποιούν την εν λόγο πληροφορία, με στόχο να προσδιορίσει την καταναλωτική συμπεριφορά του χρήστη και να κάνει τις κατάλληλες συστάσεις βάση τις προτιμήσεις κάθε χρήστη εξατομικευμένα.

3.1 Κατηγορίες Συστημάτων Προτάσεων

Τα συστήματα προτάσεων παράγουν μια λίστα προτάσεων με βάσει είτε μιας μεθόδου συνεργατικού φιλτραρίσματος (collaborative filtering) είτε μιας μεθόδου φιλτραρίσματος βασισμένου στο περιεχόμενο (content-based filtering). Οι προσεγγίσεις που βασίζονται στο συνεργατικό φιλτράρισμα οικοδομούν ένα μοντέλο βάσει της παρελθοντικής συμπεριφοράς του χρήστη (π.χ. αντικείμενα που έχουν αποκτηθεί στο παρελθόν και βαθμολογήσεις που έχει δώσει) καθώς και παρόμοιων αποφάσεων που έχουν παρθεί από άλλους χρήστες, και βάσει του μοντέλου αυτού αξιολογούν αντικείμενα και προβλέπουν ποια από αυτά είναι πιθανότερο να ενδιαφέρουν τον χρήστη. Αντίθετα, οι προσεγγίσεις φιλτραρίσματος βάσει του περιεχομένου χρησιμοποιούν ένα σύνολο από διακριτά χαρακτηριστικά ενός αντικειμένου, προκειμένου να προτείνουν επιπλέον αντικείμενα με παρόμοιες ιδιότητες.

Οι διαφορές μεταξύ συνεργατικού φιλτραρίσματος και φιλτραρίσματος βασισμένου στο περιεχόμενο μπορούν να κατανοηθούν μέσω της σύγκρισης δύο δημοφιλών συστημάτων προτάσεων μουσικής όπως είναι το Last.fm και το Pandora Radio.

- Το Pandora Radio χρησιμοποιεί μια προσέγγιση προτάσεων βασισμένη στο περιεχόμενο. Το Pandora χρησιμοποιεί τις ιδιότητες ενός καλλιτέχνη ή τραγουδιού προκειμένου να δημιουργήσει ένα σταθμό που παίζει μουσική με παρόμοιες ιδιότητες. Οι ιδιότητες μπορεί να αφορούν είτε χαρακτηριστικά των ίδιων των τραγουδιών είτε χαρακτηριστικά των καλλιτεχνών των τραγουδιών. Επιπρόσθετα τα σχόλια των χρηστών χρησιμοποιούνται για να βελτιωθούν τα αποτελέσματα του σταθμού. Πιο συγκεκριμένα δίνει έμφαση στα χαρακτηριστικά ενός τραγουδιού που έχει θετικές κριτικές και υποβαθμίζει τη βαρύτητα των χαρακτηριστικών ενός τραγουδιού που έχει αρνητικές κριτικές.

- Αντίθετα το Last.fm χρησιμοποιεί μια προσέγγιση προτάσεων βασισμένη στο συνεργατικό φιλτράρισμα. Το Last.fm στοχεύει στο να προταθεί ενός σταθμού από προτεινόμενα τραγούδια παρατηρώντας τα συγκροτήματα και τα μεμονωμένα κομμάτια που έχουν ακουστεί σε τακτική βάση από ένα χρήστη και συγκρίνοντας τα με την ακουστική συμπεριφορά άλλων χρηστών. Πιο συγκεκριμένα το Last.fm θα παίξει

κομμάτια που δεν εμφανίζονται στη βιβλιοθήκη ενός χρήστη, αλλά συχνά παίζονται από χρήστες με παρόμοια ακουστική συμπεριφορά.

Κάθε ένας από τους δύο προαναφερθείς τύπους συστάσεων έχει τα πλεονεκτήματα και τις αδυναμίες του. Αρχικά, οι μέθοδοι συνεργατικού φιλτραρίσματος, έχουν αρκετά μεγαλύτερο πεδίο εφαρμογής, καθώς μπορούν να εφαρμοστούν τόσο σε εφαρμογές που δεν υπάρχει αρκετό διαθέσιμο περιεχόμενο, το οποίο να σχετίζεται με τα αντικείμενα, καθώς επίσης και σε εφαρμογές, στις οποίες είναι δύσκολη η ανάλυση των περιεχομένων των αντικειμένων, όπως είναι οι ιδέες, οι γνώμες, κ.α. Επιπρόσθετα, οι μέθοδοι συνεργατικού φιλτραρίσματος, μπορούν να παρέχουν συγκυριακές προτάσεις, δηλαδή να προτείνουν αντικείμενα, τα οποία είναι «σχετικά» με τις προτιμήσεις του χρήστη, χωρίς ωστόσο να περιλαμβάνουν στοιχεία, τα οποία χαρακτηρίζουν το προφίλ ενός χρήστη. Για τους δύο παραπάνω λόγους, οι τεχνικές συνεργατικού φιλτραρίσματος, χρησιμοποιήθηκαν ευρέως και με αρκετά μεγάλη επιτυχία για την κατασκευή συστημάτων προτάσεων.

Ωστόσο, οι μέθοδοι συνεργατικού φιλτραρίσματος μειονεκτούν λόγο δύο σημαντικών προβλημάτων. Αρχικά, οι χρήστες συνηθίζουν να αξιολογούν ένα πολύ μικρό ποσοστό του συνόλου των αντικειμένων ενός συστήματος, με αποτέλεσμα τη δημιουργία αρκετά αραιών πινάκων ομοιότητας μεταξύ χρηστών και αντικειμένων. Για το λόγο αυτό, σε πολλές εφαρμογές, η πιθανότητα εύρεσης ενός συνόλου από χρήστες με σημαντικά παρόμοιες βαθμολογήσεις είναι αρκετά μικρή. Το πρόβλημα αυτό είναι ιδιαίτερα εμφανές κατά τη διάρκεια της έναρξης της χρήσης του συστήματος από έναν χρήστη, το οποίο ονομάζεται και ως πρόβλημα ψυχρής εκκίνησης, καθώς οι προτάσεις χάνουν σε ακρίβεια, μέχρι να μαζευτούν αρκετές πληροφορίες, σχετικά με τη συμπεριφορά ενός χρήστη. Ένα δεύτερο, αλλά πιθανόν λιγότερο σημαντικό, πρόβλημα που παρουσιάζουν οι τεχνικές του συνεργατικού φιλτραρίσματος, αφορά την αδυναμία σύστασης πρότασης ενός αντικειμένου με μηδενικές βαθμολογήσεις. Το παραπάνω πρόβλημα είναι αρκετά πιθανόν να μην υφίσταται σε τεχνικές βασισμένες στο περιεχόμενο. Στο παράδειγμα με τα συστήματα προτάσεων μουσικών τραγουδιών, ένα νέο τραγούδι, το οποίο δεν έχει καμία κριτική, πιθανόν να μπορεί να προταθεί σε ένα χρήστη από ένα σύστημα προτάσεων βασισμένο στο περιεχόμενο, καθώς μπορεί να έχει αρκετά δημοφιλή χαρακτηριστικά, όπως είναι η δυναμική μουσική και ότι ανήκει σε καλλιτέχνη μεγάλης δημοτικότητας.

Όσον αφορά τη μέθοδο του συνεργατικού φιλτραρίσματος υπάρχουν δύο διαφορετικοί τύποι αλγορίθμων:

- Συνεργατικό φιλτράρισμα βασισμένο στους χρήστες
- Συνεργατικό φιλτράρισμα βασισμένο στα αντικείμενα

Το συνεργατικό φιλτράρισμα βασισμένο στους χρήστες υπολογίζει το ενδιαφέρον ενός χρήστη για ένα αντικείμενο, βάση της πληροφορίας για τις βαθμολογίες που έχουν δώσει άλλοι «παρόμοιοι» χρήστες. Αντίθετα, στο συνεργατικό φιλτράρισμα βασισμένο στα αντικείμενα, εφαρμόζεται η ίδια ιδέα,

με τη διαφορά ότι χρησιμοποιείται η ομοιότητα που βασίζεται στα αντικείμενα και όχι στους χρήστες.

Στα πρώτα χρόνια της χρήση των αλγορίθμων συνεργατικού φιλτραρίσματος, τα συστήματα συνήθως χρησιμοποιούσαν τους αλγορίθμους που βασίζονταν στην ομοιότητα μεταξύ των χρηστών. Ωστόσο οι αλγόριθμοι αυτοί παρουσίαζαν τα παρακάτω τρία προβλήματα:

1. Η απόδοση των συστημάτων ήταν φτωχή όταν υπήρχαν πολλά αντικείμενα και όχι αρκετές βαθμολογίες
2. Ο υπολογισμός των ομοιοτήτων μεταξύ χρηστών είναι μια πολύ βαριά υπολογιστικά διαδικασία
3. Τα προφίλ των χρηστών αλλάζουν συχνά και επομένως ολόκληρο το μοντέλο έπρεπε να ξανά εκτελεστεί

Τα μειονεκτήματα αυτά επιλύει το συνεργατικό φιλτράρισμα που βασίζεται στα αντικείμενα, καθώς υπολογίζεται η ομοιότητα μεταξύ των αντικειμένων και όχι των χρηστών.

3.2 Πλεονεκτήματα Χρήσης Σύστασης Προτάσεων

Η εμπορική επιτυχία των εφαρμογή που προτείνουν προϊόντα στους χρήστες είναι ιδιαίτερα σημαντική. Η χρήση αλγορίθμων σύστασης προτάσεων σε εμπορικές εφαρμογές δεν είναι πλέον μια καινοτομία, αλλά μια επιτακτική ανάγκη με σκοπό την αύξηση της παραγωγικότητας της εκάστοτε εφαρμογής. Τα recommender systems χρησιμοποιούνται σε εφαρμογές σχεδόν οποιουδήποτε τομέα, στον οποίον είναι εφικτή μια τέτοια ενέργεια και τα πλεονεκτήματα τους είναι σαφείς [1].

- **Βασίζεται στις ενέργειες του χρήστη**
  Το μεγαλύτερο όφελος που προκύπτει από τη χρήση των συστημάτων προτάσεων είναι το γεγονός ότι χρησιμοποιούν την πληροφορία που υπάρχει για ένα χρήστη και δομούν ένα εμπλουτισμένο προφίλ, βάση του οποίου θα γίνουν οι συστάσεις στους χρήστες. Το γεγονός ότι οι συστάσεις βασίζονται στο δομημένο προφίλ ενός χρήστη και όχι σε εικασίες και τυχαίες επιλογές, αποτελεί τον κύριο λόγο ύπαρξης των recommenders, καθώς αυξάνει την πιθανότητα επιτυχίας των συστάσεων.

- **Προσαρμόζεται δυναμικά**
  Τα συστήματα σύστασης προτάσεων χρησιμοποιούν πληροφορία, η οποία ενημερώνεται δυναμικά και ως εκ τούτου είναι up to date. Πιο συγκεκριμένα, τα recommendation systems χρησιμοποιούν τη πληροφορία των αγορών των χρηστών, καθώς και τις βαθμολογίες που έχουν δώσει για προϊόντα. Αυτή η πληροφορία αλλάζει με το πέρασμα του χρόνου. Το γεγονός ότι οι αλγόριθμοι σύστασης προτάσεων έχουν δημιουργηθεί με σκοπό να προσαρμόζονται δυναμικά στις αλλαγές που παρουσιάζουν οι αγοραστικές συνήθειες

των χρηστών, έχει ως αποτέλεσμα να κάνουν συστάσεις, οι οποίες βασίζονται στην πλέον τρέχουσα πληροφορία που υπάρχει για τους χρήστες.

- **Εξατομικευμένο πάνω στο χρήστη**
  Ένα ακόμη σημαντικό χαρακτηριστικό των αλγορίθμων σύστασης προτάσεων είναι ότι προτείνουν αντικείμενα, τα οποία βασίζονται στις εξατομικευμένες προτιμήσεις ενός χρήστη. Σε αντίθεση με τρόπους σύστασης προτάσεων, οι οποίοι βασίζονται μόνο στη δημοτικότητα ενός αντικειμένου, οι recommenders κάνουν συστάσεις λαμβάνοντας υπόψη τόσο τη δημοτικότητα και την βαθμολόγηση από το σύνολο των χρηστών, όσο και τις προτιμήσεις που έχει ο χρήστης στον οποίον γίνεται η σύσταση προτάσεων. Αυτό το στοιχείο είναι πολύ σημαντικό, καθώς οι προτιμήσεις μεταξύ των χρηστών διαφέρουν και επομένως η σύσταση προτάσεων θα πρέπει να λαμβάνει υπόψη το χαρακτηριστικό αυτό.

4. Βάσεις Δεδομένων

Η αλματώδης ανάπτυξη της επιστήμης της πληροφορικής και των επικοινωνιών τα τελευταία χρόνια έχει καταστήσει την πληροφορία ως ένα από τα βασικότερα και πολυτιμότερα αγαθά. Είναι κοινός τόπος σήμερα η εκτίμηση ότι το αγαθό της πληροφορίας είναι επιθυμητό απ' όλους τους εργαζόμενους αλλά και τους εκπαιδευόμενους, ώστε να είναι πιο αποδοτικοί, ανταγωνιστικοί αλλά και παραγωγικοί στην εργασία τους [2].

Τα συστήματα βάσεων δεδομένων τα χρησιμοποιούμε για να μπορούμε να αποθηκεύσουμε, να επεξεργαστούμε αλλά και να εκμεταλλευτούμε αποδοτικά αυτόν τον τεράστιο όγκο των πληροφοριών που αυξάνονται με αλματώδεις ρυθμούς καθημερινά.

4.1 Δεδομένα & Πληροφορίες

Με τον όρο πληροφορία αναφερόμαστε συνήθως σε ειδήσεις, γεγονότα και έννοιες που αποκτήσαμε από την καθημερινή μας επικοινωνία και τα θεωρούμε ως αποκτημένη γνώση. Αντίθετα τα δεδομένα μπορούν να μην είναι κατάλληλα επεξεργασμένα και να μην είναι ταξινομημένα τα σύνολα των πληροφοριών. Ένας αυστηρός ορισμός που δίνεται για το τι είναι δεδομένα και τι πληροφορία σύμφωνα με την επιτροπή ANSI των ΗΠΑ είναι ο εξής:

- Δεδομένα είναι μια παράσταση, όπως γράμματα, αριθμοί, σύμβολα κ.α. στα οποία μπορούμε να δώσουμε κάποια έννοια

- Πληροφορία είναι η σημασία που δίνουμε σ' ένα σύνολο από δεδομένα, τα οποία μπορούμε να επεξεργαστούμε βάσει προκαθορισμένων κανόνων και να βγάλουμε έτσι κάποια χρήσιμα συμπεράσματα. Με τις πληροφορίες περιορίζεται η αβεβαιότητα που έχουμε για διάφορα πράγματα και βοηθιόμαστε έτσι στο να λάβουμε σωστές αποφάσεις.

Τα δεδομένα μπορούν να θεωρηθούν ως τρόποι αναπαράστασης εννοιών και γεγονότων που μπορούν να υποστούν διαχείριση και επεξεργασία. Η συλλογή και αποθήκευση ενός τεράστιου όγκου δεδομένων όπως απαιτούν οι κοινωνικές συνθήκες σήμερα, δεν λύνει τελείως το πρόβλημα της σωστής οργάνωσης και ταξινόμησης των δεδομένων. Τα δεδομένα θα πρέπει να οργανωθούν με τέτοιο τρόπο έτσι ώστε να μπορούμε να τα εντοπίζουμε και να τα αξιοποιούμε εύκολα και γρήγορα και τη στιγμή που τα χρειαζόμαστε.

Ένα κλασικό παράδειγμα μη σωστής οργάνωσης δεδομένων από την καθημερινότητα μας, θα μπορούσε να θεωρηθεί ένας τηλεφωνικός κατάλογος. Για παράδειγμα ο τηλεφωνικός κατάλογος της πόλης της Θεσσαλονίκης, όπου οι συνδρομητές δεν θα ήταν καταχωρημένοι αλφαβητικά σύμφωνα με το επώνυμο και το όνομά τους όπως είθισται, αλλά εντελώς τυχαία. Ένας τέτοιος τηλεφωνικός κατάλογος θα περιείχε μια τεράστια ποσότητα δεδομένων αλλά θα ήταν ουσιαστικά μη εκμεταλλεύσιμη.

Ένα άλλο παράδειγμα μη σωστής οργάνωσης δεδομένων θα ήταν μια πολύ μεγάλη βιβλιοθήκη με χιλιάδες τόμους βιβλίων και χωρίς να διαθέτει κάποιο υποτυπώδες σύστημα οργάνωσης και ταξινόμησης των βιβλίων. Ούτε οι υπάλληλοι της βιβλιοθήκης θα μπορούσαν να κάνουν τη δουλειά τους αλλά ούτε και οι επισκέπτες θα μπορούσαν να αξιοποιήσουν την πληθώρα των πληροφοριών που περιέχονται στα βιβλία. Εκτός λοιπόν από τη μόνιμη αποθήκευση των δεδομένων, χρειαζόμαστε και κάποιους τρόπους ευέλικτης και αποδοτικής οργάνωσής τους.

Χαρακτηριστικά παραδείγματα δεδομένων που απαιτούν σωστή και αποδοτική οργάνωση είναι τα εξής:

- Τα στοιχεία υπαλλήλων, πελατών, προμηθευτών και παραγγελιών μιας εμπορικής επιχείρησης.
- Τα στοιχεία υλικών μιας αποθήκης.
- Τα στοιχεία ταινιών, πελατών και δανεισμών μιας βιντεολέσχης.
- Τα στοιχεία υπαλλήλων, γιατρών, ασθενών αλλά και υλικών ενός νοσοκομείου.
- Τα στοιχεία βιβλίων, χρηστών (δανειστών) και δανεισμών μιας βιβλιοθήκης.

4.2 Συστήματα Διαχείρισης Βάσεων Δεδομένων

Μια Βάση Δεδομένων (ΒΔ) είναι ένα σύνολο αρχείων με υψηλό βαθμό οργάνωσης τα οποία είναι συνδεδεμένα μεταξύ τους με λογικές σχέσεις, έτσι ώστε να μπορούν να χρησιμοποιούνται από πολλές εφαρμογές και από πολλούς χρήστες ταυτόχρονα. Υπάρχει ένα ειδικό λογισμικό το οποίο μεσολαβεί ανάμεσα στις αρχεία δεδομένων και τις εφαρμογές που χρησιμοποιούν οι χρήστες και αποκαλείται Σύστημα Διαχείρισης Βάσης Δεδομένων (ΣΔΒΔ) ή DBMS (Data Base Management System). Το ΣΔΒΔ είναι στην ουσία ένα σύνολο από προγράμματα και υποροutίνες που έχουν να κάνουν με τον χειρισμό της βάσης δεδομένων, όσον αφορά τη δημιουργία, τροποποίηση, διαγραφή στοιχείων, με ελέγχους ασφαλείας κ.ά.

Οι χρήστες των εφαρμογών αντλούν τα στοιχεία που τους ενδιαφέρουν από τη βάση δεδομένων χωρίς να είναι σε θέση να γνωρίζουν με ποιο τρόπο είναι οργανωμένα τα δεδομένα σ' αυτήν. Το ΣΔΒΔ παίζει τον ρόλο του μεσάζοντα ανάμεσα στον χρήστη και τη βάση δεδομένων και μόνο μέσω του ΣΔΒΔ μπορεί ο χρήστης να αντλήσει πληροφορίες από τη βάση δεδομένων. Ένα ΣΔΒΔ μπορεί να είναι εγκατεστημένο σ' έναν μόνο υπολογιστή ή και σ' ένα δίκτυο υπολογιστών και μπορεί να χρησιμοποιείται από έναν χρήστη ή και από πολλούς χρήστες.

Ένα Σύστημα Βάσης Δεδομένων (ΣΒΔ) ή DBS (Data Base System) αποτελείται από το υλικό, το λογισμικό, τη βάση δεδομένων και τους χρήστες. Είναι δηλαδή ένα σύστημα με το οποίο μπορούμε να αποθηκεύσουμε και να αξιοποιήσουμε δεδομένα με τη βοήθεια ηλεκτρονικού υπολογιστή. Αναλυτικά :

- Το υλικό (hardware) αποτελείται όπως είναι γνωστό από τους ηλεκτρονικούς υπολογιστές, τα περιφερειακά, τους σκληρούς δίσκους, τις μαγνητικές ταινίες κ.ά., όπου είναι αποθηκευμένα τα αρχεία της βάσης δεδομένων αλλά και τα προγράμματα που χρησιμοποιούνται για την επεξεργασία τους.
- Το λογισμικό (software) είναι τα προγράμματα που χρησιμοποιούνται για την επεξεργασία των δεδομένων (στοιχείων) της βάσης δεδομένων.
- Η βάση δεδομένων (data base) αποτελείται από το σύνολο των αρχείων όπου είναι αποθηκευμένα τα δεδομένα του συστήματος. Τα στοιχεία αυτά μπορεί να βρίσκονται αποθηκευμένα σ' έναν φυσικό υπολογιστή αλλά και σε περισσότερους. Όμως, στον χρήστη δίνεται η εντύπωση ότι βρίσκονται συγκεντρωμένα στον ίδιο υπολογιστή. Τα δεδομένα των αρχείων αυτών είναι ενοποιημένα (data integration), δηλ. δεν υπάρχει πλεονασμός (άσκοπη επανάληψη) δεδομένων και μερισμένα (data sharing), δηλ. υπάρχει δυνατότητα ταυτόχρονης προσπέλασης των δεδομένων από πολλούς χρήστες. Ο κάθε χρήστης έχει διαφορετικά δικαιώματα και βλέπει διαφορετικό κομμάτι της βάσης δεδομένων, ανάλογα με τον σκοπό για τον οποίο συνδέεται.
- Οι χρήστες (users) μιας βάσης δεδομένων χωρίζονται στις εξής κατηγορίες :

- Τελικοί χρήστες (end users). Χρησιμοποιούν κάποια εφαρμογή για να παίρνουν στοιχεία από μια βάση δεδομένων, έχουν τις λιγότερες δυνατότητες επέμβασης στα στοιχεία της βάσης δεδομένων, χρησιμοποιούν ειδικούς κωδικούς πρόσβασης και το σύστημα τούς επιτρέπει ανάλογα πρόσβαση σε συγκεκριμένο κομμάτι της βάσης δεδομένων.
- Προγραμματιστές εφαρμογών (application programmers). Αναπτύσσουν τις εφαρμογές του ΣΒΔ σε κάποια από τις γνωστές γλώσσες προγραμματισμού.
- Διαχειριστής δεδομένων (data administrator – DA). Έχει τη διοικητική αρμοδιότητα και ευθύνη για την οργάνωση της βάσης δεδομένων και την απόδοση δικαιωμάτων πρόσβασης στους χρήστες.
- Διαχειριστής βάσης δεδομένων (database administrator – DBA). Λαμβάνει οδηγίες από τον διαχειριστή δεδομένων και είναι αυτός που διαθέτει τις τεχνικές γνώσεις και αρμοδιότητες για τη σωστή και αποδοτική λειτουργία του ΣΔΒΔ.

5. MySQL

Η MySQL [3] είναι ένα σύστημα ΒΔ, που σχεδιάστηκε για τη διαχείριση δεδομένων, σε ένα περιβάλλον διαχείρισης σχεσιακών βάσεων δεδομένων, η οποία ήταν βασισμένη στη σχεσιακή άλγεβρα. Η γλώσσα έχει δυνατότητες ανάκτησης και ενημέρωσης δεδομένων, δημιουργίας και τροποποίησης σχημάτων και σχεσιακών πινάκων και ελέγχου πρόσβασης στα δεδομένα. Ο MySQL διακομιστής ελέγχει την πρόσβαση στα δεδομένα σας, για να μπορούν να δουλεύουν πολλοί χρήστες ταυτόχρονα. Επιπρόσθετα είναι ένας πολυνηματικός διακομιστής πολλαπλών χρήσεων, ο οποίος παρέχει γρήγορη πρόσβαση και διασφαλίζει ότι μόνο πιστοποιημένοι χρήστες μπορούν να έχουν πρόσβαση. Η MySQL είναι μια δημοφιλής βάση δεδομένων για διαδικτυακά προγράμματα και ιστοσελίδες και χρησιμοποιείται σε ορισμένες από τις πιο διαδεδομένες διαδικτυακές υπηρεσίες.

5.1 Χαρακτηριστικά της MySQL

Η MySQL είναι πολύ γρήγορη, αξιόπιστη και πολύ εύκολη στη χρήση της. Έχει ένα συγκεκριμένο σύνολο χαρακτηριστικών που έχει αναπτυχθεί σε συνεργασία με τους χρήστες. Η MySQL είχε σχεδιαστεί για να είναι σε θέση να χειρίζεται μεγάλες βάσεις πολύ γρηγορότερα από αρκετές ήδη υπάρχουσες λύσεις. Προσφέρει ένα πλούσιο και χρήσιμο σύνολο συναρτήσεων. Επιπλέον, η ταχύτητα και η ασφάλεια που προσφέρει την

καθιστούν ως ένα από τα πλέον κατάλληλα συστήματα βάσεων δεδομένων για την αποθήκευση και την προσπέλαση πληροφοριών.

Η Mysql είναι ένα client/server σύστημα το οποίο αποτελείται από έναν SQL server που υποστηρίζει διαφορετικές βιβλιοθήκες και προγράμματα client, αρκετά interfaces και διαχειριστικές μμεθόδους. Επίσης είναι δυνατή η σύνδεση της σε οποιαδήποτε εφαρμογή με αποτέλεσμα ένα μικρότερο και γρηγορότερο προϊόν. Πολλές γλώσσες ή ακόμη και εφαρμογές ήδη υποστηρίζουν την Mysql.

Μερικά επιπρόσθετα χαρακτηριστικά της MySQL είναι τα εξής:

- Μπορεί εύκολα να χρησιμοποιήσει πολλαπλές κεντρικές μμονάδες επεξεργασίας, αν αυτές είναι διαθέσιμες
- Μπορεί και δουλεύει σε πολλές διαφορετικές φόρμες
- Έχει τη δυνατότητα να συνδυάσει tables από διαφορετικές βάσεις στην ίδια προσπάθεια ζήτησης των δεδομένων και μάλιστα πολύ γρήγορα
- Ασχολείται με μμεταβλητές πολλών διαφορετικών τύπων, όπως είναι μμεταβλητές μήκους 1,2,3,4,8 bytes καθώς και μεταβλητές τύπου FLOAT, DOUBLE, CHAR, VARCHAR, TEXT, DATE,TIMESTAMP, DATETIME, YEAR κ.α.
- Είναι ευέλικτη και ασφαλής επειδή όλη η μεταφορά των passwords γίνεται κρυφά όταν γίνεται η σύνδεση με τον server
- Δεν παρατηρείται “διαρροή” της μνήμης
- Όλα τα δεδομένα σώζονται στο επιλεγμένο σύνολο χαρακτήρων και η ταξινόμηση γίνεται βάση αυτού. Βέβαια αυτό μπορεί να αλλάξει όταν ο server ξεκινά από την αρχή
- Υποστήριξη ψευδώνυμων στα tables και στις στήλες αυτών
- Οι χρήστες μπορούν να συνδεθούν στον Mysql server χρησιμοποιώντας το TCP/IP, UNIX

ΕΙΚΟΝΑ 1: HTTP://COMMONS.WIKIMEDIA.ORG/WIKI/FILE:PHP-LOGO.SVG

6. PHP

Η PHP είναι μια γλώσσα προγραμματισμού για web servers, χρήσιμη για δυναμικές ιστοσελίδες, οι οποίες μπορούν να διασυνδέσουν με μια βάση δεδομένων.
Η γλώσσα PHP προσφέρεται για εύκολη διαχείριση της ροής των δεδομένων μιας εφαρμογής, και ταυτόχρονα μπορεί να μετατραπεί σε ένα πολύ ευέλικτο αλγοριθμικό εργαλείο, που εκτελεί και διαμορφώνει εντολές για προχωρημένους ή ακόμα και για μεγάλες διαδικτυακές εφαρμογές.

6.1 Ιστορία της PHP

Η ιστορία της PHP ξεκινά από το 1994, όταν ένας φοιτητής, ο Rasmus Lerdorf δημιούργησε την PHP χρησιμοποιώντας τη γλώσσα προγραμματισμού Perl ένα απλό script με όνομα php.cgi, για προσωπική χρήση. Το script αυτό είχε σαν σκοπό να διατηρεί μια λίστα στατιστικών για τα άτομα που έβλεπαν το online βιογραφικό του σημείωμα. Αργότερα αυτό το script το διέθεσε και σε φίλους του, οι οποίοι άρχισαν να του ζητούν να προσθέσει περισσότερες δυνατότητες. Η γλώσσα τότε ονομαζόταν PHP/FI από τα αρχικά Personal Home Page/ Form Interpreter. Το 1997 η PHP/FI έφθασε την έκδοση 2.0, βασιζόμενη αυτή τη φορά σε γλώσσα C και αριθμώντας περισσότερους από 50.000 ιστότοπους που τη χρησιμοποιούσαν, ενώ αργότερα την ίδια χρονιά οι Andi Gutmans και Zeev Suraski ξαναέγραψαν τη γλώσσα από την αρχή, περισσότερο στην σημερινή της μορφή. Έπειτα, οι Zeev και Andi δημιούργησαν την εταιρεία Zend (από τα αρχικά των ονομάτων τους), η οποία συνεχίζει μέχρι και σήμερα την ανάπτυξη και εξέλιξη της γλώσσας PHP. Το 1998, ακολούθησε η έκδοση 4 της PHP, τον Ιούλιο του 2004 διατέθηκε η έκδοση 5, ενώ αύτη τη στιγμή έχουν ήδη διατεθεί και οι πρώτες δοκιμαστικές εκδόσεις της

6.2 Χαρακτηριστικά της PHP

Ένα ιδιαίτερο χαρακτηριστικό της γλώσσας αυτής είναι ότι ο κώδικας της πρώτα μεταγλωττίζεται στον server και μετά φορτώνεται σαν ένα κανονικό html έγγραφο, χωρίς ο χρήστης να είναι σε θέση να δει τον αρχικό κώδικα. Έχει μηδενικό κόστος εγκατάστασης και επέκτασης με τρίτα προγράμματα διαχειριστικού περιβάλλοντος, όπως το MySQL. Η PHP προσφέρει τη δυνατότητα της σύνδεσης με μιας μεγάλης ποικιλίας από συστήματα Βάσεων Δεδομένων, όπως είναι η MySQL, η PostgreSQL, Oracle και πολλά άλλα.

Η λειτουργία του είναι αρχικά, ένα αρχείο που περιέχει κώδικα PHP και θα πρέπει να έχει και την κατάλληλη επέκταση (π.χ. *.php, *.php4, *.phtml κ.ά.). Αλλιώς η ενσωμάτωση κώδικα ενός αρχείου επέκτασης “.html” δεν θα λειτουργήσει και θα εμφανίσει στον browser τον κώδικα χωρίς καμία

επεξεργασία. Επίσης, ακόμη και όταν ένα αρχείο έχει την επέκταση “.php” θα πρέπει ο server να είναι ρυθμισμένος για να επεξεργάζεται κώδικα PHP.

Η λογική της διαδικασίας που εκτελούν τα αρχεία που περιέχουν κώδικα PHP, λέγεται server side. Από την στιγμή που ο χρήστης επιλέγει μια λειτουργία η οποία παραπέμπει σε εκτέλεση κώδικα php, ο κώδικας αλληλεπιδρά με τη βάση δεδομένων και επιστρέφει αποτελέσματα σε στατική σελίδα.

Η κατηγοριοποίηση στα server-side script είναι η εξής:

1. Εκτέλεση του περιεχομένου κώδικα σειριακά με ροή εκτέλεσης από πάνω προς τα κάτω και από τα αριστερά προς τα δεξιά
2. Αίτημα για απορρόφηση δεδομένων από μία βάση δεδομένων
3. Παραλαβή ή αποστολή δεδομένων
4. Τερματισμός της διαδικασίας με την επίδειξη αποτελεσμάτων σε html σελίδα

ΕΙΚΟΝΑ 2: HTTP://COMMONS.WIKIMEDIA.ORG/WIKI/FILE:PHP-LOGO.SVG

7. Αναγνώριση Χειριστών & Περιπτώσεων Χρήσης

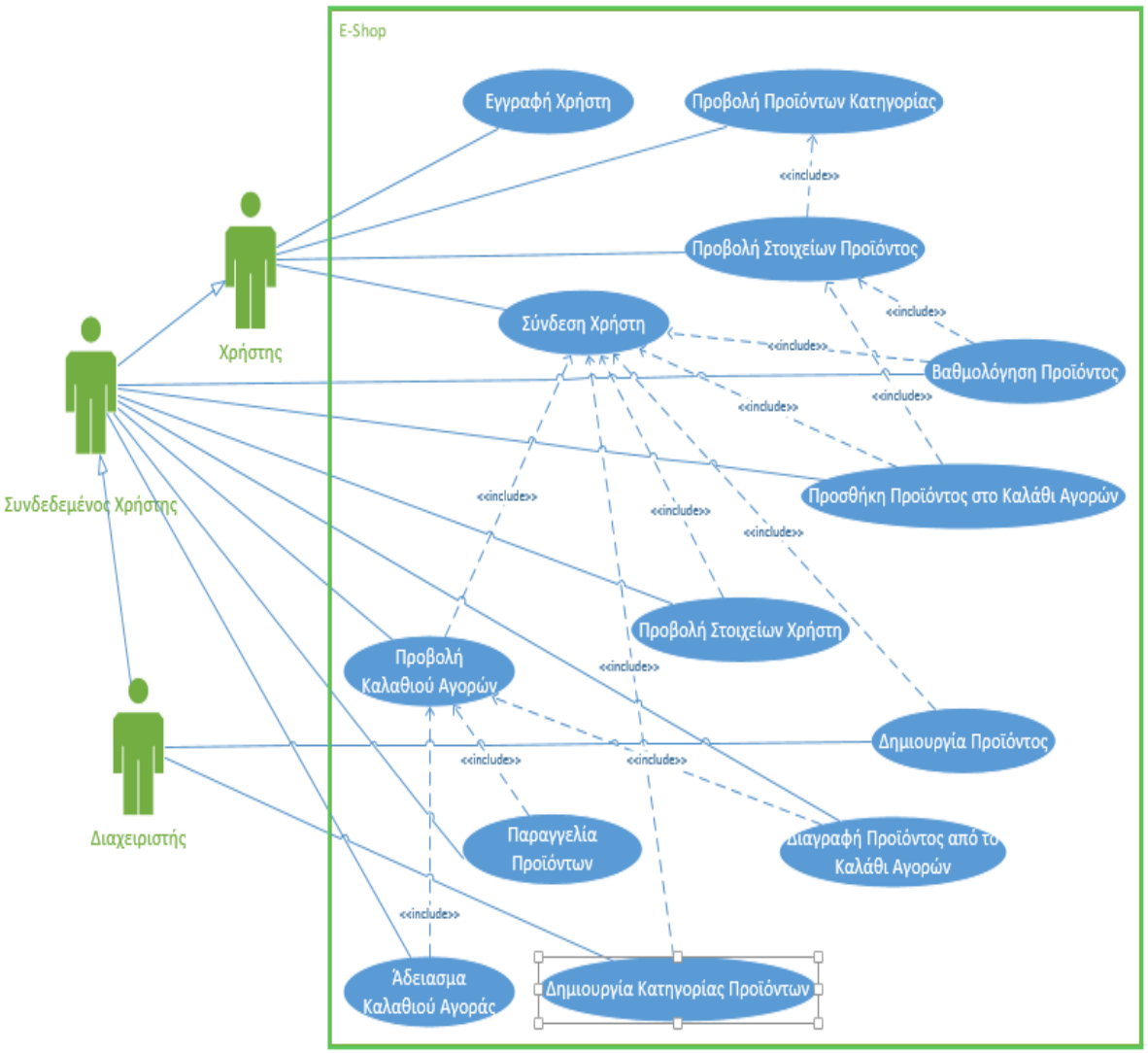
7.1. Αναγνώριση Χειριστών

- Πρωτεύοντες χειριστές
  - Διαχειριστής
  - Συνδεδεμένος χρήστης
  - Μη συνδεδεμένος χρήστης

7.2. Αναγνώριση Περιπτώσεων Χρήσεων

1. Εγγραφή Χρήστη
2. Σύνδεση Χρήστη
3. Προβολή Στοιχείων Χρήστη

4. Δημιουργία Κατηγορίας Προϊόντων
5. Δημιουργίας Προϊόντος
6. Προβολή Προϊόντων Κατηγορίας
7. Προβολή Στοιχείων Προϊόντος
8. Διαγραφή Προϊόντος
9. Βαθμολόγηση Προϊόντος
10. Προσθήκη Προϊόντος στο Καλάθι Αγορών
11. Προβολή Καλαθιού Αγορών
12. Άδειασμα Καλαθιού Αγορών
13. Διαγραφή Προϊόντος από το Καλάθι Αγορών
14. Παραγγελία Προϊόντων
15. Προβολή Λίστας Χρηστών
16. Προβολή Ιστορικού Αγορών



ΕΙΚΟΝΑ 2: ΔΙΑΓΡΑΜΜΑ ΠΕΡΙΠΤΩΣΕΩΝ ΧΡΗΣΗΣ

8. Περιγραφή Περιπτώσεων Χρήσης

1. Εγγραφή Χρήστη

Κύριο Σενάριο Επιτυχίας(ΚΣΕ):

1. Ο πελάτης συμπληρώνει τα προσωπικά του στοιχεία, ένα όνομα χρήστη(username) και το κωδικό πρόσβασης που επιθυμεί να έχει στην εφαρμογή.
2. Το σύστημα ελέγχει την εγκυρότητα των δεδομένων που εισήγαγε ο χρήστης
3. Το σύστημα επικυρώνει το λογαριασμό του χρήστη
4. Το σύστημα δημιουργεί το προσωπικό καλάθι αγορών του χρήστη

Επεκτάσεις

2α. Υπάρχουν δεδομένα που δεν έχουν συμπληρωθεί

1. Το Σύστημα εμφανίζει κατάλληλο μήνυμα λάθους
2. Επιστροφή στο βήμα 1 του ΚΣΕ

2β. Τα στοιχεία δεν έχουν έγκυρη μορφή

1. Το σύστημα εμφανίζει κατάλληλο μήνυμα λάθους
2. Επιστροφή στο βήμα 1 του ΚΣΕ

2γ. Ο κωδικό πρόσβασης δεν συμπίπτει με τον κωδικό επιβεβαίωσης

1. Το σύστημα εμφανίζει κατάλληλο μήνυμα λάθους
2. Επιστροφή στο βήμα 1 του ΚΣΕ

2δ. Το όνομα χρήστη δεν είναι διαθέσιμο

1. Το σύστημα εμφανίζει κατάλληλο μήνυμα λάθους
2. Επιστροφή στο βήμα 2 του ΚΣΕ

2. Σύνδεση Εγγεγραμμένου Χρήστη

Κύριο Σενάριο Επιτυχίας(ΚΣΕ):

1. Ο πελάτης εισάγει το όνομα χρήστη(username) και τον κωδικό πρόσβασης(password) που του αντιστοιχεί
2. Το σύστημα επιβεβαιώνει τα στοιχεία του χρήστη
3. Ο πελάτης εισάγεται στο σύστημα

Επεκτάσεις

1α. Ο χρήστης είναι ήδη συνδεδεμένος στο σύστημα

1. Επιστροφή στο βήμα 3 του ΚΣΕ

2α. Υπάρχουν δεδομένα που δεν έχουν συμπληρωθεί

1. Το σύστημα εμφανίζει κατάλληλο μήνυμα λάθους
2. Επιστροφή στο βήμα 1 του ΚΣΕ

2β. Το όνομα χρήστη και ο κωδικός πρόσβασης δεν συμπίπτουν

1. Το σύστημα εμφανίζει κατάλληλο μήνυμα λάθους
2. Επιστροφή στο βήμα 1 του ΚΣΕ

3. Προβολή Στοιχείων Χρήστη

Κύριο Σενάριο Επιτυχίας(ΚΣΕ):

1. Το σύστημα επιβεβαιώνει ότι ο χρήστης είναι συνδεδεμένος στο σύστημα.
2. Το σύστημα προβάλλει τα στοιχεία του χρήστη

Επεκτάσεις

1α. Ο χρήστης δεν είναι συνδεδεμένος στο σύστημα

1. Σύνδεση Χρήστη

4. Δημιουργία Κατηγορίας Προϊόντων

Κύριο Σενάριο Επιτυχίας(ΚΣΕ):

1. Το σύστημα επιβεβαιώνει ότι ο χρήστης είναι διαχειριστής του συστήματος
2. Ο χρήστης πληκτρολογεί το όνομα της κατηγορίας
3. Ο χρήστης εισάγει την φωτογραφίας της κατηγορίας
4. Το σύστημα δημιουργεί τη νέα κατηγορία προϊόντων

Επεκτάσεις

1α. Ο χρήστης δεν είναι συνδεδεμένος στο σύστημα

1. Σύνδεση Χρήστη

1β. Ο χρήστης δεν είναι διαχειριστής του συστήματος

1. Το σύστημα εμφανίζει κατάλληλο μήνυμα λάθους

2α. Το όνομα της κατηγορίας δεν έχει συμπληρωθεί

1. Το σύστημα εμφανίζει κατάλληλο μήνυμα λάθους
2. Επιστροφή στο βήμα 2 του ΚΣΕ

2β. Το όνομα της κατηγορίας δεν είναι διαθέσιμο

1. Το σύστημα εμφανίζει κατάλληλο μήνυμα λάθους
2. Επιστροφή στο βήμα 2 του ΚΣΕ

3α. Δεν εισήχθη η φωτογραφία της κατηγορίας

1. Το σύστημα εμφανίζει κατάλληλο μήνυμα λάθους
2. Επιστροφή στο βήμα 3 του ΚΣΕ

5. Δημιουργία Προϊόντος

Κύριο Σενάριο Επιτυχίας(ΚΣΕ):

1. Το σύστημα επιβεβαιώνει ότι ο χρήστης είναι διαχειριστής του συστήματος
2. Ο χρήστης πληκτρολογεί το όνομα του προϊόντος
3. Ο χρήστης επιλέγει την κατηγορία που ανήκει το προϊόν
4. Ο χρήστης εισάγει τη φωτογραφία του προϊόντος
5. Ο χρήστης εισάγει τη τιμή του προϊόντος
6. Το σύστημα καταχωρεί το προϊόν

Επεκτάσεις

1α. Ο χρήστης δεν είναι συνδεδεμένος στο σύστημα

1. Σύνδεση Χρήστη

1β. Ο χρήστης δεν είναι διαχειριστής του συστήματος

1. Το σύστημα εμφανίζει κατάλληλο μήνυμα λάθους

2α. Το όνομα του προϊόντος δεν έχει συμπληρωθεί

1. Το μήνυμα εμφανίζει κατάλληλο μήνυμα λάθους
2. Επιστροφή στο βήμα 2 του ΚΣΕ

2α. Το όνομα του προϊόντος δεν είναι διαθέσιμο

1. Το σύστημα εμφανίζει κατάλληλο μήνυμα λάθους
2. Επιστροφή στο βήμα 2 του ΚΣΕ

4α. Δεν εισήχθη η φωτογραφία του προϊόντος

1. Το σύστημα εμφανίζει κατάλληλο μήνυμα λάθους
2. Επιστροφή στο βήμα 4 του ΚΣΕ

5α. Δεν εισήχθη η τιμή του προϊόντος

1. Το σύστημα εμφανίζει κατάλληλο μήνυμα λάθους
2. Επιστροφή στο βήμα 5 του ΚΣΕ

5β. Η τιμή εισόδου που δόθηκε δεν είναι έγκυρη

1. Το σύστημα εμφανίζει κατάλληλο μήνυμα λάθους
2. Επιστροφή στο βήμα 5 του ΚΣΕ

6. Προβολή Προϊόντων Κατηγορίας

Κύριο Σενάριο Επιτυχίας(ΚΣΕ):

1. Το σύστημα εμφανίζει μια λίστα με τις διαθέσιμες κατηγορίες προϊόντων
2. Ο χρήστης επιλέγει την επιθυμητή κατηγορία προϊόντων
3. Το σύστημα εμφανίζει μια λίστα με 3 προτεινόμενα προς το χρήστη προϊόντα
4. Το σύστημα εμφανίζει τα προϊόντα της εφαρμογής που ανήκουν στην επιλεγμένη κατηγορία

Επεκτάσεις

1α. Δεν υπάρχουν διαθέσιμες κατηγορίες προϊόντος

1. Το σύστημα εμφανίζει κατάλληλο μήνυμα

3α. Ο χρήστης δεν είναι συνδεδεμένος

1. Επιστροφή στο βήμα 4 του ΚΣΕ

3β. Δεν υπάρχουν προϊόντα στη κατηγορία αυτή

1. Το σύστημα εμφανίζει κατάλληλο μήνυμα

7. Προβολή Στοιχείων Προϊόντος

Κύριο Σενάριο Επιτυχίας(ΚΣΕ):

1. Προβολή Προϊόντων Κατηγορίας
2. Ο χρήστης επιλέγει το επιθυμητό προϊόν
3. Το σύστημα προβάλλει τα στοιχεία του επιλεγμένου προϊόντος
4. Το σύστημα εμφανίζει την υφιστάμενη βαθμολογία προϊόντος

Επεκτάσεις

4α. Ο χρήστη δεν είναι συνδεδεμένος

1. Τέλος σεναρίου

4β. Ο χρήστης δεν έχει βαθμολογήσει το προϊόν

1. Η βαθμολογία του προϊόντος είναι κενή

8. Διαγραφή Προϊόντος

Κύριο Σενάριο Επιτυχίας(ΚΣΕ):

1. Προβολή Στοιχείων Προϊόντος
2. Ο διαχειριστής επιλέγει τη διαγραφή του προϊόντος
3. Το σύστημα διαγράφει το προϊόν

Επεκτάσεις

2α. Ο χρήστης δεν είναι διαχειριστής του συστήματος

1. Το σύστημα εμφανίζει κατάλληλο μήνυμα λάθους
2. Τέλος σεναρίου

9. Βαθμολόγηση Προϊόντος

Κύριο Σενάριο Επιτυχίας(ΚΣΕ):

1. Προβολή Στοιχείων Προϊόντος
2. Ο χρήστης επιλέγει την τιμή της βαθμολογίας του προϊόντος

Επεκτάσεις

2α. Ο χρήστης δεν είναι συνδεδεμένος

1. Σύνδεση Χρήστη

10. Προσθήκη Προϊόντος στο Καλάθι Αγορών

Κύριο Σενάριο Επιτυχίας(ΚΣΕ):

1. Προβολή Στοιχείων Προϊόντος
2. Ο χρήστης επιλέγει την επιθυμητή ποσότητα
3. Το σύστημα εισάγει το προϊόν στο καλάθι αγορών του χρήστη

Επεκτάσεις

2α. Ο χρήστης δεν είναι συνδεδεμένος στο σύστημα

1. Σύνδεση Χρήστη

2β. Ο χρήστης δεν εισήγαγε την επιθυμητή ποσότητα

1. Το σύστημα εμφανίζει κατάλληλο μήνυμα λάθους
2. Επιστροφή στο βήμα 2 του ΚΣΕ

2γ. Η δοσμένη ποσότητα δεν είναι έγκυρη

1. Το σύστημα εμφανίζει κατάλληλο μήνυμα λάθους
2. Επιστροφή στο βήμα 2 του ΚΣΕ

11. Προβολή Καλαθιού Αγορών

Κύριο Σενάριο Επιτυχίας(ΚΣΕ):

1. Το σύστημα επιβεβαιώνει ότι ο χρήστης είναι συνδεδεμένος στο σύστημα
2. Το σύστημα προβάλλει τις πληροφορίες των προϊόντων που βρίσκονται στο καλάθι του χρήστη
3. Το σύστημα προβάλλει το συνολικό κόστος των προϊόντων που βρίσκονται στο καλάθι των αγορών στις επιλεγμένες ποσότητες

Επεκτάσεις

1α. Ο χρήστης δεν είναι συνδεδεμένος στο σύστημα

1. Σύνδεση Χρήστη

2α. Το καλάθι των αγορών του χρήστη είναι άδειο

1. Το σύστημα εμφανίζει κατάλληλο μήνυμα

12. Άδειασμα Καλαθιού Αγοράς

Κύριο Σενάριο Επιτυχίας(ΚΣΕ):

1. Προβολή Καλαθιού Αγορών
2. Ο χρήστης επιλέγει το άδειασμα του καλαθιού αγορών
3. Το σύστημα αδειάζει το καλάθι αγορών του χρήστη και ενημερώνει το χρήστη με κατάλληλο μήνυμα

13. Διαγραφή Προϊόντος από το Καλάθι Αγορών

Κύριο Σενάριο Επιτυχίας(ΚΣΕ):

1. Προβολή Καλαθιού Αγορών
2. Ο χρήστης επιλέγει το επιθυμητό προϊόν
3. Το σύστημα διαγράφει το προϊόν από το καλάθι αγορών και ανανεώνει τη συνολική τιμή βάση των εναπομεινάντων προϊόντων στο καλάθι αγορών

Επεκτάσεις

3α. Το καλάθι αγορών είναι άδειο

1. Το σύστημα εμφανίζει κατάλληλο μήνυμα

14. Παραγγελία Προϊόντων

Κύριο Σενάριο Επιτυχίας(ΚΣΕ):

1. Προβολή Καλαθιού Αγορών
2. Ο χρήστης επιλέγει την επιλογή ολοκλήρωσης παραγγελίας
3. Ο χρήστης συμπληρώνει τον αριθμό της πιστωτικής του κάρτας
4. Ο χρήστης συμπληρώνει τη διεύθυνση αποστολής
5. Το σύστημα αδειάζει το καλάθι αγορών και αποθηκεύει τα δεδομένα της παραγγελίας

Επεκτάσεις

3α. Ο αριθμός πιστωτικής κάρτας δεν έχει συμπληρωθεί

1. Το σύστημα εμφανίζει κατάλληλο μήνυμα λάθους
2. Επιστροφή στο βήμα 3 του ΚΣΕ

4α. Η διεύθυνση αποστολής δεν έχει συμπληρωθεί

1. Το σύστημα εμφανίζει κατάλληλο μήνυμα λάθους
2. Επιστροφή στο βήμα 4 του ΚΣΕ

15. Προβολή Λίστας Χρηστών

Κύριο Σενάριο Επιτυχίας(ΚΣΕ):

1. Το σύστημα επιβεβαιώνει ότι ο χρήστης είναι διαχειριστής του συστήματος
2. Το σύστημα προβάλει τη λίστα των χρηστών της εφαρμογής

Επεκτάσεις

1α. Ο χρήστης δεν είναι συνδεδεμένος στο σύστημα

1. Σύνδεση Χρήστη

1β. Ο χρήστης δεν είναι διαχειριστής του συστήματος

2. Το σύστημα εμφανίζει κατάλληλο μήνυμα λάθους

16. Προβολή Ιστορικού Αγορών

Κύριο Σενάριο Επιτυχίας(ΚΣΕ):

1. Το σύστημα επιβεβαιώνει ότι ο χρήστης είναι συνδεδεμένος
2. Το σύστημα προβάλλει τις αγορές που έχει κάνει ο χρήστης στο παρελθόν
3. Ο χρήστης επιλέγει την επιθυμητή αγορά
4. Το σύστημα προβάλει τις πληροφορίες που αφορούν την επιλεγμένη αγορά

Επεκτάσεις

1α. Ο χρήστης δεν είναι συνδεδεμένος στο σύστημα

1. Σύνδεση Χρήστη

2α. Ο χρήστης δεν έχει κάνει αγορές στο παρελθόν

1. Το σύστημα του εμφανίζει κατάλληλο μήνυμα
2. Τέλος σεναρίου

9. Προσωποποιημένη Διεπαφή Χρήστη

Στο σύστημα υπάρχουν δύο τύποι πελατών, οι μη συνδεδεμένοι χρήστες και οι συνδεδεμένοι χρήστες. Η εφαρμογή προσφέρει στους πελάτες μια προσωποποιημένη διεπαφή, ανάλογα με τον τύπου αυτών.

Υπάρχει ένας Recommender, ο οποίος ανάλογα με τον τύπο πελάτη εφαρμόζει διαφορετική πολιτική συστάσεων σχετικά με τα προτεινόμενα προϊόντα. Πιο συγκεκριμένα, αφού ο πελάτης επιλέξει την κατηγορία των προϊόντων, για την οποία επιθυμεί να προβληθούν τα προϊόντα της, στη συνέχεια θα παρουσιαστεί η λίστα με τα προϊόντα, πάνω από την οποία θα προβληθούν τα τρία προτεινόμενα προϊόντα. Τα προτεινόμενα προϊόντα, αποτελούνται από προϊόντα της κατηγορίας αυτής, τα οποία δεν έχει αγοράσει στο παρελθόν ο πελάτης. Υπάρχουν δύο τύποι αλγόριθμοι συστάσεων:

- Απλός Recommender
- Weighted Slope One Recommender

Ο απλός Recommender, χρησιμοποιείται για τη σύσταση προτάσεων στους μη συνδεδεμένους χρήστες. Οι μη συνδεδεμένοι χρήστες είναι πελάτες, οι οποίοι δεν έχουν λογαριασμό στο σύστημα. Ωστόσο το γεγονός ότι δεν έχουν λογαριασμό στην εφαρμογή αυτόματα σημαίνει πως η εφαρμογή δεν έχει πληροφορίες για τους χρήστες αυτούς. Για το λόγο αυτό, η σύσταση προτάσεων δεν μπορεί να προσωποποιημένοι για αυτούς τους χρήστες,

καθώς δεν υπάρχει γνώση για τις αγοραστικές τους συνήθειες. Για το λόγο αυτό, ο τρόπος σύστασης του απλού Recommender στηρίζεται στην σύσταση των 5 κορυφαίων προϊόντων σε πωλήσεις της κατηγορίας.

Αντίθετα για τους συνδεδεμένους χρήστες χρησιμοποιείται ο Weighted Slope One αλγόριθμος για τη σύσταση προτάσεων. Η Weighted Slope One [4] είναι μια δημοφιλής μέθοδος συνεργατικού φιλτραρίσματος βασισμένη στα αντικείμενα. Οι προσεγγίσεις που βασίζονται στο συνεργατικό φιλτράρισμα οικοδομούν ένα μοντέλο βάσει της παρελθοντικής συμπεριφοράς του χρήστη (π.χ. αντικείμενα που έχουν αποκτηθεί στο παρελθόν και βαθμολογήσεις που έχει δώσει) καθώς και παρόμοιων αποφάσεων που έχουν παρθεί από άλλους χρήστες, και βάσει του μοντέλου αυτού αξιολογούν αντικείμενα και προβλέπουν ποια από αυτά είναι πιθανότερο να ενδιαφέρουν τον χρήστη. Η Weighted Slope One είναι μια επέκταση της κλασσικής Slope One, η οποία βασίζεται τόσο στην έννοια της διαφορικής δημοτικότητας μεταξύ των αντικειμένων, δηλαδή πόσο περισσότερο προτιμάται ένα αντικείμενο σε σχέση με ένα άλλο, όσο και στη στάθμιση των προβλέψεων, βάση του αριθμού των χρηστών που βαθμολόγησαν ένα αντικείμενο. Το πλεονέκτημα της μεθόδου αυτής είναι ότι έχει σχετικά μικρή πολυπλοκότητα υπολογισμών, ενώ ταυτόχρονα η ορθότητα της βρίσκεται στο ίδιο επίπεδο με έναν πιο πολύπλοκο και δαπανηρό υπολογιστικά αλγόριθμο. Η ιδιότητα αυτή αποτελεί ένα μεγάλο πλεονέκτημα του WSO, καθώς το μεγαλύτερο πρόβλημα που παρατηρείται στους αλγορίθμους σύστασης προτάσεων είναι ότι η ο χρόνος που απαιτείται για τη σύσταση προτάσεων αυξάνεται με γοργό ρυθμό, όσο το σύνολο των δεδομένων(χρηστών, αντικειμένων, βαθμολογιών) μεγαλώνει.

9.1 Περιγραφή WSO

Ο υπολογισμός των προβλέψεων με βάση τη μέθοδο αυτή χωρίζεται σε δύο βήματα.

Βήμα 1: Υπολογισμός Διαφορικής Απόκλισης

Στη πρώτη φάση θα υπολογιστεί η διαφορική απόκλιση μεταξύ κάθε ζεύγους αντικειμένων. Έστω U και S το σύνολο των χρηστών και των αντικειμένων του συστήματος αντίστοιχα. Ορίζουμε ως $r_{u,i}$ την βαθμολογία που έχει δώσει ο χρήστης *u* στο αντικείμενο *i*. Επιπρόσθετα ορίζουμε ως $S_{i,j}$ το σύνολο των χρηστών που έχουν βαθμολογήσει τόσο το αντικείμενο *i* όσο και το αντικείμενο *j* και ως $|S_{i,j}|$ τον αριθμό των στοιχείων του συνόλου αυτού. Για τον υπολογισμό της απόκλισης του αντικειμένου *i* ως προς το αντικείμενο *j,* χρησιμοποιείται ο ακόλουθος τύπος:

$$dev_{i,j} = \sum_{u \in S_{i,j}} \frac{(r_{u,i} - r_{u,j})}{|S_{i,j}|}$$

Βήμα 2: Πρόβλεψη Βαθμολογιών

Στη δεύτερη φάση γίνονται οι προβλέψεις βάσει των αποκλίσεων που έχουν υπολογιστεί στο Βήμα 1. Σκοπός είναι να οριστεί ο τύπος υπολογισμού της πρόβλεψης που θα γίνει πάνω σε ένα αντικείμενο *i* όσον αφορά ένα χρήστη *u*. Έστω R_u το σύνολο των αντικειμένων που έχουν βαθμολογηθεί από τον χρήστη *u*. Για τον υπολογισμό της πρόβλεψης μιας βαθμολογίας θα χρησιμοποιηθεί ο ακόλουθος τύπος:

$$p_{u,i} = \frac{\sum_{j \in R_u}(dev_{i,j}+r_{u,j})*|S_{i,j}|}{\sum_{j \in R_u}|S_{i,j}|}$$

Το δεύτερο βήμα επαναλαμβάνεται για κάθε αντικείμενο που δεν έχει αγοράσει ή βαθμολογήσει ο χρήστης στο παρελθόν και ανήκει στην επιλεγμένη κατηγορία προϊόντων. Αφού υπολογιστεί για κάθε ένα από τα προαναφερθέν προϊόντα η εκτιμώμενη βαθμολογία, τότε ο αλγόριθμος σύστασης προτάσεων θα ταξινομήσει τα προϊόντα με βάση την εκτιμώμενη βαθμολογία και εν τέλει θα προτείνει στο χρήστη τα κορυφαία 3 προϊόντα για αυτόν.

Είναι σημαντικό να τονιστεί, πως η Weighted Slope One βασίζεται στις βαθμολογίες που έχουν δώσει οι χρήστες και επομένως, προκειμένου να παρέχεται σύσταση από τον Recommender θα πρέπει:

α) να υπάρχουν προϊόντα αυτής της κατηγορίας που έχουν αξιολογηθεί από άλλους χρήστες και ταυτόχρονα δεν έχουν αγοραστεί στο παρελθόν από το χρήστη στον οποίο γίνεται η σύσταση προτάσεων

β) να έχει βαθμολογήσει ο χρήστης στον οποίον γίνεται η σύσταση προτάσεων στο παρελθόν προϊόντα, προκειμένου να είναι δυνατός ο υπολογισμός της ομοιότητας που εμφανίζουν τα προϊόντα μεταξύ τους. Με τον τρόπο αυτό δίνετε κίνητρο στους χρήστες της εφαρμογής να βαθμολογούν προϊόντα που έχουν αγοράσει.

Σε περίπτωση που η WSO δεν επιστρέψει αποτελέσματα, επειδή δεν συμβαίνει ένα από τα παραπάνω, τότε το σύστημα θα προτείνει τα 3 πιο δημοφιλή προϊόντα της κατηγορίας στο χρήστη, τα οποία δεν έχει αγοράσει στο παρελθόν.

10. Profiler

Η εφαρμογή έχει τη δυνατότητα να παρακολουθεί τις κινήσεις των χρηστών και να δημιουργεί ένα αντίστοιχο προφίλ για αυτούς. Πιο συγκεκριμένα, ο profiler προσπαθεί να προβλέψει τις εξής πληροφορίες:

- Οικονομική κατάσταση
- Γονεϊκή κατάσταση
- Χορτοφαγία
- Ηλικία
- Φύλο
- Χώρα διαμονής

Για την εύρεση της χώρας διαμονής, χρησιμοποιείται η εύρεση των συντεταγμένων του υπολογιστή του χρήστη. Όσον αφορά τον εντοπισμό των υπόλοιπων στοιχείων, χρησιμοποιείται ένα σύνολο από κανόνες. Οι κανόνες αυτοί θα εκτελεστούν στο σύνολο των αγορών που έχει κάνει ο χρήστης στο παρελθόν.

10.1 Οικονομική Κατάσταση

Στόχος του profiler είναι να προβλέψει αν η οικονομική κατάσταση του χρήστη είναι καλή ή όχι. Για τον εντοπισμό αυτής της πληροφορίας χρησιμοποιούνται οι ακόλουθοι 7 κανόνες:

1. Ύπαρξη αγοράς αξίας μεγαλύτερη των 150 ευρώ → **Καλή οικονομική κατάσταση**
2. Αγορά προϊόντων κινέζικης κουζίνας → **Καλή οικονομική κατάσταση**
3. Αγορά των πιο ακριβών προϊόντων ανά κατηγορία → **Καλή οικονομική κατάσταση**
4. Μεγάλη ποσότητα κρεάτων → **Καλή οικονομική κατάσταση**
5. Μεγάλη ποσότητα τυριών → **Καλή οικονομική κατάσταση**
6. Μεγάλη ποσότητα ξηρών καρπών → **Καλή οικονομική κατάσταση**
7. Αγορά των πιο φθηνών προϊόντων ανά κατηγορία → **Κακή οικονομική κατάσταση**

10.2 Γονεϊκή Κατάσταση

Στόχος είναι η εξαγωγή της πληροφορίας ύπαρξης ή όχι παιδιών. Για την εύρεσης αυτής της πληροφορίας χρησιμοποιήθηκαν οι ακόλουθοι 3 κανόνες:

1. Αγορά παιδιών προϊόντων → **Είναι γονέας**
2. Αγορά παιδιών τροφών → **Είναι γονέας**
3. Αγορά μεγάλης ποσότητας γάλατος → **Είναι γονέας**

Σε περίπτωση που δεν ικανοποιείται κανένας από τους προαναφερθείς τρεις κανόνες, τότε ο χρήστης θεωρείται πως δεν έχει παιδιά.

10.3 Χορτοφαγία

Ο profiler προσπαθεί να καταλάβει αν ο χρήστης είναι χορτοφάγος ή όχι. Για την εύρεση αυτής της πληροφορίας χρησιμοποιεί έναν απλό και συνάμα ισχυρό κανόνα. Σε περίπτωση που σε προηγούμενη αγορά ο χρήστης έχει αγοράσει φαγητά, ωστόσο κανένα από τα φαγώσιμα είδη δεν είναι κρεατικό(τόσο της ίδιας αγοράς όσο και άλλης), τότε ο χρήστης είναι χορτοφάγος. Σε περίπτωση που στο παρελθόν έχει αγοράσει κρεατικά προϊόντα, τότε ο χρήστης δεν είναι χορτοφάγος. Σε οποιαδήποτε άλλη περίπτωση, ο profiler δεν μπορεί να αποφασίσει αν ο χρήστης είναι χορτοφάγος ή όχι.

10.4 Ηλικία

Ο profiler προσπαθεί να προβλέψει το ελάχιστο όριο ηλικίας του χρήστη. Για τον υπολογισμό του ελάχιστου ορίου ηλικίας χρησιμοποιούνται οι ακόλουθοι 15 κανόνες:

1. Αγορά ποτών → ≥ **16 ετών**
2. Αγορά προϊόντων από τα ανδρικά είδη → ≥ **16 ετών**
3. Αγορά είδη κουζίνας → ≥ **18 ετών**
4. Αγορά καθαριστικών ειδών → ≥ **18 ετών**
5. Αγορά όσπριων ή λαχανικών → ≥ **18 ετών**
6. Αγορά προϊόντων αυτοκινήτου → ≥ **18 ετών**
7. Αγορά ερωτικών εξαρτημάτων → ≥ **18 ετών**
8. Αγορά κατεψυγμένων παρασκευασμάτων → ≥ **18 ετών**
9. Μεγάλη ποσότητα ξηρών καρπών → ≥ **18 ετών**

10. Αγορά διακοσμητικών σπιτιού → ≥ **20 ετών**
11. Αγορά σνακ και σοκολατοειδή → ≥ **25 ετών**
12. Αγορά μεγάλης ποσότητας γάλατος → ≥ **25 ετών**
13. Αγορά παιδικών ειδών → ≥ **25 ετών**
14. Αγορά παιδικών τροφών → ≥ **25 ετών**
15. Αγορά προϊόντων για ηλικιωμένους → ≥ **70 ετών**

Σε περίπτωση που από τη χρήση των παραπάνω κανόνων εξαχθούν δύο διαφορετικά ελάχιστα όρια, τότε θα κρατήσουμε το μεγαλύτερο ελάχιστο όριο, καθώς είναι πιο αυστηρό όριο και συνάμα δεν αναιρεί το μικρότερο.

10.5 Φύλο

Η τελευταία πληροφορίας που προσπαθεί να εκμαιεύσει ο profiler είναι το φύλο του χρήστη(άνδρας ή γυναίκα). Για την εύρεση αυτής της πληροφορίας χρησιμοποιούνται οι ακόλουθοι 8 κανόνες:

1. Αγορά γυναικείων ειδών → **Γυναίκα**
2. Αγορά είδη κουζίνας → **Γυναίκα**
3. Αγορά διακοσμητικών σπιτιού → **Γυναίκα**
4. Αγορά καλλυντικών → **Γυναίκα**
5. Αγορά είδη περιποίησης → **Γυναίκα**
6. Αγορά αντρικών ειδών → **Άνδρας**
7. Αγορά κατεψυγμένων προϊόντων → **Άνδρας**
8. Αγορά προϊόντων αυτοκινήτου → **Άνδρας**

11. Μοντέλο Σχεδίασης Εφαρμογής

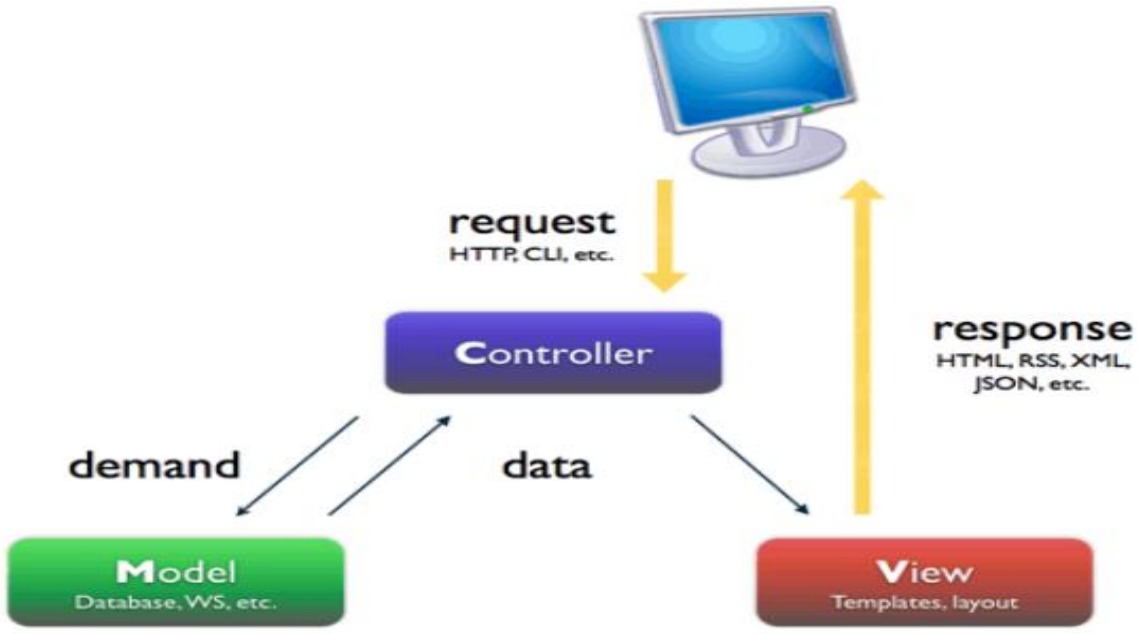
Το μοντέλο σχεδίασης πάνω στο οποίο βασίστηκε η εφαρμογή είναι το MVC. Το MVC(Model-View-Controller) Το MVC είναι ένα πολύ δημοφιλές πρότυπο σχεδίασης λογισμικού για την ανάπτυξη εφαρμογών στο Web. Βάση του MVC η εφαρμογή χωρίζεται σε τρία λογικά επίπεδα:

- Model: Είναι το χαμηλότερο επίπεδο του προτύπου, το οποίο είναι υπεύθυνο για τη διατήρηση των δεδομένων. Οποιαδήποτε αλληλεπίδραση με τη βάση δεδομένων της εφαρμογής υποστηρίζεται από το model.

- View: Αποτελεί το κομμάτι της εφαρμογής που βλέπει ο χρήστης και είναι υπεύθυνο για την παρουσίαση των διεπαφών στην οθόνη του χρήστη.

- Controller: Δέχονται όλες τις αιτήσεις του χρήστη και τις επεξεργάζονται, χρησιμοποιώντας το κατάλληλο Model και παρουσιάζοντας το κατάλληλο View για να τις ικανοποιήσουν. Οι Controllers ελέγχουν την ροή της εφαρμογής και είναι ο συνδετικός κρίκος μεταξύ Models και Views.

Το MVC έγινε πολύ δημοφιλές γιατί μέσω του διαχωρισμού της εφαρμογής σε τρία επίπεδα καταφέρνει να περιορίσει τις εξαρτήσεις μεταξύ των διαφορετικών τμημάτων του κώδικα και εν τέλει να απομονώσει τη λογική της εφαρμογής από το επίπεδο διεπαφής με το χρήστη. Η ιδιότητα αυτή διευκολύνει τη συγγραφή του κώδικα της εφαρμογής, καθώς μπορεί εύκολα να παραλληλοποιηθεί σε πολλαπλούς ομάδες προγραμματιστών έχοντας από ελάχιστες έως και μηδαμινές εξαρτήσεις. Η ιδιότητα αυτή είναι ιδιαίτερα χρήσιμη σε μεγάλες και απαιτητικές εφαρμογές. Επιπρόσθετα το MVC μοντέλο διευκολύνει τη συντήρηση του κώδικας μιας εφαρμογής, κάνοντας πιο εύκολη την επέκταση των λειτουργιών αυτής στο μέλλον.



ΕΙΚΟΝΑ 3:ΜΟΝΤΕΛΟ MVC

Σύμφωνα με το MVC, ο controller λαμβάνει όλα τα αιτήματα για την εφαρμογή και στη συνέχεια σε συνεργασία με το model ανακτά και επεξεργάζεται όλα τα απαραίτητα δεδομένα που θα χρειαστούν για την απόκριση στην αίτηση του χρήστη. Στη συνέχεια ο controller εφοδιάζει το view με τα δεδομένα αυτά. Στο σημείο αυτό το view είναι έτοιμο να εμφανίσει στο χρήστη την απάντηση στο

αίτημα του με έναν ευπαρουσίαστο τρόπο. Η λογική ροή του MVC Προτύπου παρουσιάζεται στην παραπάνω εικόνα.

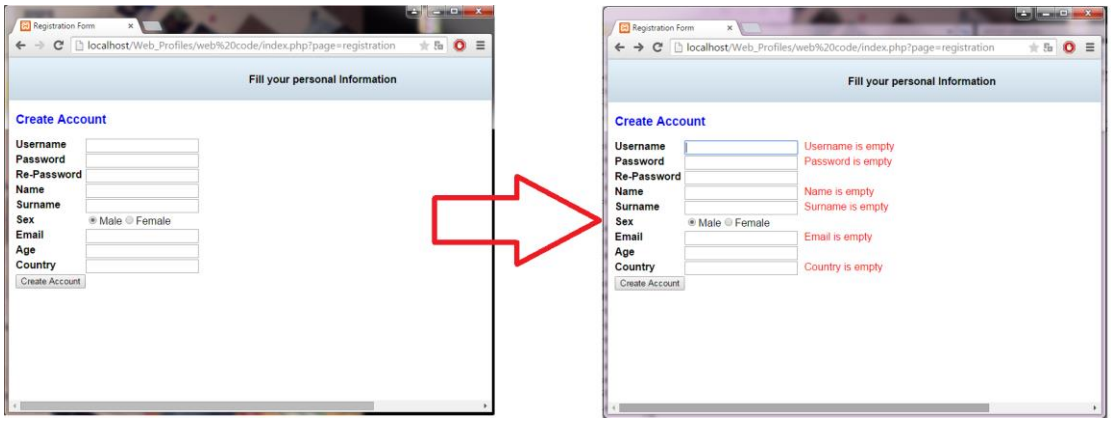
12. Διεπαφή Χρήστη

Στο κεφάλαιο αυτό παρουσιάζονται μέσα από εικόνες ορισμένες από τις επιλογές που δίνονται στο χρήστη ανάλογα με τα δικαιώματα που έχει στην εφαρμογή.

12.1 Εγγραφή νέου χρήστη

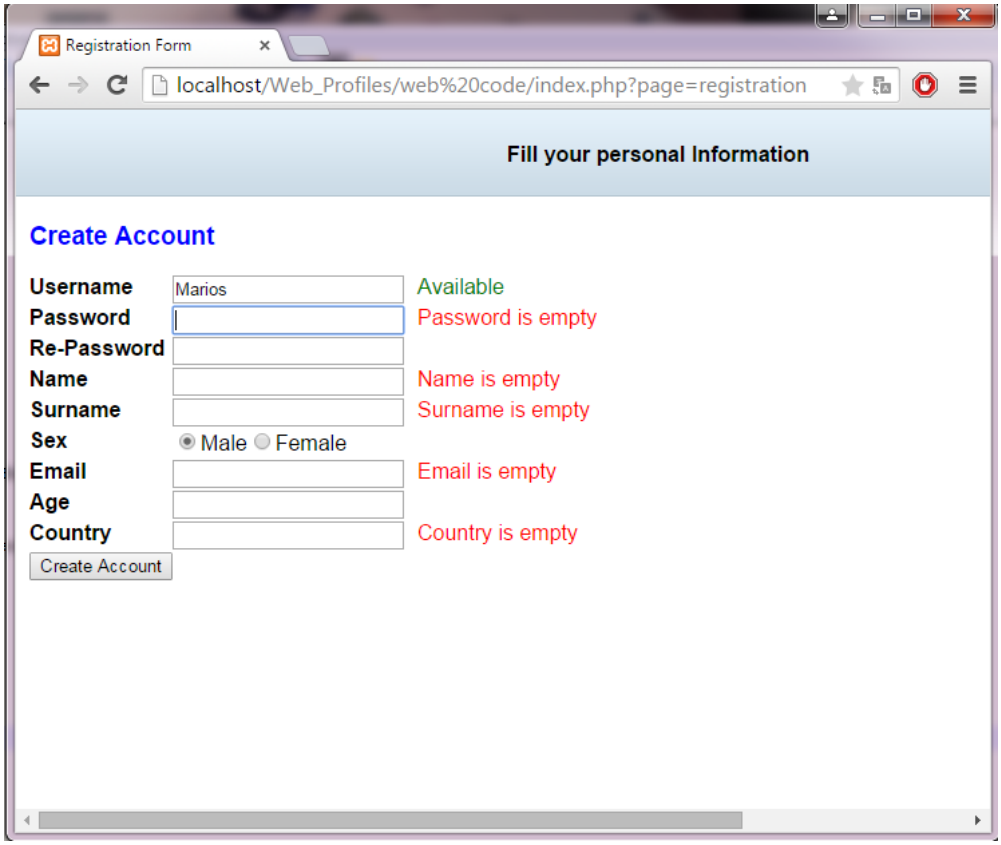
Αρχικά ένας νέος χρήστης θα πρέπει να συμπληρώσει τα στοιχεία που απαιτούνται προκειμένου να καταχωρηθεί στη βάση. Παρακάτω παρουσιάζονται οι διάφορες περιπτώσεις λαθών κατά την δήλωση των στοιχείων του στη φόρμα “registration” καθώς και η επιτυχής εγγραφή του εν λόγω χρήστη.

Σε περίπτωση που ο χρήστης δεν συμπληρώσει κανένα από τα απαιτούμενα στοιχεία και προβεί σε δημιουργία λογαριασμού (create account), τότε εμφανίζεται κατάλληλο μήνυμα λάθους, όπως φαίνεται και στην παρακάτω εικόνα.

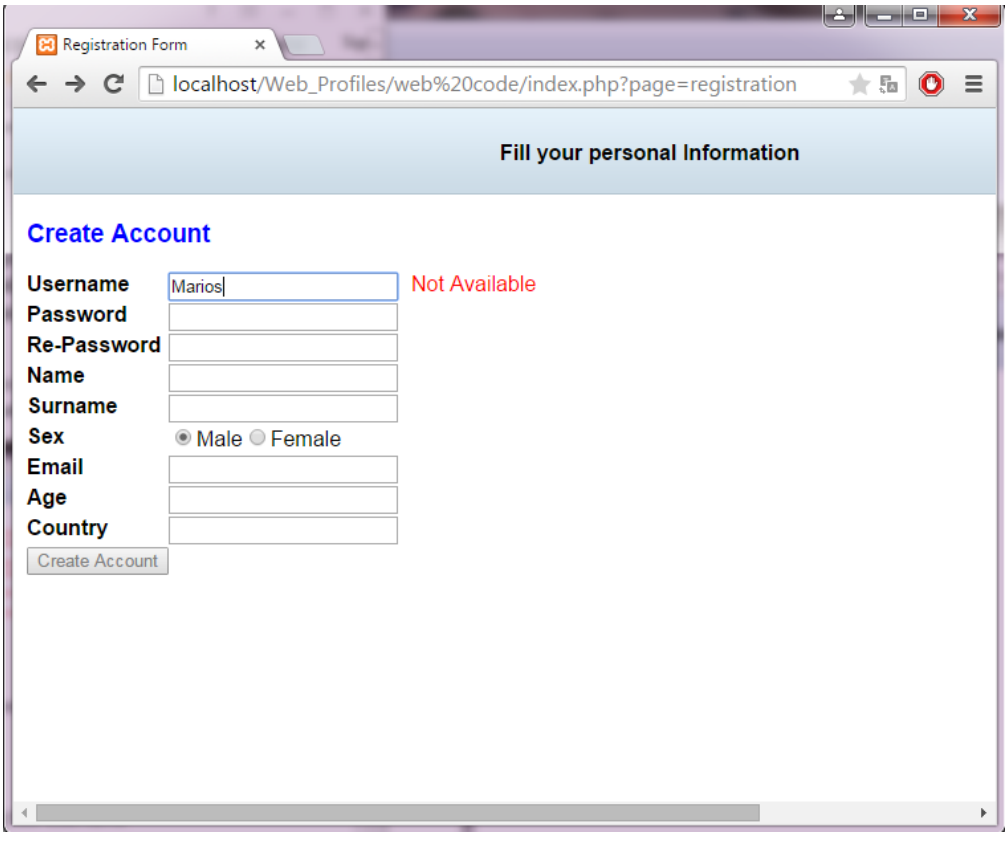


Εικόνα 4: Άδεια δήλωση στοιχείων

Άλλο ένα χαρακτηριστικό της βάσης είναι ο έλεγχος διαθεσιμότητας του username.Ακολουθούν δύο φωτογραφίες, όπου στην πρώτη είναι διαθέσιμο το username που πληκτρολογείτε, ενώ στη δεύτερη, αφού έχει προηγηθεί η εγγραφή του χρήστη “Marios”, να ενημερώνεται ο χρήστης για τη μη διαθεσιμότητα του username.

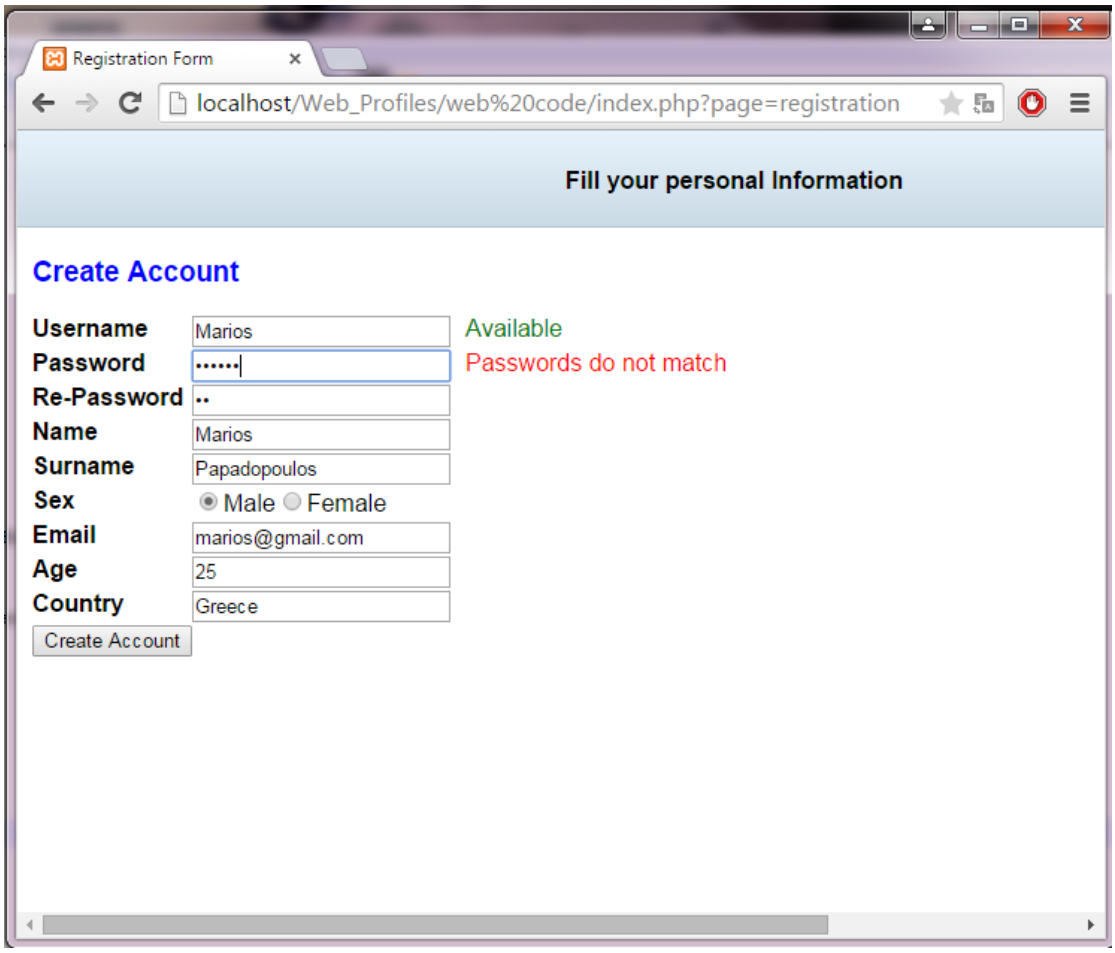


Εικόνα 5: Διαθέσιμο username στην εγγραφή χρήστη



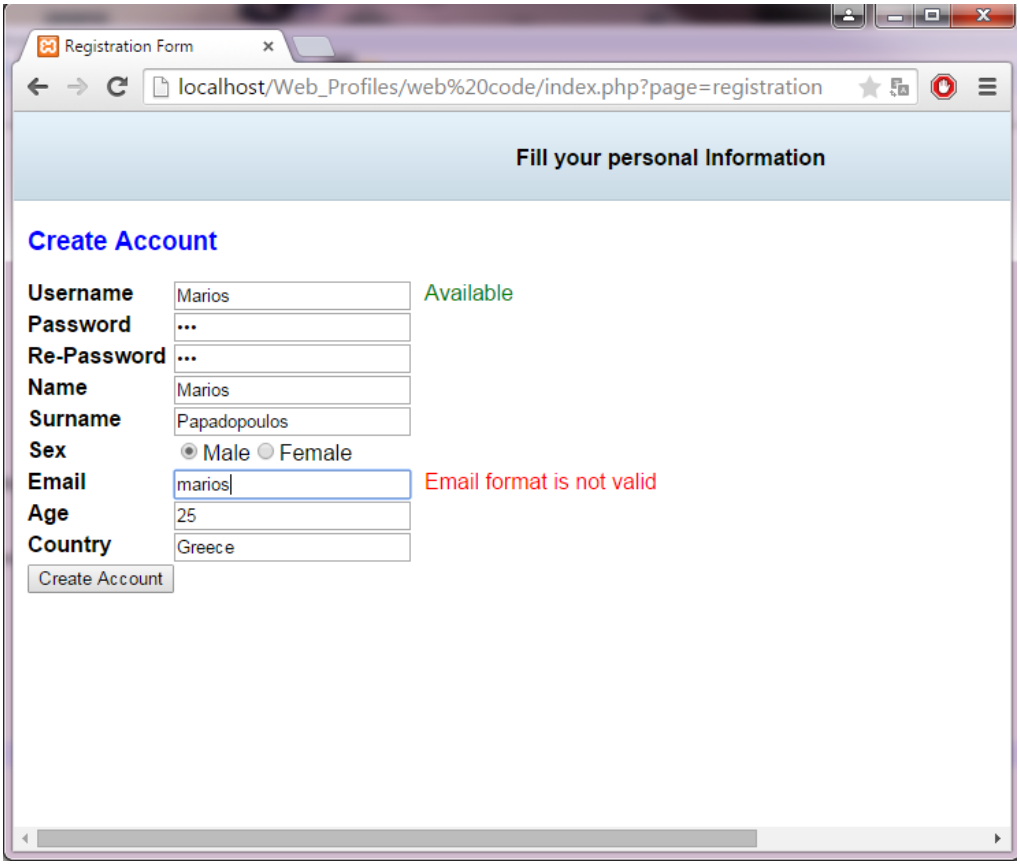
Εικόνα 6: Μη διαθέσιμο username

Παράλληλα γίνεται έλεγχος εγκυρότητας του κωδικού του χρήστη. Αρχικά ο χρήστης πληκτρολογεί τον κωδικό της επιλογής του στο πεδίο “Password”. Εν συνεχεία είναι απαιτούμενο να πληκτρολογήσει τον ίδιο κωδικό στο πεδίο “Re-Password”, με σκοπό να εξασφαλιστεί το ότι δεν έγινε κάποιο λάθος. Σε περίπτωση που οι δύο κωδικοί δεν ταιριάζουν επιστρέφετε προειδοποιητικό μήνυμα.



Εικόνα 7: Οι κωδικοί δεν ταιριάζουν

Ένας ακόμα σημαντικός έλεγχος που γίνεται είναι η πιστοποίηση του email, όπου το δοσμένο email θα πρέπει να πληρεί της προϋποθέσει ενός έγκυρου email. Έτσι όταν ο χρήστης πληκτρολογεί μη έγκυρο format email, τότε επιστρέφεται προειδοποιητικό μήνυμα για την μη κατάλληλη συμπλήρωση του εν λόγω πεδίου, όπως φαίνεται και στην παρακάτω εικόνα.

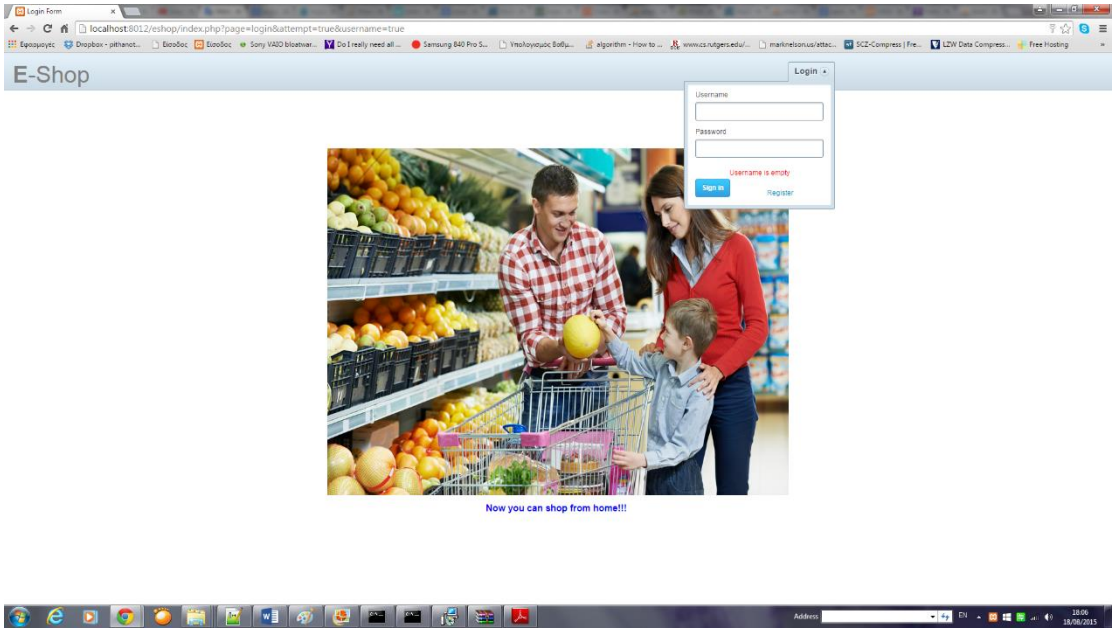


Εικόνα 8: Λάθος format email

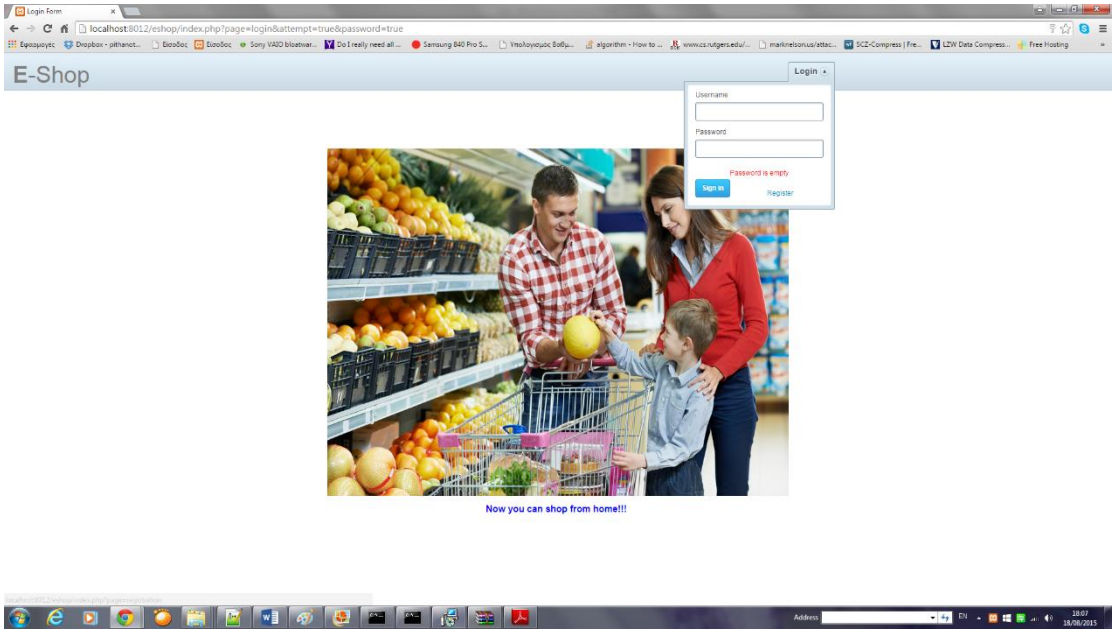
Τέλος αφού ο χρήστης συμπληρώσει σωστά όλα τα πεδία και επιλέξει το κουμπί “Create Account”, αποθηκεύονται τα στοιχεία του στη βάση και είναι πλέον σε θέσει να χρησιμοποιήσει τη φόρμα login που του επιτρέπει την είσοδο στο κύριος μέρος της εφαρμογής.

12.2 Σύνδεση Εγγεγραμμένου Χρήστη

Αφού κάποιος χρήστης έχει προβεί με επιτυχία στην εγγραφή του στο σύστημα, σειρά έχει η είσοδος του σε αυτό. Σε περίπτωση που τα στοιχεία τα οποία έχει πληκτρολογεί είναι λάθος, τότε επιστρέφεται κατάλληλο μήνυμα. Ποιο συγκεκριμένα σε περίπτωση που δεν συμπληρώσει το πεδίο “username” ή το πεδίο “password” εμφανίζεται προειδοποιητικό μήνυμα όπως παρουσιάζεται στις ακόλουθες εικόνες.

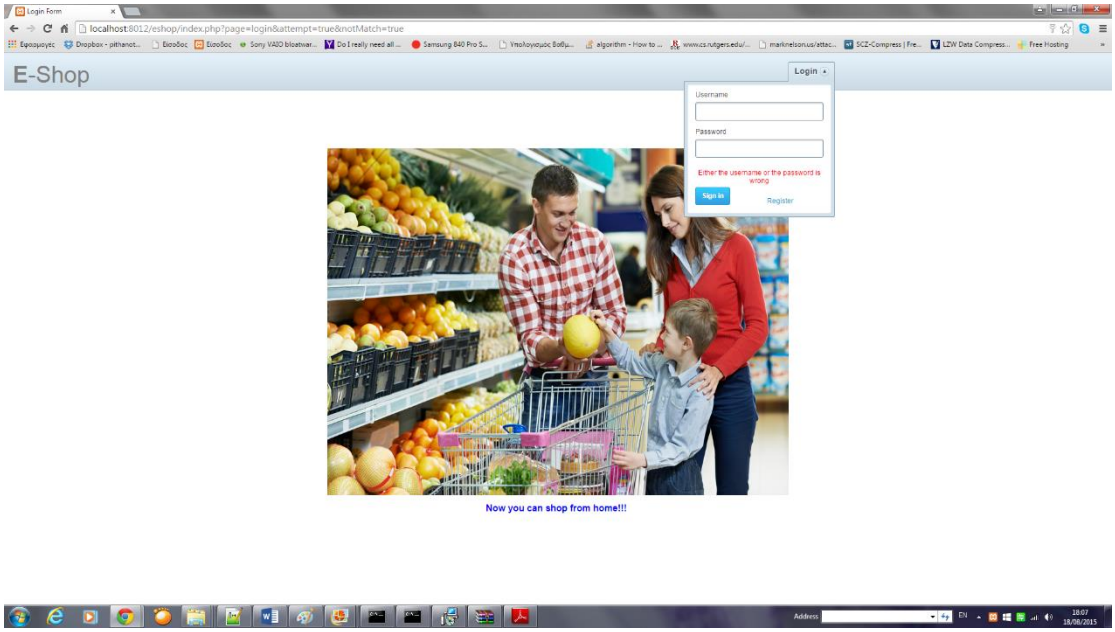


Εικόνα 9: Μη συμπληρωμένο username



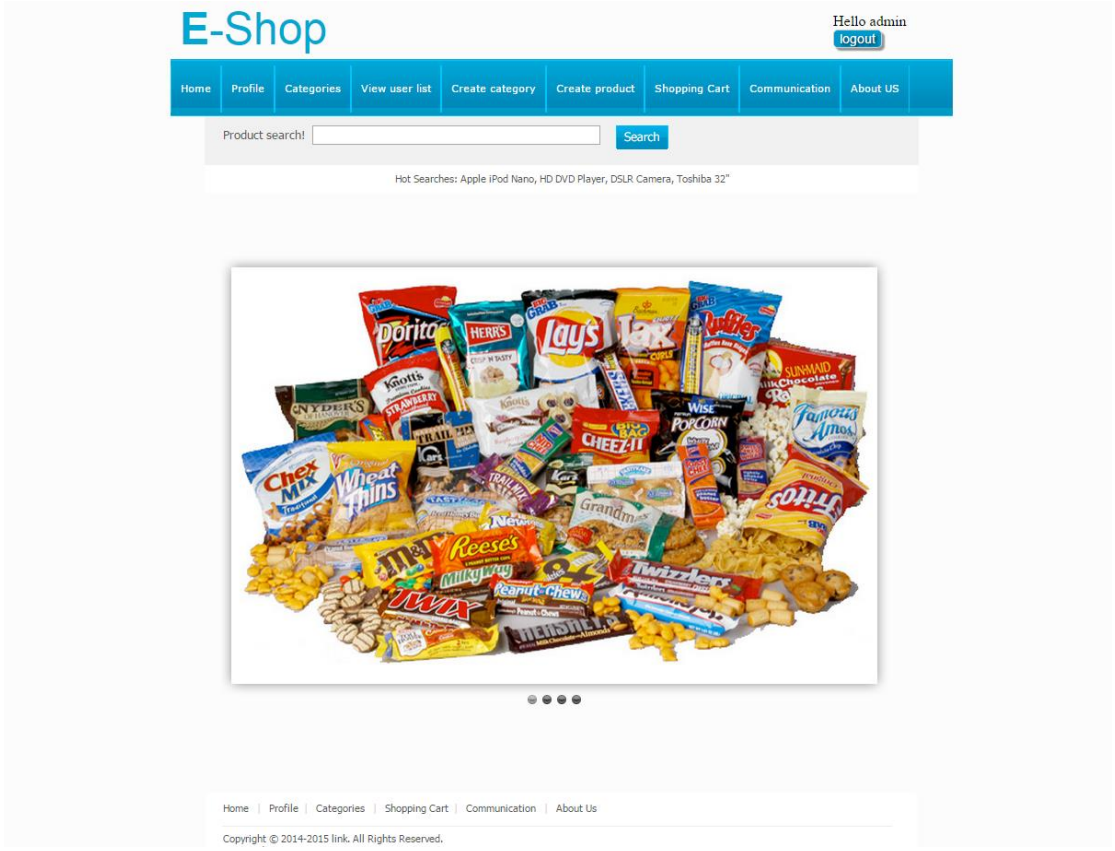
Εικόνα 10: Μη δοσμένο password

Μια άλλη περίπτωση είναι η συμπλήρωση λάθος στοιχείων, δηλαδή ένας εγγεγραμμένος χρήστης να πληκτρολογήσει λάθος το Username η Password που του αντιστοιχεί.



Εικόνα 11: Λάθος συνδυασμός username και password

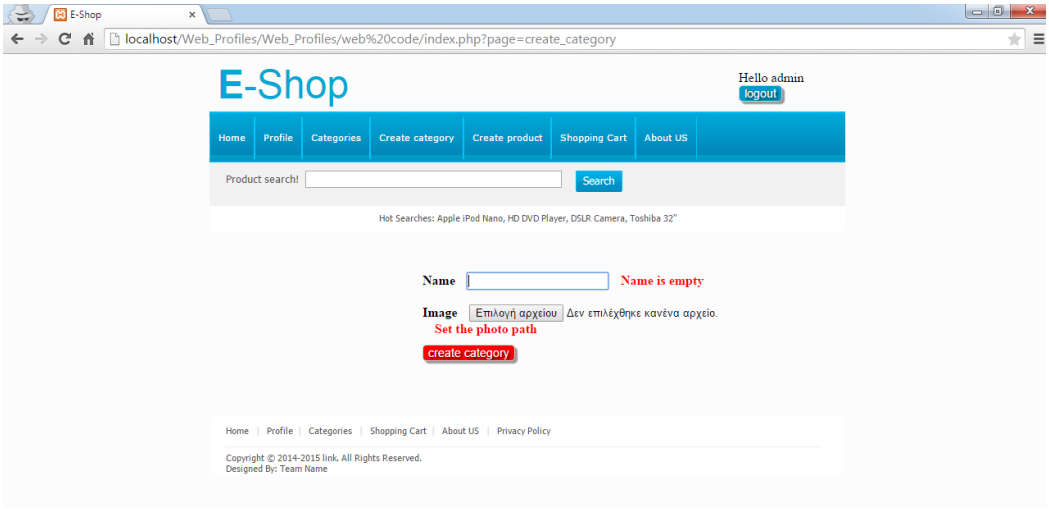
Τέλος σε περίπτωση σωστή συμπλήρωσης των στοιχείων του χρήστη το σύστημα τον ανακατευθύνει στην αρχική σελίδα της εφαρμογής.



Εικόνα 12: Επιτυχής σύνδεση

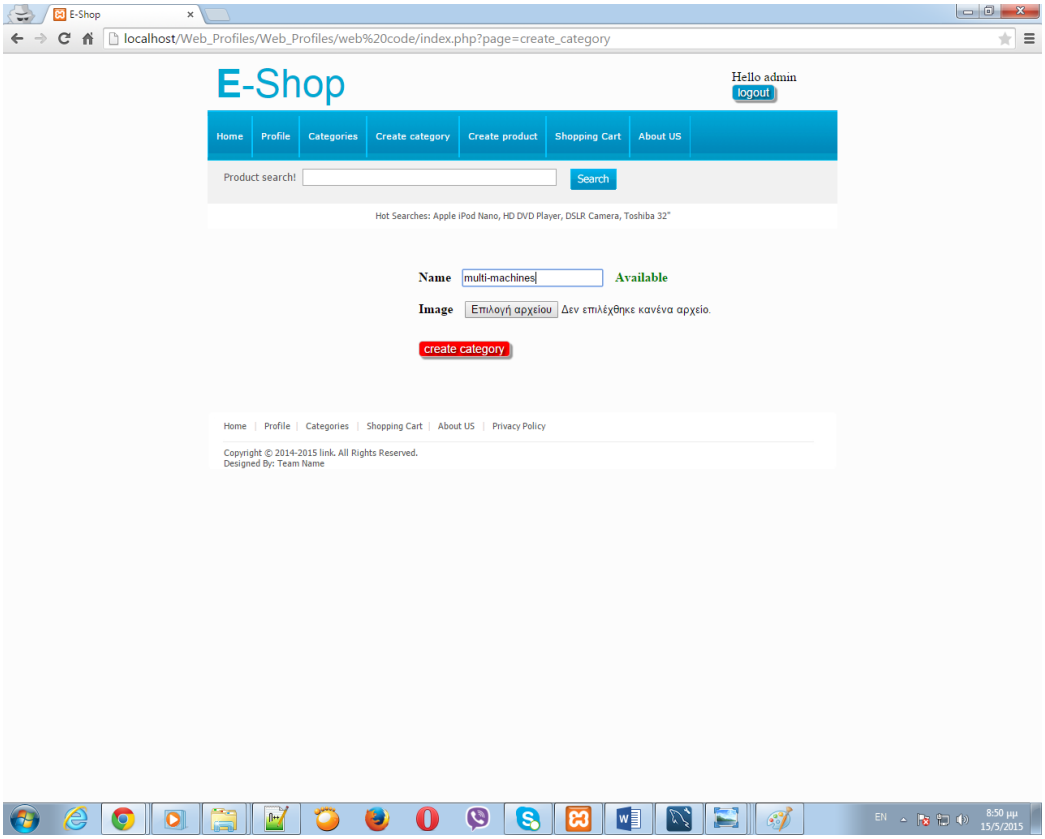
12.3 Δημιουργία Κατηγορίας Προϊόντων

Η παρούσα δυνατότητα επιτρέπει την δημιουργία νέας κατηγορίας προϊόντος από τον administrator της εφαρμογής. Πρώτα γίνετε έλεγχος για τον αν έχουν δοθεί τα ζητούμενα στοιχεία και εμφανίζεται κατάλληλο μήνυμα σε περίπτωση που δεν συμπληρώθηκαν όλα τα πεδία.

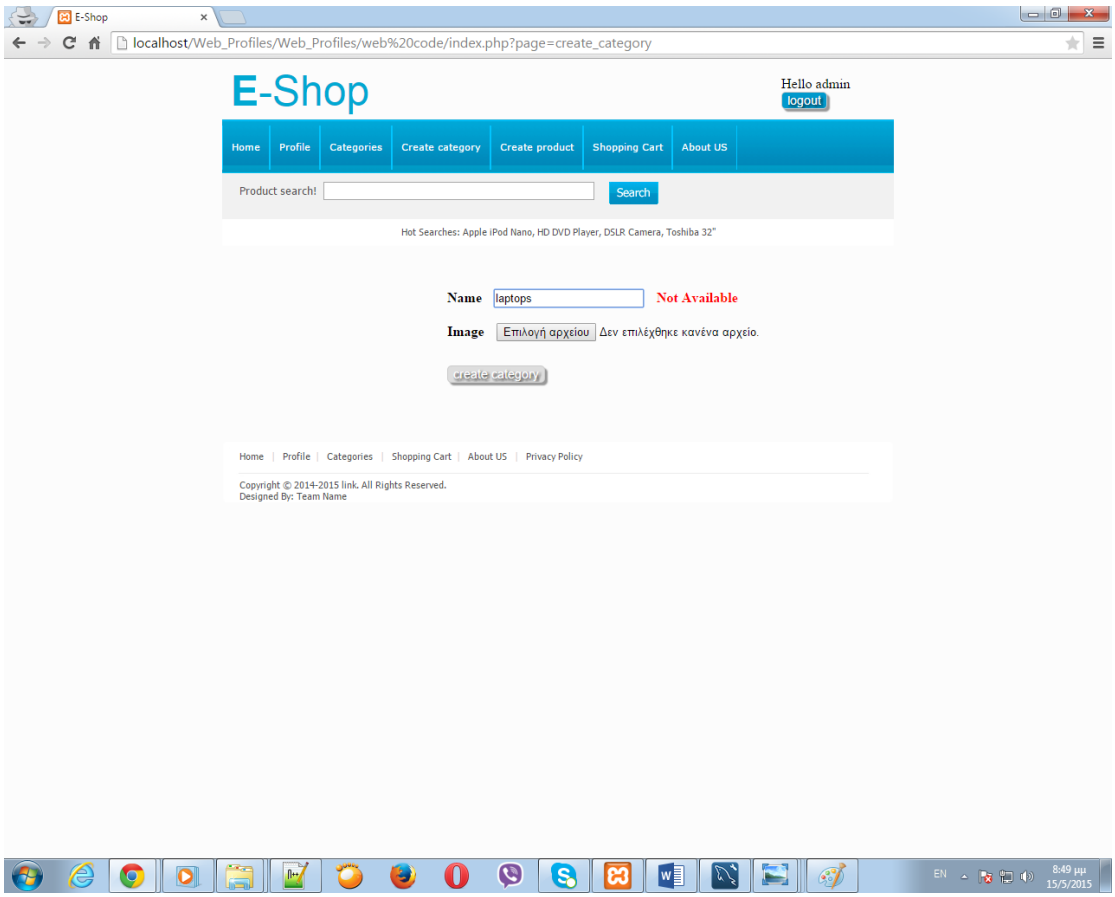


Εικόνα 13: Μη δοσμένο όνομα κατηγορίας

Επιπλέον γίνεται έλεγχος για τη διαθεσιμότητα του ονόματος της κατηγορίας. Παρακάτω παρουσιάζεται μια εικόνα όπου το όνομα που πληκτρολογείτε είναι διαθέσιμο και μία δεύτερη όπου το όνομα της κατηγορίας είναι ήδη δεσμευμένο.



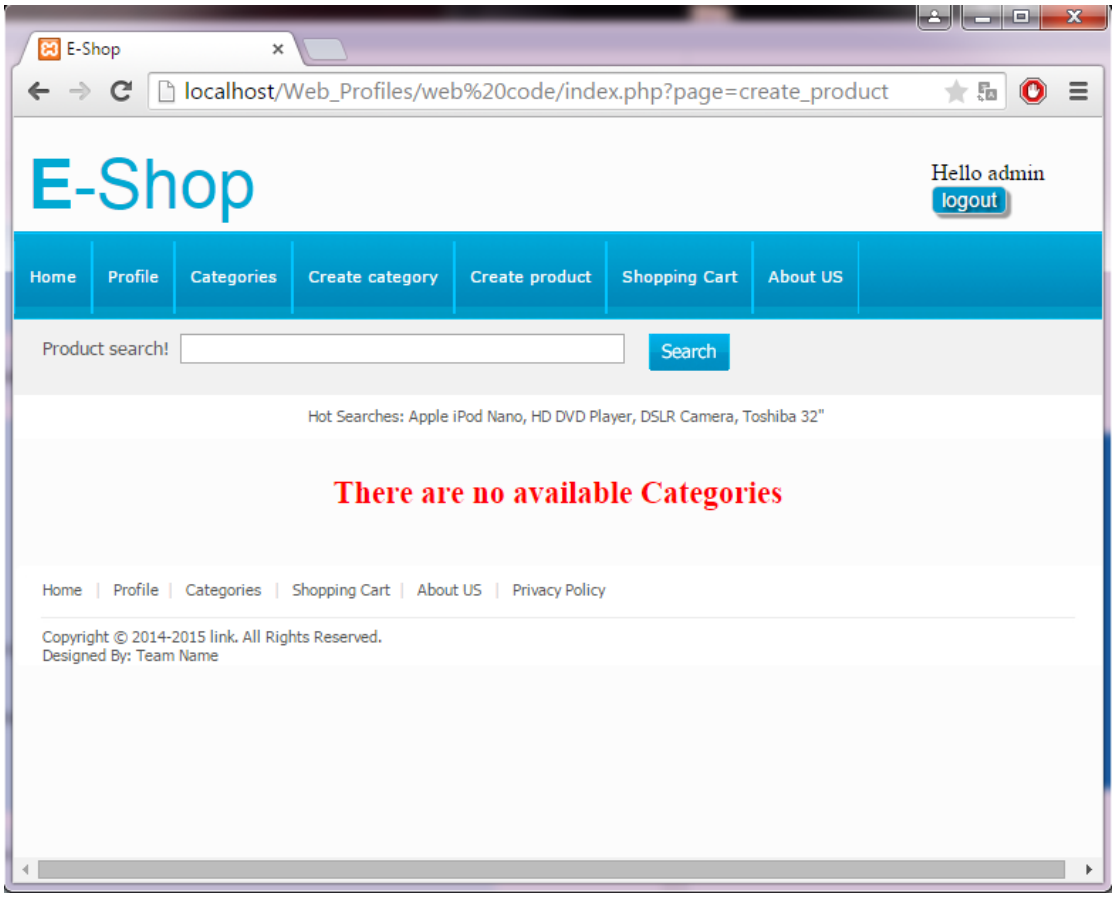
Εικόνα 14: Διαθέσιμο όνομα κατηγορίας



Εικόνα 15: Μη διαθέσιμο όνομα κατηγορίας

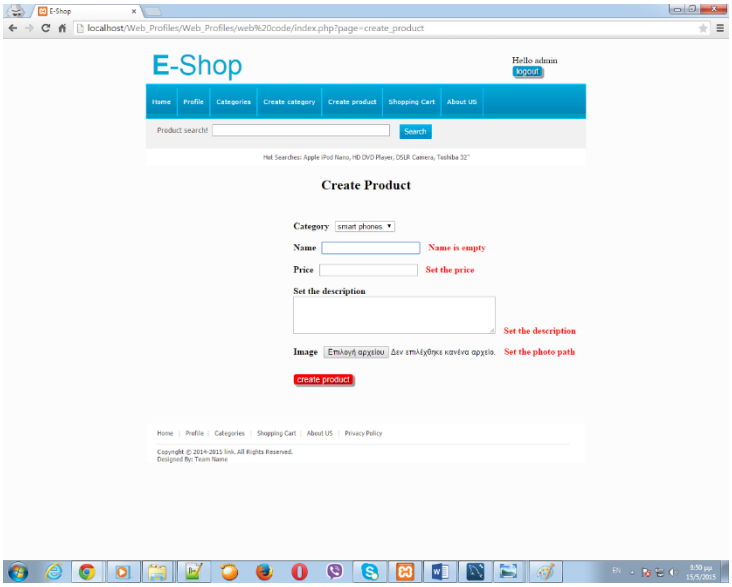
12.4 Δημιουργία Προϊόντος

Παρακάτω παρουσιάζονται εικόνες από την εφαρμογή κατά τη δημιουργία νέου προϊόντος. Η λειτουργία αυτή είναι διαθέσιμη μόνο στο διαχειριστή του συστήματος.



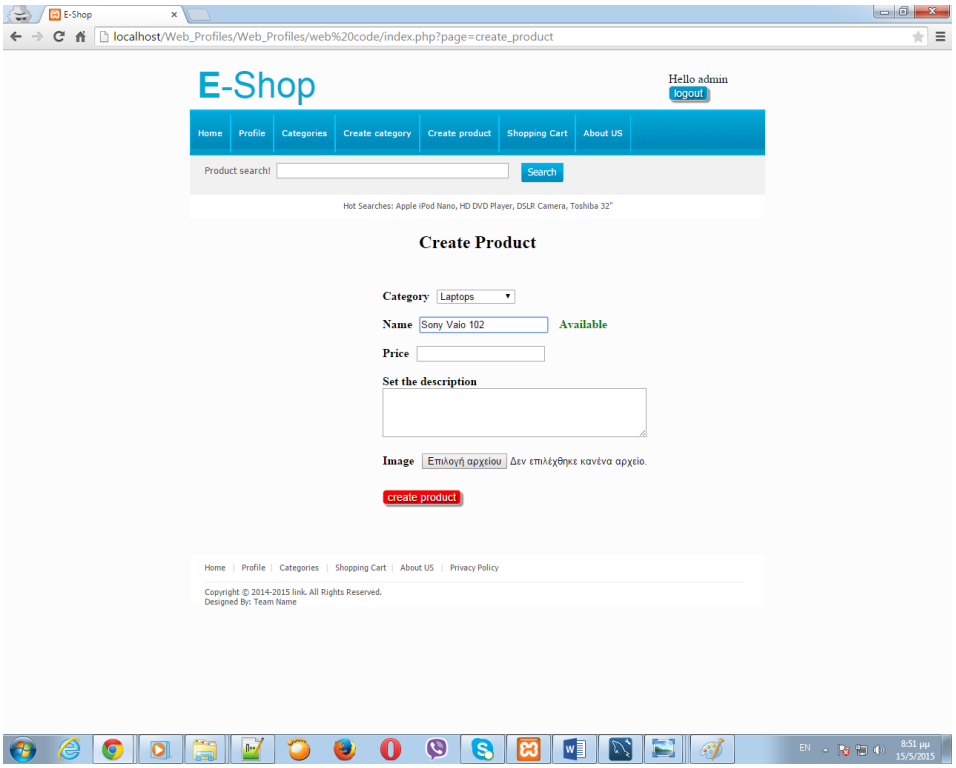
Εικόνα 16: Μη διαθέσιμες κατηγορίες

Στην παραπάνω φωτογραφία ο χρήστης δεν είναι σε θέση να δημιουργήσει νέο προϊόν, καθώς δεν υπάρχουν διαθέσιμες κατηγορίες προϊόντων.

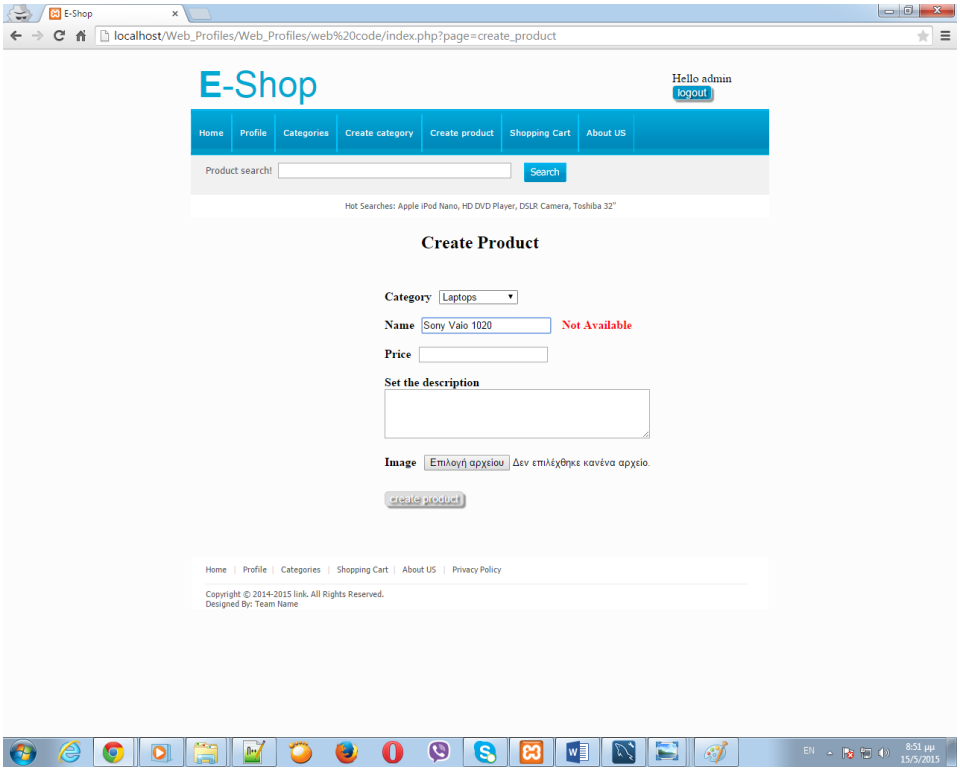


Εικόνα 17: Μη δοσμένα στοιχεία προϊόντος

Όπως βλέπουμε παραπάνω παρουσιάζονται τα μηνύματα που επιστρέφονται σε περίπτωση που ο administrator δεν έχει συμπληρώσει τα απαιτούμενα πεδία. Στη συνέχεια παρουσιάζεται η επιβεβαίωση έγκυρου ονόματος προϊόντος αλλά και η απόρριψη του ονόματος λόγο κατοχύρωσης του από προηγούμενη δημιουργία προϊόντος με το ίδιο όνομα.

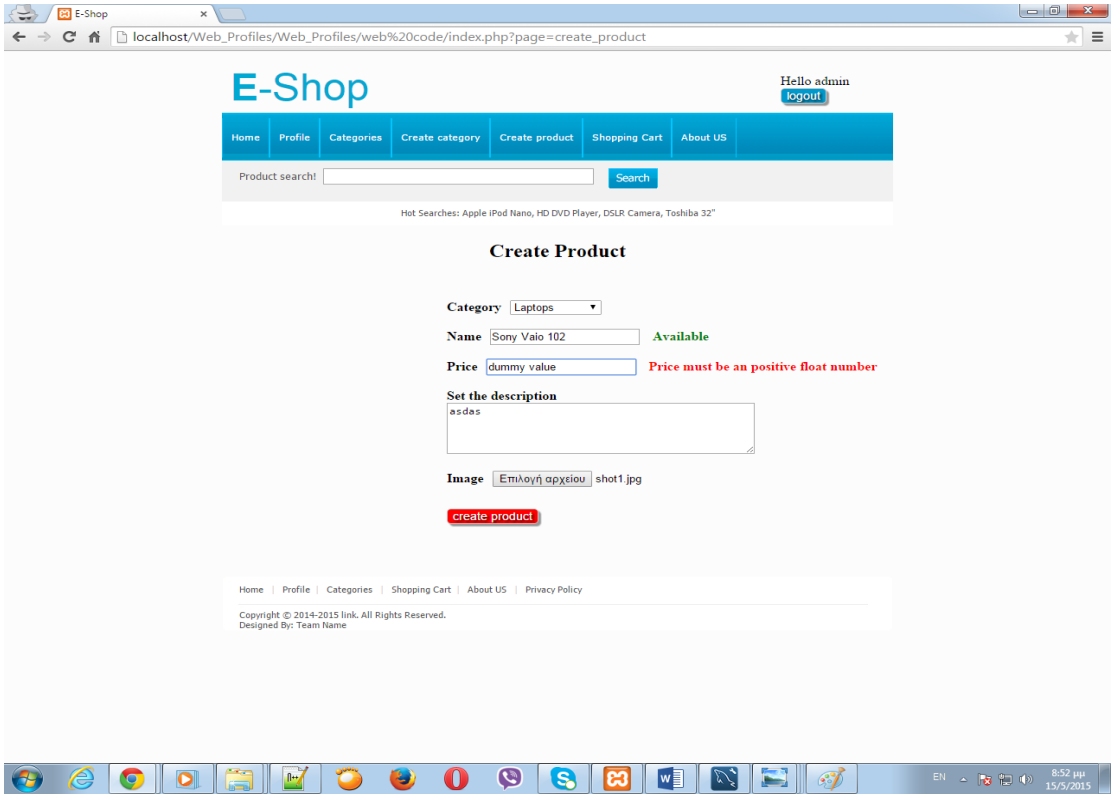


Εικόνα 18: Διαθέσιμο όνομα προϊόντος



Εικόνα 19: Μη διαθέσιμο όνομα προϊόντος

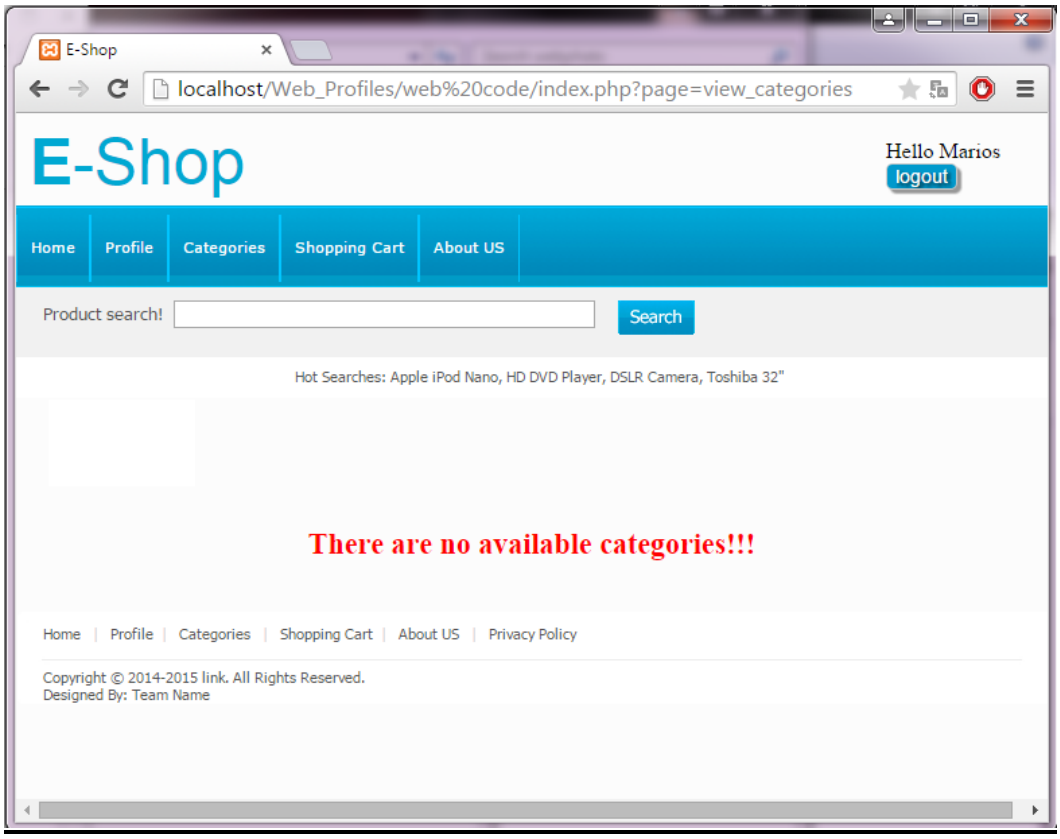
Τέλος στο πεδίο που απαιτείτε να συμπληρωθεί η τιμή του προϊόντος γίνεται έλεγχος προκειμένου να είναι έγκυρη, όπως φαίνεται και στην παρακάτω εικόνα, όπου έχει εισαχθεί μη έγκυρη τιμή.



Εικόνα 20: Λανθασμένο format τιμής προϊόντων

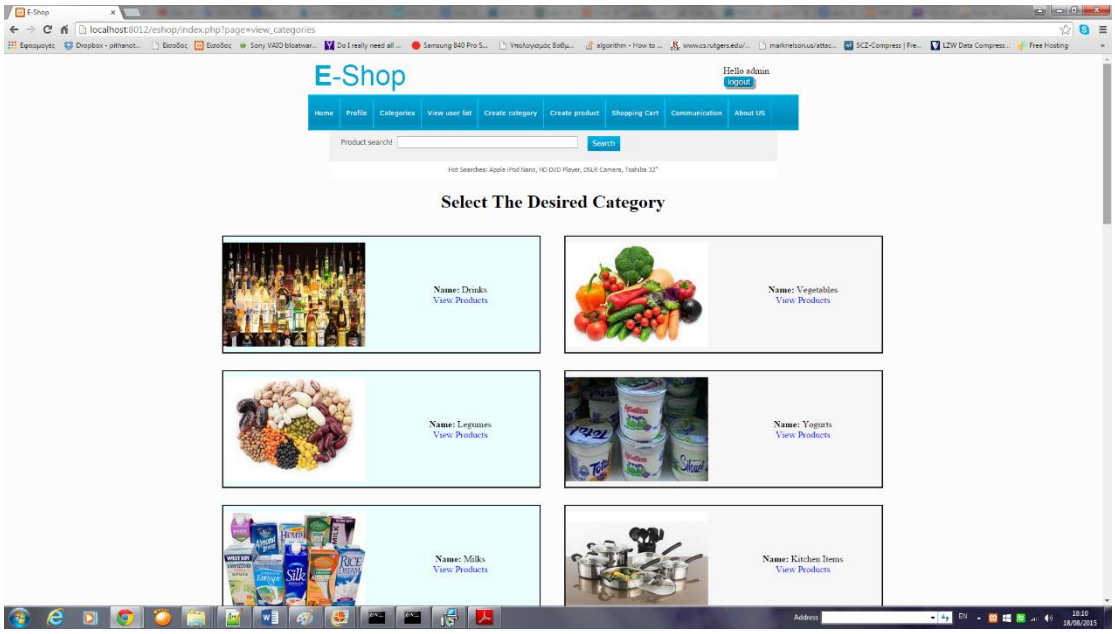
12.5 Προβολή Διαθέσιμων Κατηγορίας

Αυτή η λειτουργία αφορά την προβολή των διαθέσιμων κατηγοριών και είναι διαθέσιμη τόσο σε συνδεδεμένους χρήστες, όσο και σε μη συνδεδεμένους. Αρχικά παρουσιάζετε η σελίδα όπου δεν έχει προστεθεί καμία κατηγορία.



Εικόνα 21: Κενή λίστα κατηγοριών

Αντίθετα στην επόμενη εικόνα εμφανίζεται η ιστοσελίδα αφού έχουν προστεθεί ορισμένες κατηγορίες.

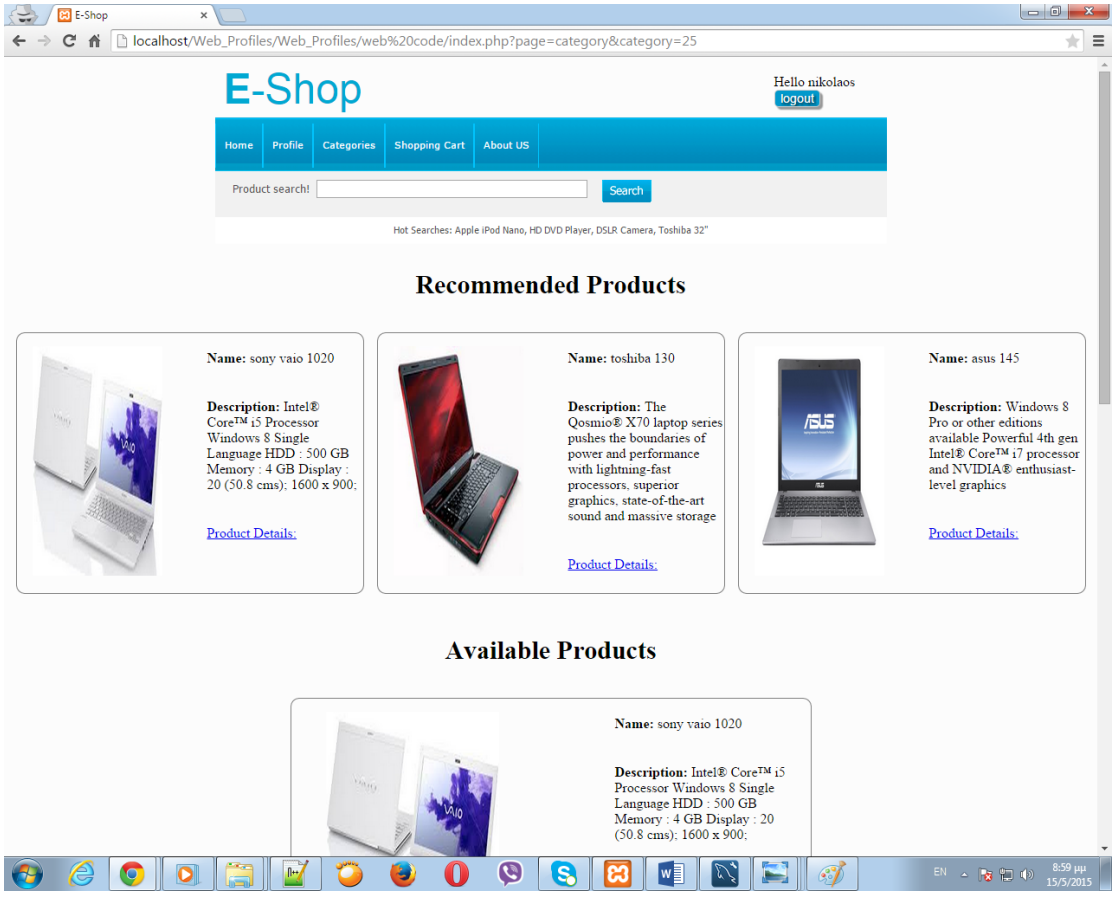


Εικόνα 22: Προβολή διαθέσιμων κατηγοριών

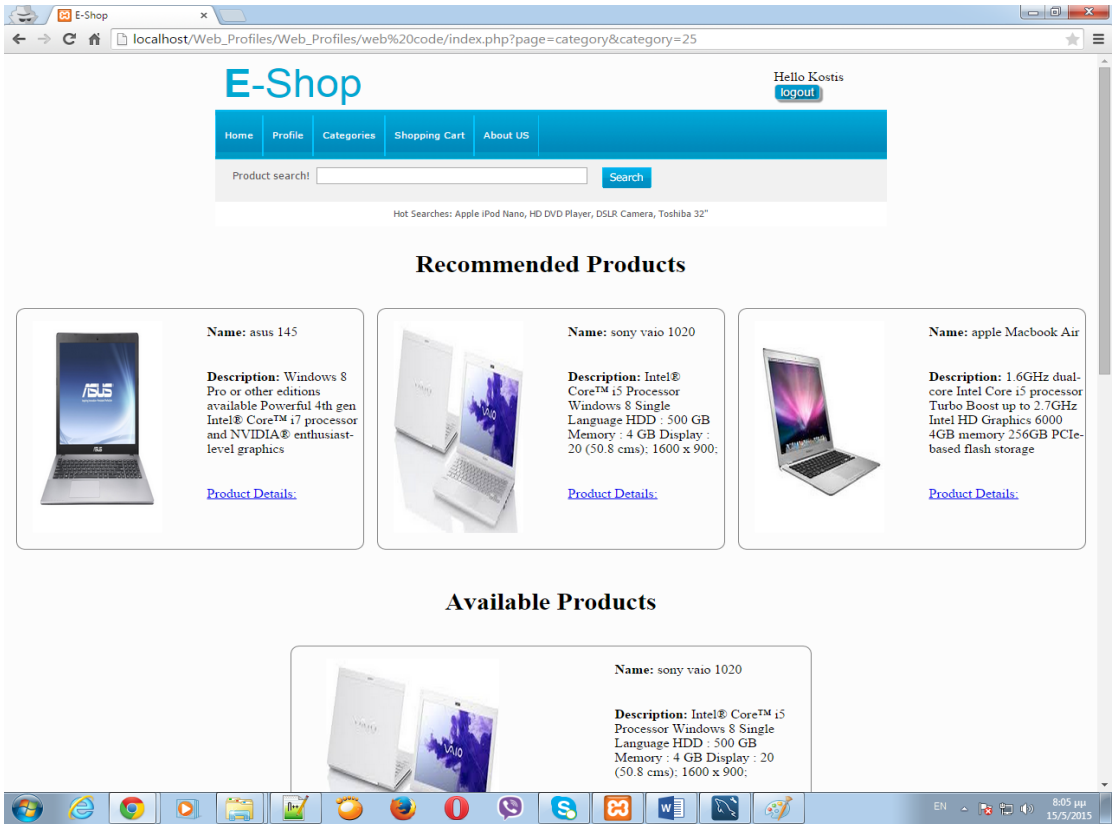
12.6 Προβολή Προϊόντων Κατηγορίας

Η λειτουργία αυτή εμφανίζει τα προϊόντα που ανήκουν σε μια συγκεκριμένη κατηγορία. Η λειτουργία αυτή είναι διαθέσιμη τόσο σε μη συνδεδεμένους όσο και σε συνδεδεμένους χρήστες.

Στις παρακάτω δύο εικόνες παρουσιάζεται η λίστα με τα διαθέσιμα προϊόντα, καθώς και τα προτεινόμενα προϊόντα ανά χρήστη. Παρατίθενται οι λίστες από δύο ξεχωριστούς χρήστες, στους οποίους λόγω της εξατομικευμένης σύστασης προτάσεων εμφανίζεται διαφορετική λίστα προτεινόμενων προϊόντων



Εικόνα 23: Λίστα Προτεινόμενων προϊόντων στο χρήστη Nikolaos

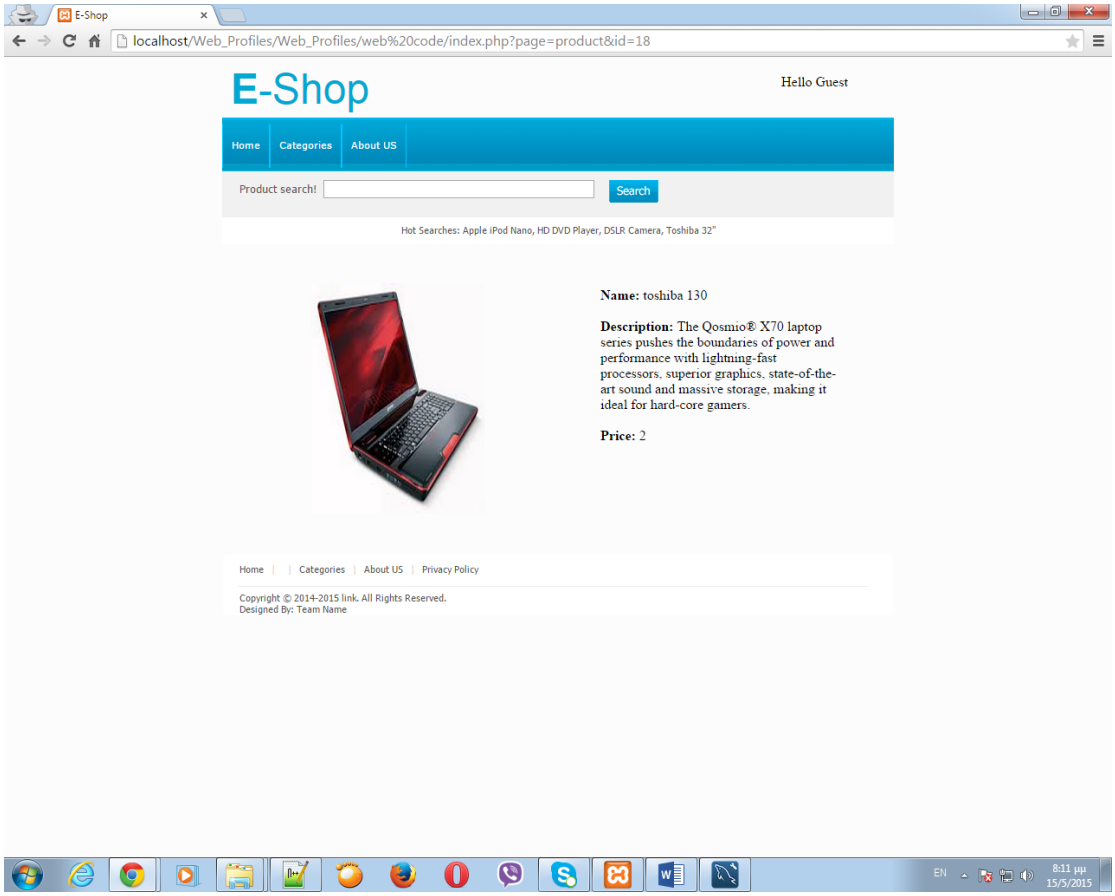


Εικόνα 24: Λίστα προτεινόμενων προϊόντων στο χρήστη Kostis

12.7 Προβολή Στοιχείων Προϊόντος

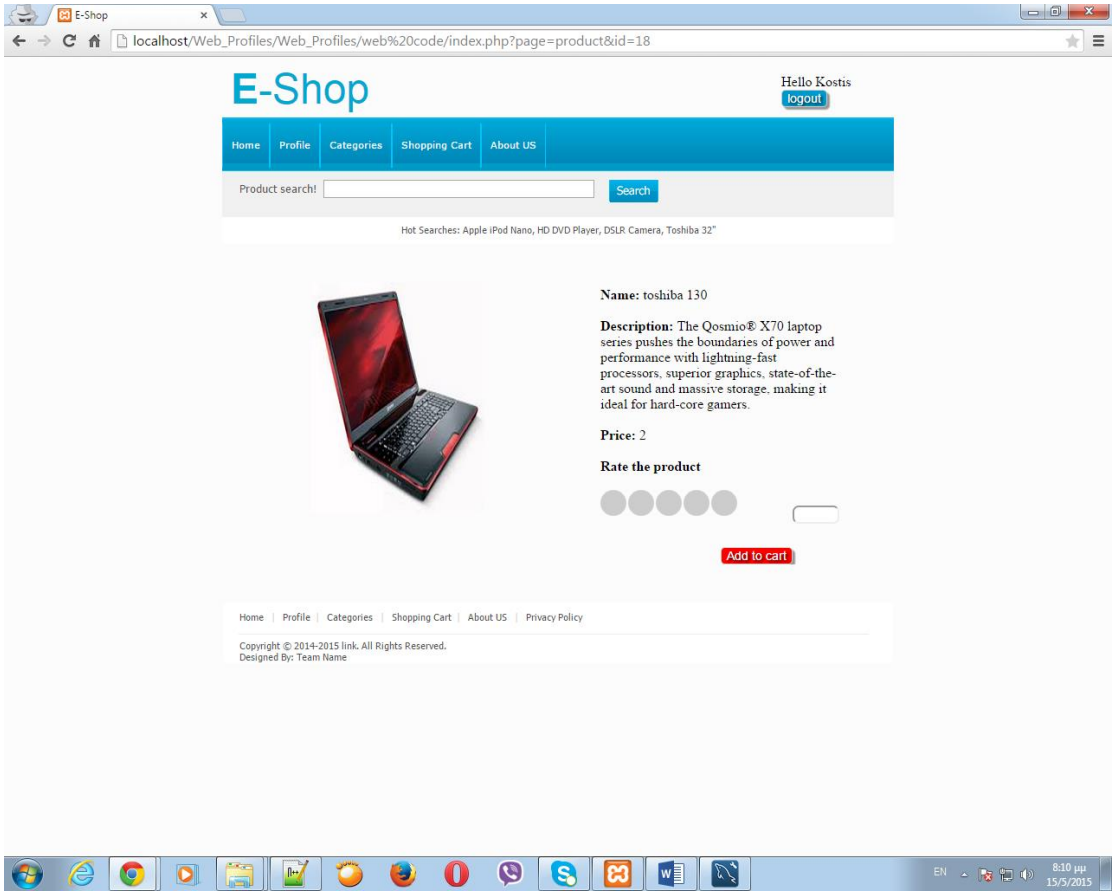
Η λειτουργία αυτή είναι διαθέσιμη τόσο σε συνδεδεμένους όσο και σε μη συνδεδεμένους χρήστες και εμφανίζει τις πληροφορίες ενός προϊόντος.

Ο μη συνδεδεμένος χρήστης μπορεί να δει μόνο τις βασικές πληροφορίες ενός προϊόντος, όπως παρουσιάζεται και στην επόμενη εικόνα.



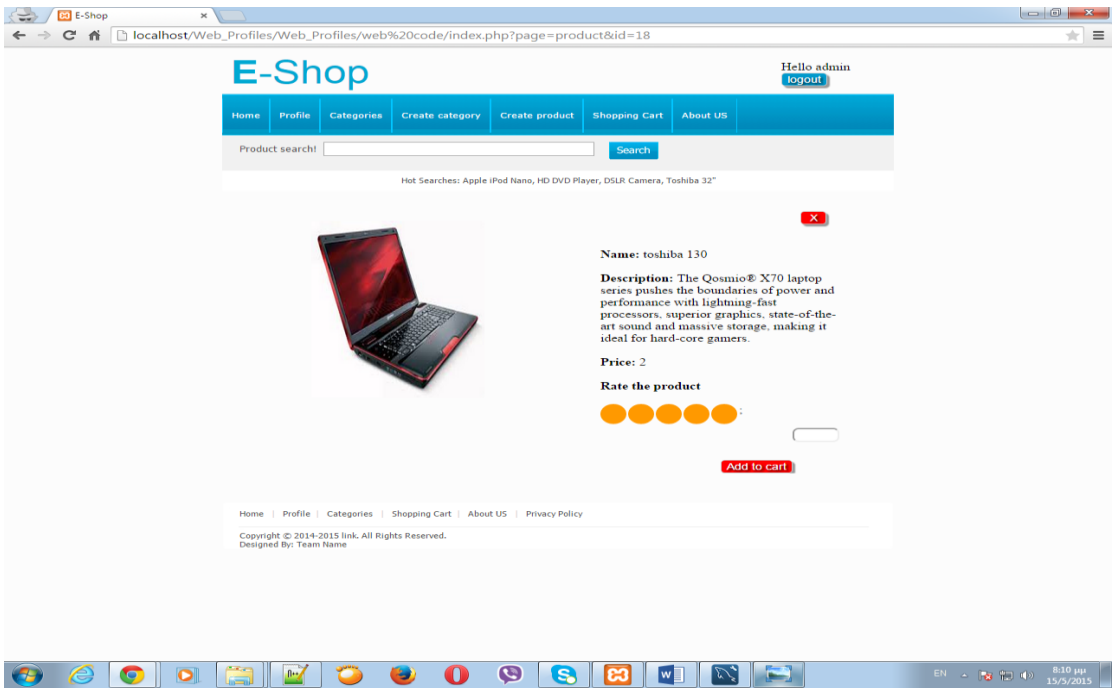
Εικόνα 25: Προβολή προϊόντος για μη συνδεδεμένους χρήστες

Αντίθετα, στους συνδεδεμένους απλούς χρήστες εμφανίζεται τόσο η δυνατότητα να βαθμολογήσουν το προϊόν, όσο και να το προσθέσουν στο καλάθι αγορών στην επιθυμητή τιμή. Σε περίπτωση που ο χρήστης έχει βαθμολογήσει στο παρελθόν το προϊόν, τότε εμφανίζεται η τρέχουσα τιμή βαθμολόγησης.



Εικόνα 26: Προβολή προϊόντος για απλούς συνδεδεμένους χρήστες

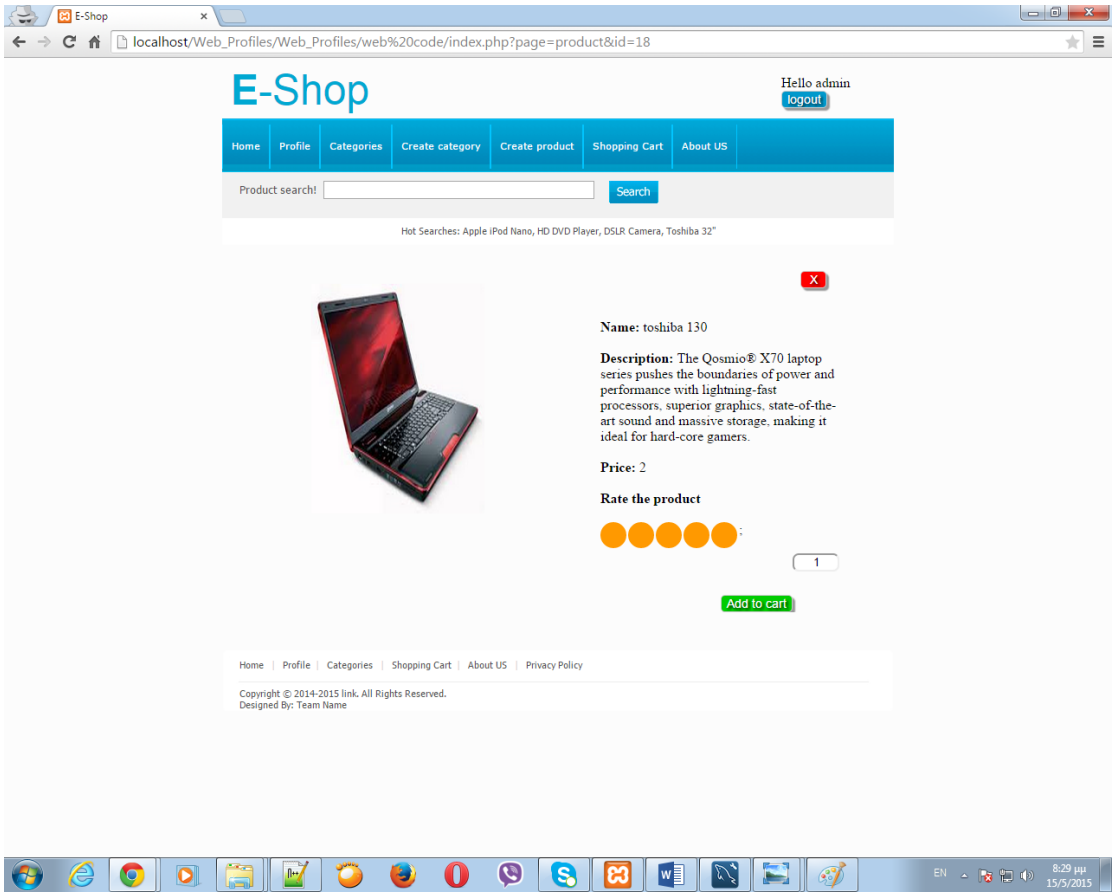
Τέλος ο διαχειριστής του συστήματος έχει την επιπρόσθετη δυνατότητα να διαγράψει το προϊόν.



Εικόνα 27: Προβολή προϊόντος στο διαχειριστή

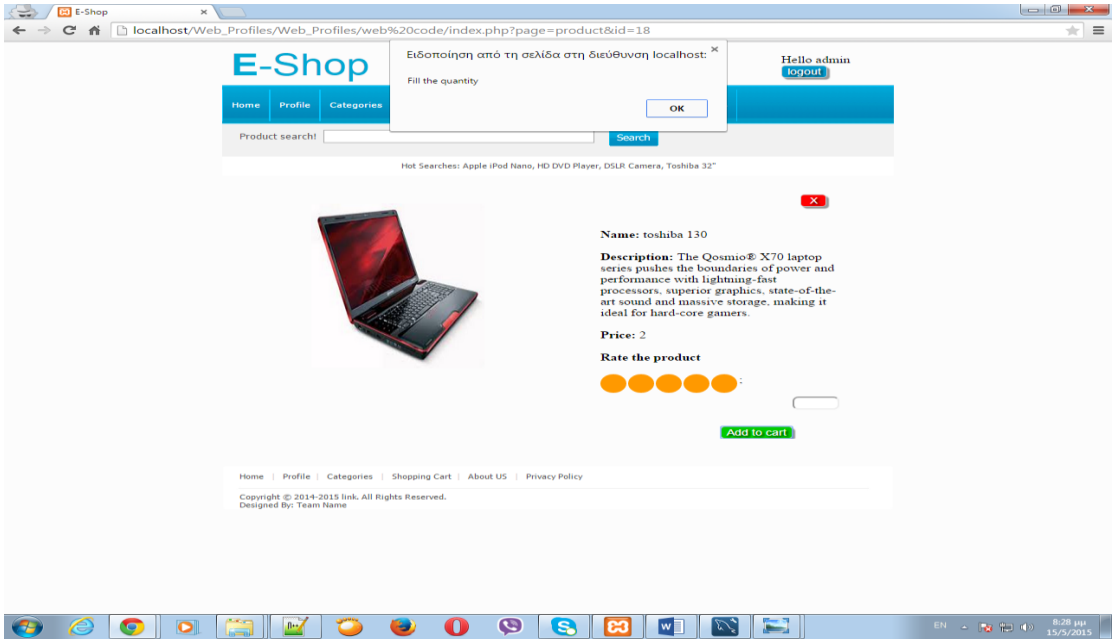
12.8 Εισαγωγή Προϊόντος Στο Καλάθι Αγορών

Οι συνδεδεμένοι χρήστες έχουν τη δυνατότητα να εισάγουν ένα προϊόν στο καλάθι αγορών στην επιθυμητή ποσότητα. Για να γίνει εισαγωγή ενός προϊόντος στο καλάθι αγορών θα πρέπει πρώτα να επιλεχθεί η επιθυμητή ποσότητα και στη συνέχεια να πατηθεί το κουμπί “Add to cart”, όπως φαίνεται και στην επόμενη φωτογραφία.



Εικόνα 28: Εισαγωγή προϊόντος στο καλάθι αγορών

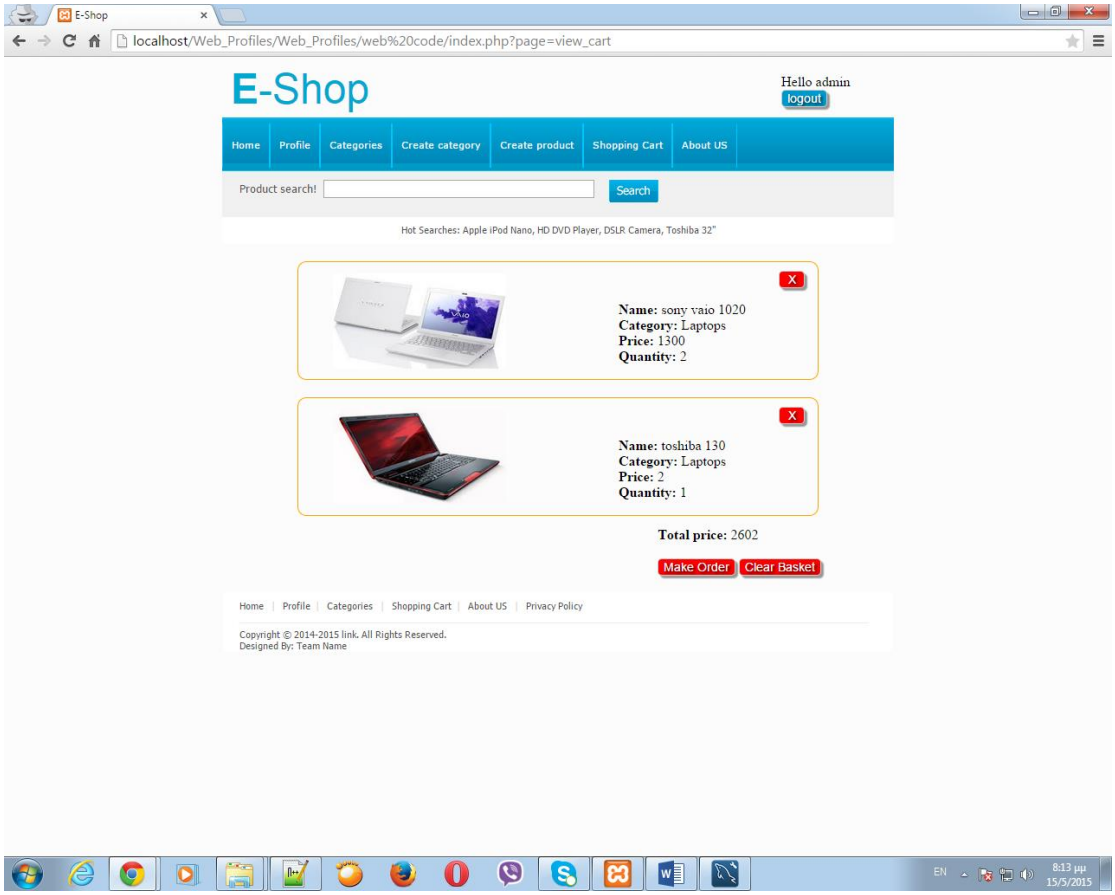
Στην επόμενη φωτογραφία εμφανίζεται η απόπειρα να εισαχθεί ένα προϊόν στο καλάθι αγορών, χωρίς να έχει πρώτα δοθεί η επιθυμητή ποσότητα.



Εικόνα 29: Μη δοθείσα ποσότητα

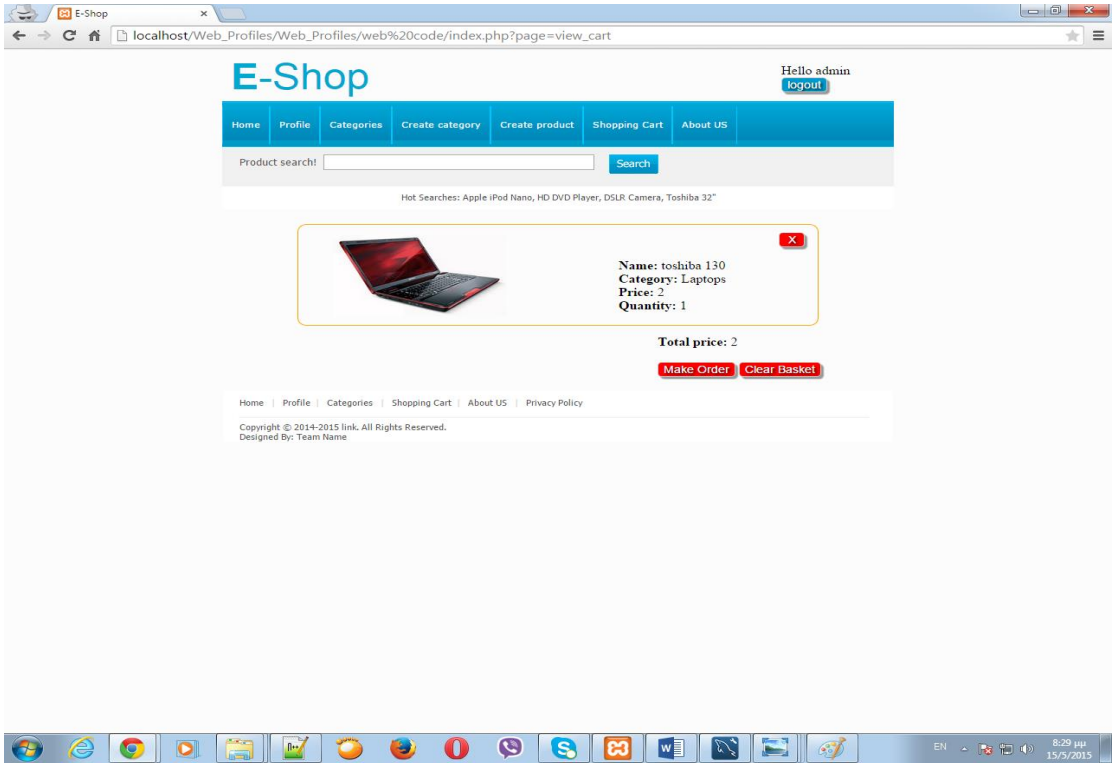
12.9 Προβολή Καλαθιού Αγορών

Ένας συνδεδεμένος χρήστης έχει τη δυνατότητα να προβάλλει το καλάθι αγορών του. Η ενέργεια αυτή εμφανίζει όλα τα προϊόντα που βρίσκονται στο καλάθι αγορών στην επιθυμητή ποσότητα. Επίσης εμφανίζεται η συνολική τιμή του καλαθιού αγορών, η οποία λαμβάνει υπόψη τόσο την τιμή των επιλεγμένων προϊόντων, όσο και την επιλεγμένη τιμή, όπως παρουσιάζεται στη επόμενη εικόνα. Επιπρόσθετα δίνεται η δυνατότητα σε ένα χρήστη να αδειάσει το καλάθι αγορών του(Clear basket), καθώς και να παραγγείλει τα προϊόντα που βρίσκονται στο καλάθι αγορών(Make order).



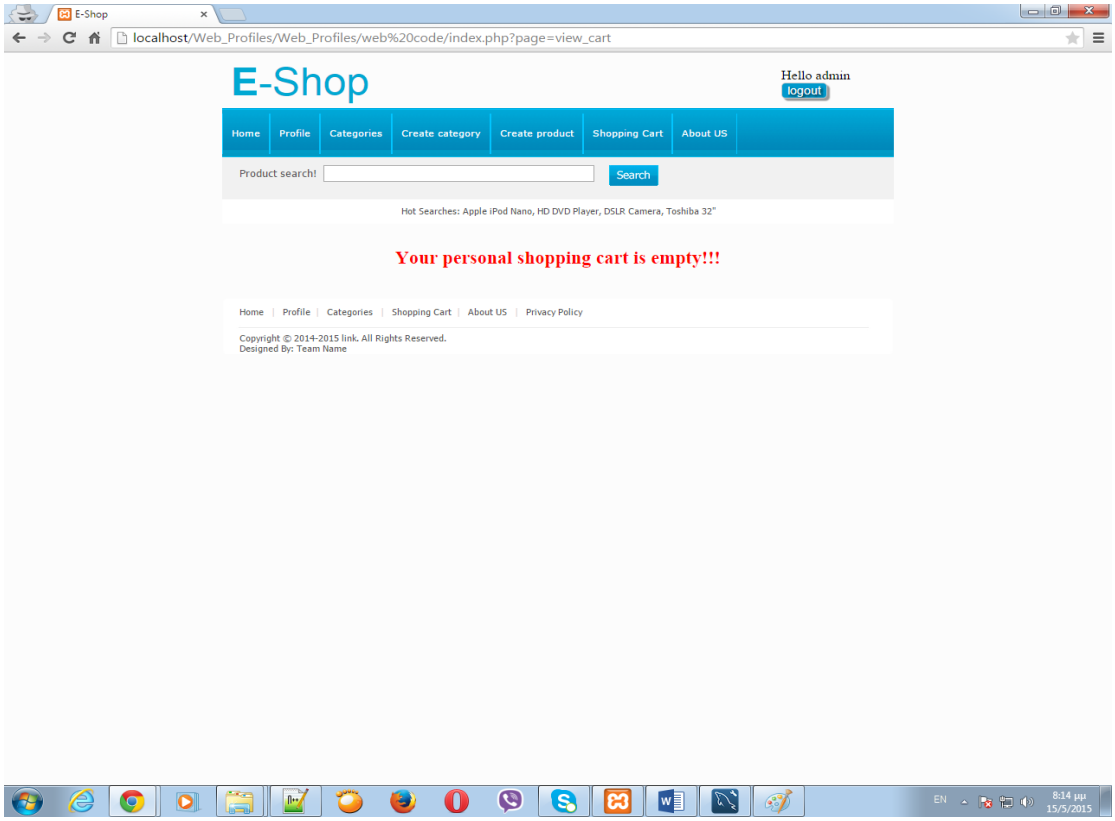
Εικόνα 30: Προβολή καλαθιού αγορών

Ο χρήστης έχει τη δυνατότητα να διαγράψει ένα ή παραπάνω από τα προϊόντα που βρίσκονται στο καλάθι αγορών του. Στη περίπτωση αυτή ενημερώνεται το καλάθι αγορών και ανανεώνεται η συνολική τιμή του καλαθιού αγορών. Στην επόμενη εικόνα παρουσιάζεται το καλάθι αγορών του χρήστη, αφού έχει πρώτα διαγραφεί το προϊόν “sony vaio 1020”. Επειδή το προϊόν αυτό βρισκόταν σε ποσότητα δύο μέσα στο καλάθι αγορών, επομένως η νέα συνολική τιμή στο καλάθι αγορών θα είναι ίση με 2602-2x1300=2 ευρώ, όπως φαίνεται και στην επόμενη εικόνα.



Εικόνα 31: Προβολή καλαθιού μετά διαγραφής προϊόντος

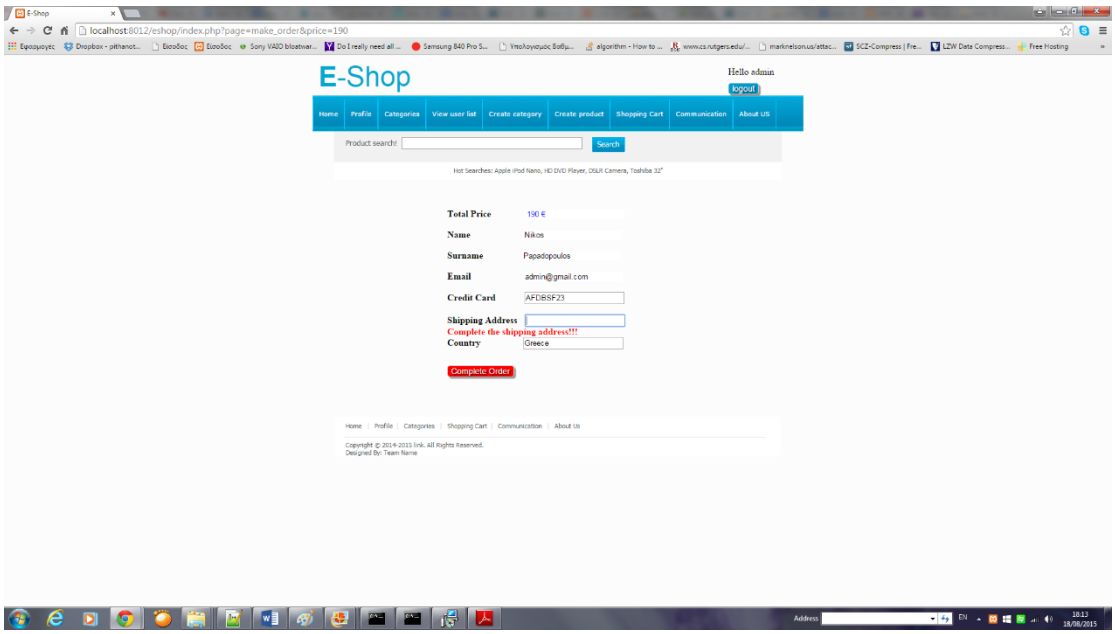
Σε περίπτωση που το καλάθι αγορών είναι άδειο, τότε εμφανίζεται αντίστοιχο μήνυμα, όπως φαίνεται στη παρακάτω εικόνα.



Εικόνα 32: Άδειο καλάθι αγορών

12.10 Ολοκλήρωση Παραγγελίας

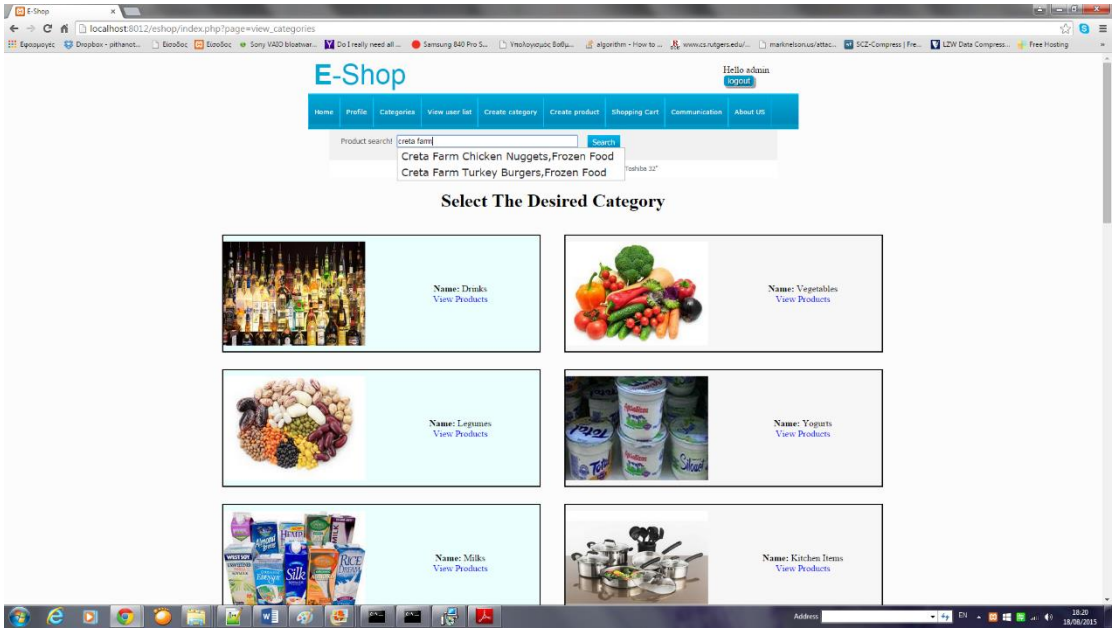
Ένας συνδεδεμένος χρήστης είναι σε θέση να ολοκληρώσει τη παραγγελία, αγοράζοντας τα προϊόντα που βρίσκονται στο καλάθι αγορών του. Αν ο χρήστης επιλέξει την επιλογή παραγγελίας, τότε θα πρέπει εισάγει τον αριθμό της πιστωτικής κάρτας, την διεύθυνση αποστολής της παραγγελίας και τη χώρα αποστολής. Σε περίπτωση που δεν έχει συμπληρωθεί μία από τις τρεις προαναφερθείσες πληροφορίες, τότε εμφανίζεται κατάλληλο μήνυμα, όπως φαίνεται και στην επόμενη εικόνα.



Εικόνα 33: Έλλειψη διεύθυνσης αποστολής παραγγελίας

12.11 Μπάρα Αναζήτησης Προϊόντος

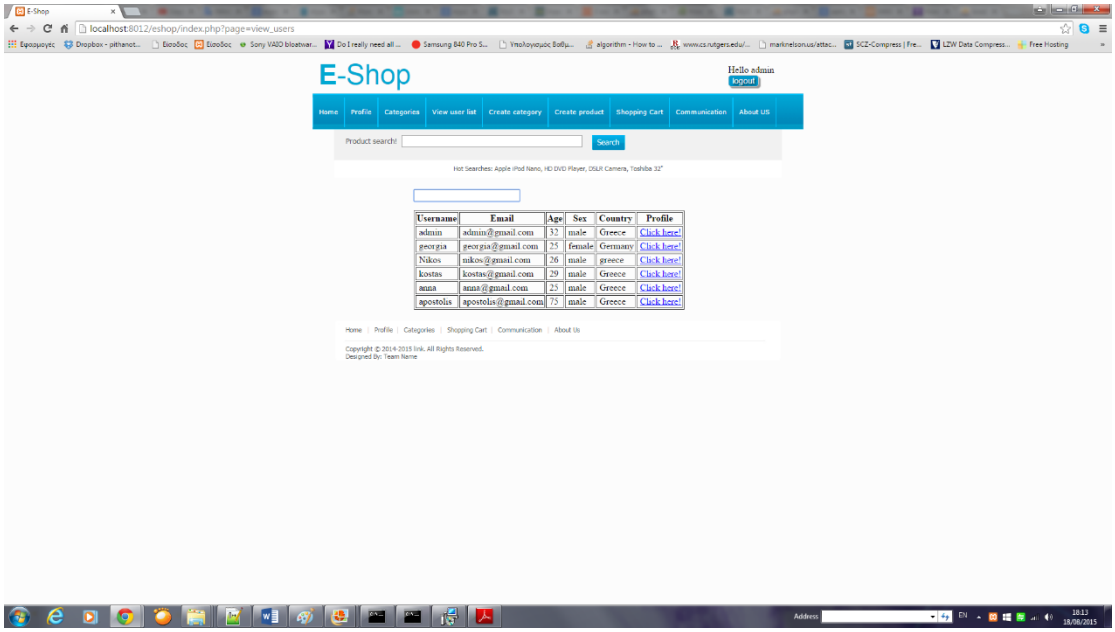
Η εφαρμογή δίνει την δυνατότητα στους χρήστες να εντοπίσουν αυτόματα ένα συγκεκριμένο προϊόν, πληκτρολογώντας το όνομα του. Με βάση τα στοιχεία που έχουν δοθεί στην μπάρα αναζήτησης, προβάλλονται τα προϊόντα(ακολουθούμεθα από το όνομα της κατηγορίας στην οποία ανήκουν), των οποίων το όνομα ξεκινά με βάση τη πληκτρολογημένη λέξη. Τα προβαλλόμενα προϊόντα ενημερώνονται δυναμικά, με βάση τα δεδομένα στη μπάρα αναζήτησης. Σε περίπτωση που υπάρχουν παραπάνω από τρία προϊόντα με πρόθεμα την πληκτρολογούμενη λέξη, τότε θα προβάλλονται τα τρία πιο δημοφιλή. Η αναζήτηση αρχίζει να προβάλλει προϊόντα αφού έχουν πληκτρολογηθεί τουλάχιστον τρία γράμματα.



Εικόνα 34: Αναζήτηση προϊόντος με βάση κάποιο πρόθεμα

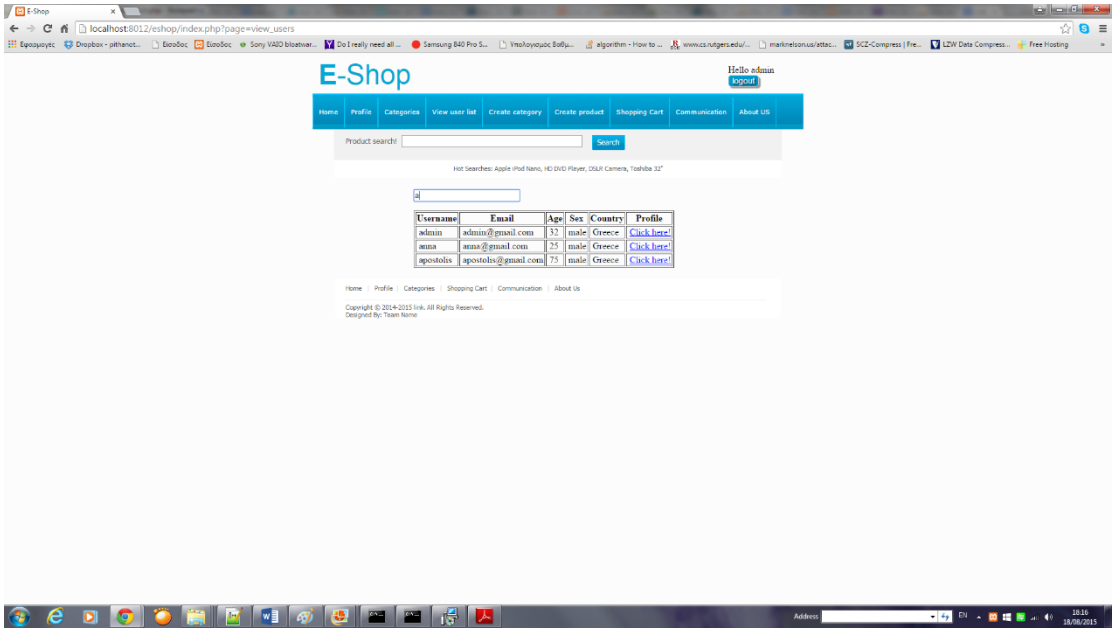
12.12 Προβολή Λίστας Χρηστών

Ο διαχειριστής της εφαρμογής έχει τη δυνατότητα να πλοηγηθεί στη λίστα των χρηστών της εφαρμογής και να επιλέξει το χρήστη του οποίου επιθυμεί να επισκεφτεί το προφίλ. Στην επόμενη εικόνα παρουσιάζεται η λίστα των χρηστών της εφαρμογής.



Εικόνα 35: Προβολή λίστας χρηστών

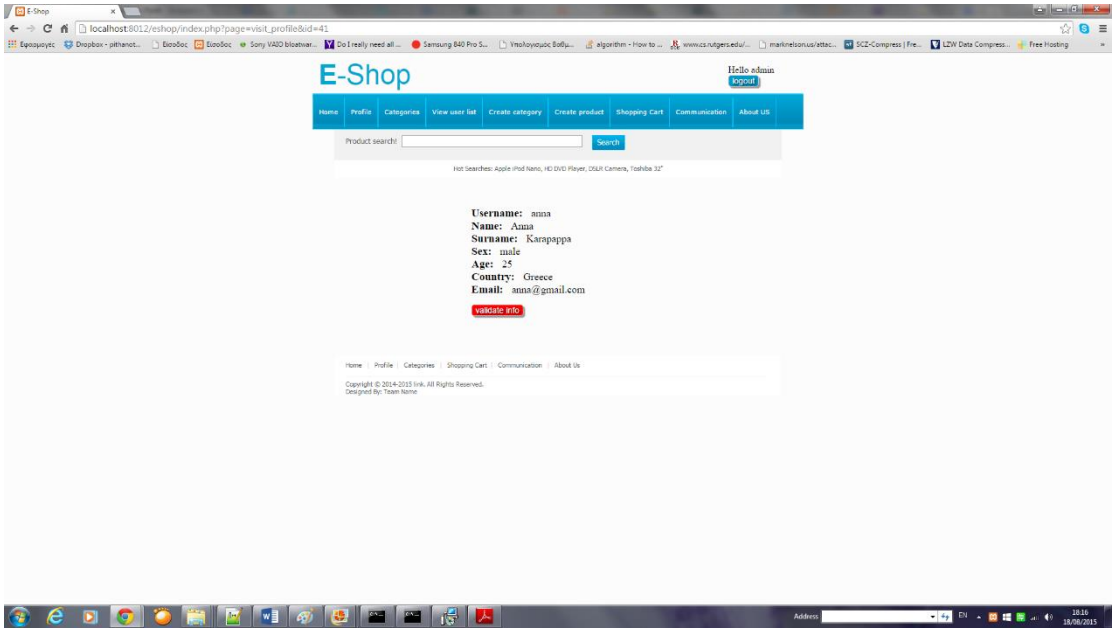
Επιπλέον, ο διαχειριστής έχει τη δυνατότητα να φιλτράρει τη λίστα των χρηστών, με βάση την ειδική μπάρας αναζήτησης που προσφέρει η συγκεκριμένα σελίδα. Πληκτρολογώντας στη μπάρα αναζήτησης, η λίστα επιστρέφει δυναμικά μόνο τους χρήστες, των οποίων το username ξεκινάει με βάση της πληκτρολογημένης λέξης. Στην επόμενη εικόνα παρουσιάζεται η λίστα που προκύπτει, αφού ο διαχειριστής έχει πληκτρολογήσει το γράμμα «α» στην ειδικά μπάρα αναζήτησης.



Εικόνα 36: Προβολή λίστας χρηστών που το username ξεκινάει με το γράμμα «α»

12.12 Επίσκεψη Προφίλ Χρήστης

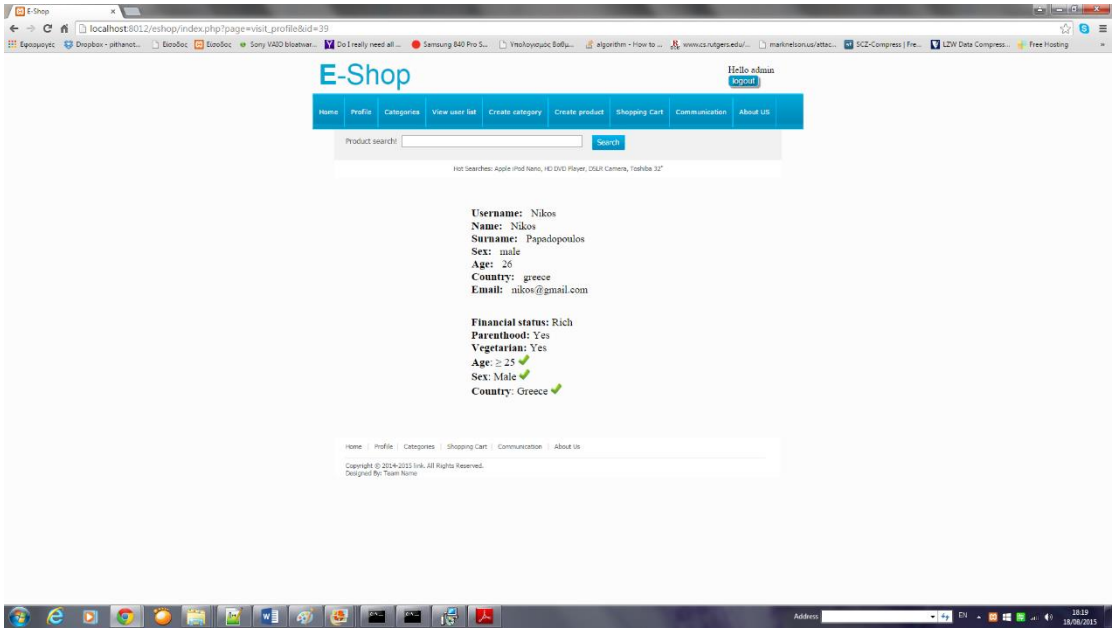
Ο διαχειριστής έχει τη δυνατότητα να πλοηγηθεί στο προφίλ ενός χρήστης. Μέσω της λειτουργίας της πλοήγησης στη λίστα χρηστών, ο διαχειριστής μπορεί να επισκεφτεί το προφίλ ενός χρήστη, πατώντας το αντίστοιχο link. Στην επόμενη εικόνα εμφανίζεται το προφίλ του επιλεγμένου χρήστη.



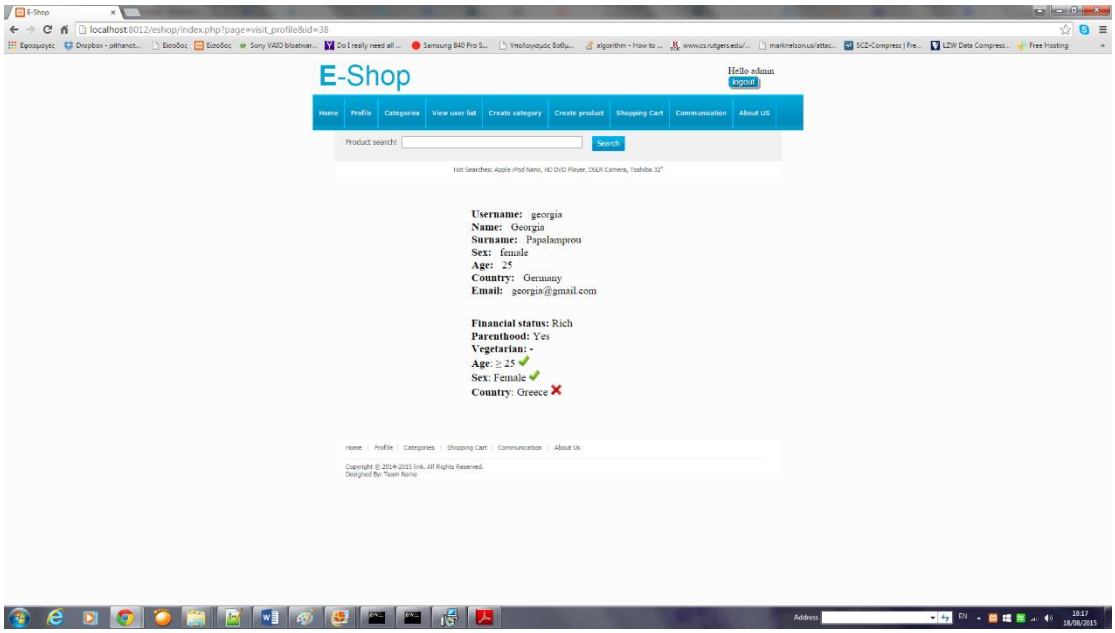
Εικόνα 37: Προβολή προφίλ χρήστη

Ο διαχειριστής, μέσω του κουμπιού «validate info» μπορεί να χρησιμοποιήσει τον profiler της εφαρμογής, όπως αυτός παρουσιάζεται στην ενότητα 10.

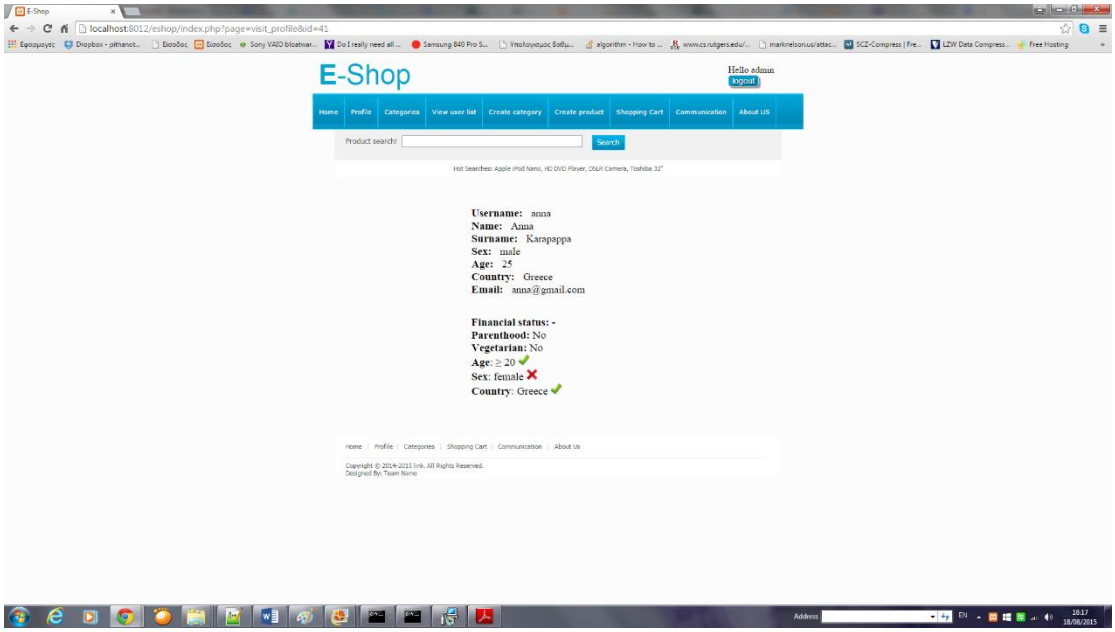
Στην επόμενη εικόνα παρουσιάζονται τα αποτελέσματα που επέστρεψε ο profiler για το χρήστη «Nikos». Το «Financial status» περιγράφει την οικονομική κατάσταση του χρήστη, το «Parenthood» αναφέρεται στην ύπαρξη ή όχι παιδιών και το «Vegetarian» αναφέρεται στο αν ο χρήστης είναι χορτοφάγος ή όχι. Επιπρόσθετα, το «Age» αναφέρει την ελάχιστη ηλικία που προβλέπεται πως έχει ο χρήστης, το «Sex» το προβλεπόμενο φύλο του χρήστη και το «Country» τη χώρα διαμονής του χρήστη. Σε περίπτωση που σε μία από τις επιλογές «Age», «Sex» και «Country» υπάρχει το εικονίδιο ορθής επιβεβαίωσης, τότε η πρόβλεψη συμπίπτει με τα δηλωμένα στοιχεία. Σε αντίθετη περίπτωση, η πρόβλεψη έρχεται σε αντίθεση με τα δηλωμένα στοιχεία.



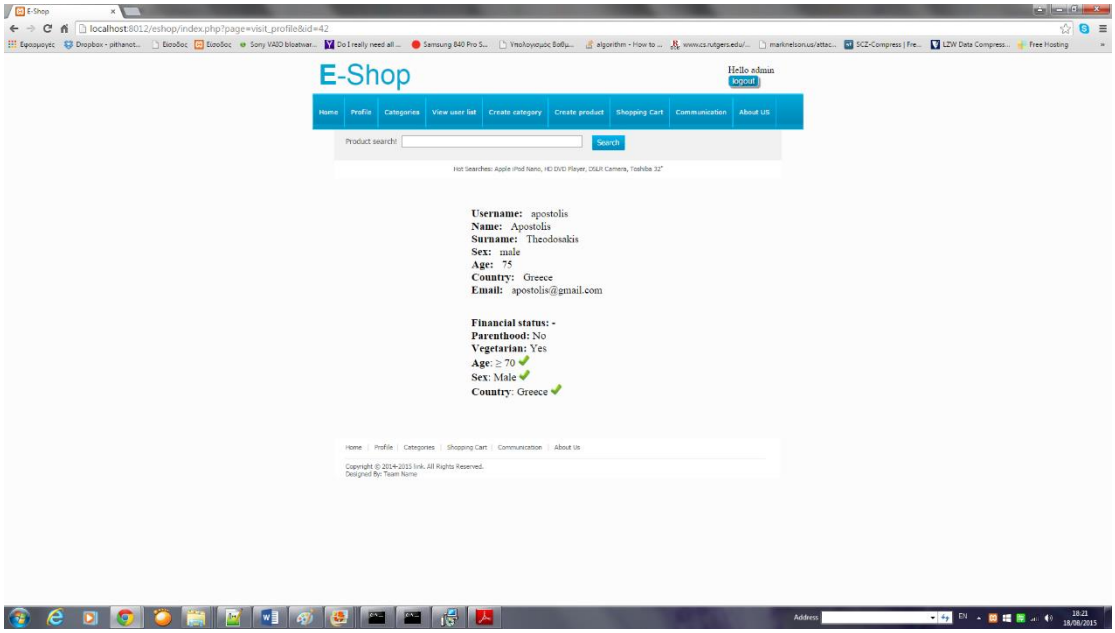
Εικόνα 38: Τα προβλεπόμενα στοιχεία συμπίπτουν με τις εισαχθέντες πληροφορίες



Εικόνα 39: Η προβλεπόμενη χώρα διαμονής έρχεται σε αντίθεση με την εισαχθείσα χώρα



Εικόνα 40: Το προβλεπόμενο φύλο έρχεται σε αντίθεση με το δηλωμένο φύλο



Εικόνα 41: Επιπρόσθετη φωτογραφία όπου τα δηλωμένα στοιχεία συμπίπτουν με τα προβλεπόμενα

13. Βάση Δεδομένων

Στην ενότητα αυτή θα γίνει η περιγραφή της βάσης δεδομένων που χρησιμοποιεί η εφαρμογή τόσο για να αποθηκεύει πληροφορίες σχετικά με τους χρήστες, τα προϊόντα και όλων των υπόλοιπων οντοτήτων, όσο και για να ανακτά τις πληροφορίες αυτές. Η εφαρμογή χρησιμοποιεί ως βάση δεδομένων την MySQL, η οποία αποθηκεύει τα δεδομένα της σε σχεσιακούς πίνακες.

13.1 Script Βάσης

```
-- MySQL Script generated by MySQL Workbench
-- 05/04/15 16:04:58
-- Model: New Model    Version: 1.0
SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS,
FOREIGN_KEY_CHECKS=0;
SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='TRADITIONAL,
ALLOW_INVALID_DATES';

-- -----------------------------------------------------
-- Schema web_profiles
-- -----------------------------------------------------
DROP SCHEMA IF EXISTS `web_profiles`;
CREATE SCHEMA IF NOT EXISTS `web_profiles` DEFAULT CHARACTER SET utf8;
USE `web_profiles`;

-- -----------------------------------------------------
-- Table `web_profiles`. `Users`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `web_profiles`. `users` ;

CREATE TABLE IF NOT EXISTS `web_profiles`. `users` (
  `ID` INT(11) NOT NULL AUTO_INCREMENT,
  `username` VARCHAR(45) NOT NULL,
  `name` VARCHAR(45) NOT NULL,
  `surname` VARCHAR(45) NOT NULL,
  `password` VARCHAR(45) NOT NULL,
  `email` VARCHAR(45) NOT NULL,
  `age` INT(11) NOT NULL,
  `sex` ENUM('male','female') NOT NULL,
  `country` VARCHAR(45) NOT NULL,
  `usertype` ENUM('simple_user','admin') NOT NULL DEFAULT 'simple_user',
  PRIMARY KEY (`ID`),
  UNIQUE INDEX `ID_UNIQUE` (`ID` ASC),
```

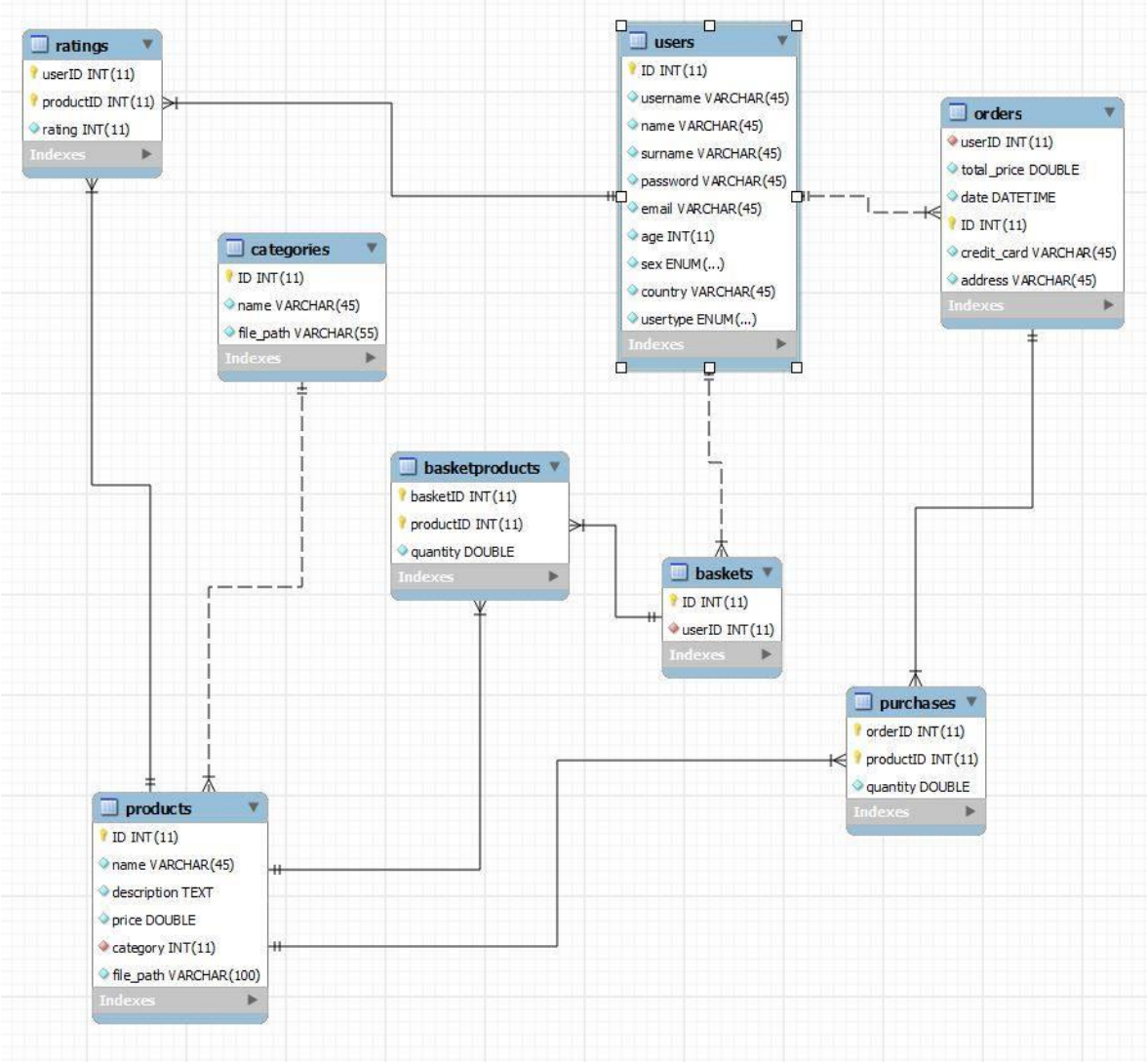
```
  UNIQUE INDEX `username_UNIQUE` (`username` ASC))
ENGINE = InnoDB
AUTO_INCREMENT = 37
DEFAULT CHARACTER SET = utf8;

-- -----------------------------------------------------
-- Insert records in `web_profiles`.`users`
-- -----------------------------------------------------
INSERT INTO
`web_profiles`.`users`(`username`,`name`,`surname`,`password`,`email`,`age`,`sex`,`c
ountry`,`usertype`)
VALUES('admin','Nikos','Papadopoulos','7110eda4d09e062aa5e4a390b0a572ac0d2c
0220', 'admin@gmail.com',32,'male','Greece','admin');

-- -----------------------------------------------------
-- Table `web_profiles`.`baskets`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `web_profiles`.`baskets` ;

CREATE TABLE IF NOT EXISTS `web_profiles`.`baskets` (
  `ID` INT(11) NOT NULL AUTO_INCREMENT,
  `userID` INT(11) NOT NULL,
  PRIMARY KEY (`ID`),
  UNIQUE INDEX `ID_UNIQUE` (`ID` ASC))
ENGINE = InnoDB
AUTO_INCREMENT = 4
DEFAULT CHARACTER SET = utf8;


-- -----------------------------------------------------
-- Table `web_profiles`.`categories`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `web_profiles`.`categories` ;

CREATE TABLE IF NOT EXISTS `web_profiles`.`categories` (
  `ID` INT(11) NOT NULL AUTO_INCREMENT,
  `name` VARCHAR(45) NOT NULL,
  `file_path` VARCHAR(55) NOT NULL,
  PRIMARY KEY (`ID`),
  UNIQUE INDEX `ID_UNIQUE` (`ID` ASC),
  UNIQUE INDEX `name_UNIQUE` (`name` ASC))
ENGINE = InnoDB
AUTO_INCREMENT = 23
DEFAULT CHARACTER SET = utf8;


-- -----------------------------------------------------
```

```
-- Table `web_profiles`.`products`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `web_profiles`.`products` ;

CREATE TABLE IF NOT EXISTS `web_profiles`.`products` (
  `ID` INT(11) NOT NULL AUTO_INCREMENT,
  `name` VARCHAR(45) NOT NULL,
  `description` TEXT NOT NULL,
  `price` DOUBLE NOT NULL,
  `category` INT(11) NOT NULL,
  `file_path` VARCHAR(100) NOT NULL,
  PRIMARY KEY (`ID`),
  UNIQUE INDEX `ID_UNIQUE` (`ID` ASC))
ENGINE = InnoDB
AUTO_INCREMENT = 14
DEFAULT CHARACTER SET = utf8;


-- -----------------------------------------------------
-- Table `web_profiles`.`basketproducts`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `web_profiles`.`basketproducts` ;

CREATE TABLE IF NOT EXISTS `web_profiles`.`basketproducts` (
  `basketID` INT(11) NOT NULL,
  `productID` INT(11) NOT NULL,
  `quantity` DOUBLE NOT NULL,
  PRIMARY KEY (`basketID`, `productID`),
  INDEX `fk_Baskets_has_Products_Baskets1_idx` (`basketID` ASC))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;


-- -----------------------------------------------------
-- Table `web_profiles`.`orders`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `web_profiles`.`orders` ;

CREATE TABLE IF NOT EXISTS `web_profiles`.`orders` (
  `userID` INT(11) NOT NULL,
  `total_price` DOUBLE NOT NULL,
  `date` DATETIME NOT NULL,
  `ID` INT(11) NOT NULL AUTO_INCREMENT,
  `credit_card` VARCHAR(45) NOT NULL,
  `address` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`ID`),
  UNIQUE INDEX `ID_UNIQUE` (`ID` ASC))
```

```
ENGINE = InnoDB
AUTO_INCREMENT = 20
DEFAULT CHARACTER SET = utf8;


-- -----------------------------------------------------
-- Table `web_profiles`.`purchases`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `web_profiles`.`purchases` ;

CREATE TABLE IF NOT EXISTS `web_profiles`.`purchases` (
  `orderID` INT(11) NOT NULL,
  `productID` INT(11) NOT NULL,
  `quantity` DOUBLE NOT NULL,
  PRIMARY KEY (`orderID`, `productID`),
  INDEX `fk_Orders_has_Products_Orders1_idx` (`orderID` ASC))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;


-- -----------------------------------------------------
-- Table `web_profiles`.`ratings`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `web_profiles`.`ratings` ;

CREATE TABLE IF NOT EXISTS `web_profiles`.`ratings` (
  `userID` INT(11) NOT NULL,
  `productID` INT(11) NOT NULL,
  `rating` INT(11) NOT NULL,
  PRIMARY KEY (`userID`, `productID`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;


SET SQL_MODE=@OLD_SQL_MODE;
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;
```

13.2 Επεξήγηση Βάσης Δεδομένων

Στην υπό ενότητα αυτή θα γίνει επεξήγηση των πινάκων της βάσης της εφαρμογής.



ΕΙΚΟΝΑ 1:ΣΧΗΜΑ ΒΑΣΗΣ ΔΕΔΟΜΕΝΩΝ

Στη βάση δεδομένων της εφαρμογής υπάρχουν οι εξής 8 πίνακες:

- users
- categories
- products
- ratings
- baskets
- basketproducts

- orders
- purchases

users: Στον πίνακα αυτό αποθηκεύονται οι πληροφορίες που αφορούν τους χρήστες του συστήματος. Ο πίνακας αυτός έχει 10 πεδία:

- ID: Μοναδικός αριθμός που ταυτοποιεί ένα χρήστη
- username: Όνομα χρήστη στην εφαρμογή
- password: Συνθηματικό χρήστη στην εφαρμογή
- name: Το πραγματικό όνομα χρήστη
- surname: Το επίθετο του χρήστη
- email: Το email του χρήστη
- age: Η ηλικία του χρήστη
- sex: Το φύλο του χρήστη
- country: Η χώρα του χρήστη
- usertype: Το είδος του λογαριασμού του χρήστη στην εφαρμογή(πελάτης ή διαχειριστής)

Ο πίνακας αυτός έχει δύο ευρετήρια:

- ID: Πρωτεύων ευρετήριο
- username: Δευτερεύον ευρετήριο

categories: Στον πίνακα αυτόν αποθηκεύονται οι πληροφορίες που αφορούν τις κατηγορίες προϊόντος του συστήματος. Ο πίνακας αυτός αποτελείται από 3 πεδία:

- ID: Μοναδικός αριθμός που ταυτοποιεί μια κατηγορία προϊόντων
- name: Όνομα της κατηγορίας προϊόντων
- file_path: Όνομα του αρχείου φωτογραφίας που αντιστοιχεί στη συγκεκριμένη κατηγορία προϊόντων

Ο πίνακας αυτός έχει δύο ευρετήρια:

- ID: πρωτεύον ευρετήριο
- name: Δευτερεύον ευρετήριο

products: Στον πίνακα αυτόν αποθηκεύονται οι πληροφορίες που αφορούν τα προϊόντα της εφαρμογής. Το table αυτό αποτελείται από 6 πεδία:

- ID: Μοναδικός αριθμός που ταυτοποιεί ένα προϊόν
- name: Όνομα του προϊόντος
- description: Περιγραφή του προϊόντος
- price: Τιμή προϊόντος
- file_path: Όνομα του αρχείου φωτογραφίας που αντιστοιχεί στο συγκεκριμένο προϊόν
- category: Ξένο κλειδί προς το ID της κατηγορίας προϊόντος που ανήκει το συγκεκριμένο προϊόν

Ο πίνακας αυτός έχει δύο ευρετήρια:

- ID(Πρωτεύον ευρετήριο)
- category(Δευτερεύον ευρετήριο)

ratings: Στον πίνακα αυτό φυλάσσονται οι πληροφορίες σχετικά με τις βαθμολογίες που έχουν δώσει οι χρήστες σε προϊόντα. Ο πίνακας αυτός έχει 3 πεδία:

- userID: Ξένο κλειδί προς ID του χρήστη που έδωσε τη βαθμολογία
- productID: Ξένο κλειδί προς το ID του προϊόντος που βαθμολογήθηκε
- rating: Η βαθμολογία που δόθηκε

Ο πίνακας αυτός έχει τρία ευρετήρια:

- userID+productID(Πρωτεύον ευρετήριο)
- userID(Δευτερεύον ευρετήριο)
- productID(Δευτερεύον ευρετήριο)

baskets: Ο πίνακας αυτός αποθηκεύει τις βασικές πληροφορίες που αφορούν το καλάθι αγορών των χρηστών. Ο πίνακας αυτός αποτελείται από 2 πεδία:

- ID: Μοναδικός αριθμός που προσδιορίζει το καλάθι αγορών ενός χρήστη
- userID: Ξένο κλειδί προς το ID του χρήστη στον οποίο ανήκει το καλάθι αγορών

Ο πίνακας αυτός έχει δύο ευρετήρια:

- ID(Πρωτεύον ευρετήριο)
- userID(Δευτερεύον ευρετήριο)

basketproducts: Στον πίνακα αυτό αποθηκεύονται οι πληροφορίες σχετικά με τα περιεχόμενα των καλαθιών αγορών των χρηστών. Ο πίνακας αυτός έχει 3 πεδία:

- basketID: Ξένο κλειδί προς το ID του καλαθιού αγορών που ανήκει το προϊόν
- productID: Ξένο κλειδί προς το ID του προϊόντος που βρίσκεται μέσα στο καλάθι αγορών
- quantity: Η ποσότητα στην οποία βρίσκεται το προϊόν μέσα στο καλάθι αγορών

Ο πίνακας αυτός χρησιμοποιεί τρία ευρετήρια:

- basketID+productID(Πρωτεύον ευρετήριο)
- basketID(Δευτερεύον ευρετήριο)
- productID(Δευτερεύον ευρετήριο)

orders: Στον πίνακα αυτό αποθηκεύονται οι βασικές πληροφορίες που αφορούν τις παραγγελίας που έχουν κάνει οι χρήστες του συστήματος. Ο πίνακας αυτός αποτελείται από 6 πεδία:

- ID: Μοναδικός αριθμός που χαρακτηρίζει μια και μοναδική παραγγελία ενός πελάτη
- userID: Ξένο κλειδί προς το ID του χρήστη, ο οποίος έκανε τη παραγγελία
- total_price: Συνολικό κόστος της παραγγελίας που έκανε ο πελάτης, λαμβάνοντας υπόψη τόσο τις τιμές των προϊόντων που βρίσκονταν στο καλάθι των αγορών, όσο και των αντίστοιχών ποσοτήτων
- date: Ημερομηνία και ώρα όπου έγινε η παραγγελία των προϊόντων
- credit_card: Αριθμός πιστωτικής κάρτας
- address: Διεύθυνση αποστολής της παραγγελίας

Ο πίνακας αυτός έχει δύο ευρετήρια:

- ID(Πρωτεύον ευρετήριο)
- userID(Δευτερεύον ευρετήριο)

purchases: Στον πίνακα αυτό αποθηκεύονται τα προϊόντα των παραγγελιών. Ο πίνακας αυτός αποτελείται από 3 πεδία:

- orderID: Ξένο κλειδί προς το ID της παραγγελίας
- productID: Ξένο κλειδί προς το ID του προϊόντος που έχει παραγγελθεί
- quantity: Η ποσότητα στην οποία παραγγέλθηκε το προϊόν

Ο πίνακας αυτός αποτελείται από 3 ευρετήρια:

- orderID+productID(Πρωτεύον ευρετήριο)
- productID(Δευτερεύον ευρετήριο)
- orderID(Δευτερεύον ευρετήριο

14. Ασφάλεια Εφαρμογής

Μια πρώτη μορφή ασφάλειας που παρέχετε από την εφαρμογή είναι η κρυπτογράφηση του κωδικού των χρηστών. Το password είναι ένα ευαίσθητο προσωπικό δεδομένο, καθώς μπορεί να αποτελεί τον κωδικό πρόσβαση και σε ένα άλλο σύνολο από λογαριασμούς του χρήστη σε διαφορετικές εφαρμογές. Για το λόγο αυτό, ο χρήστης επιθυμεί την απόκρυψη του, ακόμη και από τους διαχειριστές της Βάσης Δεδομένων της εφαρμογής. Για το λόγο αυτό, γίνετε κρυπτογράφηση του κωδικού πρόσβασης, με τη χρήση της συνάρτησης sha1. Με τον τρόπο αυτό, αρχικά επιτυγχάνεται η αυθεντικοποίηση του κωδικού πρόσβασης του χρήστη στη διαδικασία της «Σύνδεσης Χρήστης» στην εφαρμογή. Ο λόγος είναι ότι η sha1 είναι μια συνάρτηση 1-1, το οποίο σημαίνει ότι μια είσοδος θα έχει πάντα την ίδια έξοδο, και ότι είναι αρκετά μικρή η πιθανότητα μια διαφορετική είσοδος να παράγει την ίδια έξοδο με μία άλλη. Κατά δεύτερον επιτυγχάνεται η απόκρυψη του πραγματικού κωδικού του χρήστη, καθώς στη βάση αποθηκεύεται μια κρυπτογραφημένη μορφή αυτού.

Μια δεύτερη μορφή προστασίας, η οποία παρέχεται από την εφαρμογή είναι η άμυνα απέναντι σε SQL injection επιθέσεις. Μέσω της επίθεσης αυτής, ο επιτιθέμενος προσπαθεί να τροποποιήσει τις SQL εντολές που θα τρέξουν στο server, με στόχο είτε να αποσπάει ευαίσθητες πληροφορίες είτε να αποκτήσει πρόσβαση σε δεδομένα/σελίδες που δεν έχει δικαίωμα. Για την προστασία ενάντια σε αυτού του είδους επιθέσεων χρησιμοποιηθήκαν τα prepare statements. Μέσω της χρήσης των prepare statements η ανάλυση και η σύνθεση του επερωτήματος γίνεται πριν έρθουν τα δεδομένα που εισήγαγε ο χρήστης. Με τον τρόπο αυτό επιτυγχάνεται η μη τροποποίηση της μορφής της SQL δήλωσης από τα δεδομένα που εισήγαγε ο χρήστης.

Μια τρίτη και τελευταία μορφή προστασίας είναι απέναντι σε επιθέσεις Cross Site Scripting(XSS). Η επίθεση αυτή αναφέρεται στην προσπάθεια εκμετάλλευσης διαφόρων ευπαθειών των υπολογιστικών συστημάτων σε εισαγωγή κώδικα HTML ή javascript σε κάποιο ιστοχώρο.

Μέσω της επίθεσης XSS, ο επιτιθέμενος μπορεί να πετύχει:

- Κλοπή κωδικών ή άλλων προσωπικών δεδομένων
- Αλλαγή ρυθμίσεων του ιστοχώρου
- Κλοπή των cookies
- Ψεύτικη διαφήμιση

Η παραπάνω ευπάθεια αναφέρεται στην αδυναμία του συστήματος που υποστηρίζει ο ιστόχωρος να φιλτράρει και να απορρίψει τυχόν επιβλαβείς εισόδους. Για την προστασία απέναντι σε τέτοιου είδους επιθέσεις, το σύστημα φιλτράρει τα εξωτερικά δεδομένα, με στόχο την τροποποίηση των ειδικών χαρακτήρων της HTML και κυρίως της Javascript ώστε να είναι ακίνδυνα για την ομαλή λειτουργίας της εφαρμογής.

15. Επίδοση εφαρμογής

Αρχικά χρησιμοποιήθηκε η τεχνολογία του AJAX [5] για τη δημιουργία ασύγχρονων μεθόδων για την κλήση δεδομένων από τη Βάση Δεδομένων. Με τον τρόπο αυτό επιτυγχάνεται η διαδραστικότητα του χρήστη με την εφαρμογή, δίχως να γίνεται διακοπή κάθε φορά που ζητούνται δεδομένα από τη βάση, έως ότου αυτά να επιστραφούν.

Επιπρόσθετα γίνεται χρήση των prepared statements [6], τα οποία αυξάνουν την απόδοση της MySQL βάσης, καθώς το overhead για τη μεταγλώττισης και τη βελτιστοποίηση ενός statement λαμβάνει χώρα μόνο μια φορά, παρόλο που το ίδιο θα μπορεί να εκτελεστεί παραπάνω από μία φορά.

Μία ακόμη τεχνική για τη βελτιστοποίηση της απόδοσης της εφαρμογής είναι η χρήση ευρετηρίων στη βάση δεδομένων. Μέσω της χρήσης ευρετηρίων μπορεί να μειωθεί αρκετά ο χρόνος απόκρισης της βάσης δεδομένων σε ερωτήματα ανάκτησης, καθώς μπορεί να εντοπίσει γρηγορότερα τα δεδομένα που χρειάζεται. Με τον τρόπο αυτό μειώνεται το σύνολο των δεδομένων που πρέπει να επεξεργαστούν, καθώς και το σύνολο των blocks που θα φέρει η βάση από το σκληρό δίσκο προς τη μνήμη.

Τέλος, η εφαρμογή αποθηκεύει στη μνήμη του υπολογιστή τα βασικά δεδομένα του ενεργού χρήστη, όπως είναι το ID, το username και ο τύπος χρήστη. Τα στοιχεία αυτά(και ιδιαίτερα το ID του χρήστη) χρειάζονται σχεδόν σε κάθε ενέργεια που κάνει ο χρήστης. Μέσω της αποθήκευσης τους στη μνήμη του υπολογιστή, αποφεύγεται η εκτέλεση περιττών επερωτημάτων στη βάση δεδομένων, προκειμένου να ανακτηθούν τα δεδομένα αυτά.

Πίνακας Ορολογίας

Marketing	Προώθηση προϊόντων
Online	Πληροφορία διαθέσιμη στο διαδίκτυο
Recommender systems	Συστήματα σύστασης προτάσεων
Recommendation systems	Συστήματα σύστασης προτάσεων
Client	Πελάτης
Server	Εξυπηρετητής
Interface	Διεπαφή
Table	Πίνακας
Browser	Φυλλομετρητής
Username	Όνομα χρήστη
Password	Κωδικός πρόσβασης
Up to date	Ενημερωμένη
Administrator	Διαχειριστής
Format	Μορφή
Blocks	Μπλοκ δίσκου
Injection	Μόλυνση
Interpreter	Διερμηνευτής
Link	Σύνδεσμος

Συντμήσεις – Αρκτικόλεξα - Ακρώνυμα

WSO	Weighted Slope One
ΒΔ	Βάση δεδομένων
MVC	Model-View-Controller

Αναφορές

[1] http://www.uie.com/articles/recommendation_systems/

[2] http://dide.flo.sch.gr/Plinet/Tutorials/Tutorials-DataBasesTheory.html

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

[4] Danilo Meneze, Anisio Lacerda, Leila Silva, Adriano Veloso, Nivio Ziviani, «Weighted slope one predictors revisited» 13 May 2013

[5] https://developer.mozilla.org/en-US/docs/AJAX/Getting_Started

[6] http://en.wikipedia.org/wiki/Prepared_statement

Code Appendix

1. Κλάση WeightedSlopeOne

Η κλάση αυτή προσφέρει τη λειτουργικότητα του Weighted Slope One αλγόριθμου για τη σύσταση προτάσεων στους συνδεδεμένους χρήστης.

```
<?php

require_once('Database.php');
require_once('Category.php');
require_once('Product.php');

Class WeightedSlopeOne {

        public $deviations;
        public $cards;
        public $ratings;
        public $numberOfItems;
        public $numberOfUsers;

        public function __construct($ratings = '', $categoryID) {

                ini_set('memory_limit', '-1');

                $this->ratings = $ratings;
```

```
                $this->deviations = array();
                $this->cards = array();

                $connection = Database::createNewConnection();
                $query = "SELECT COUNT(*) AS `users` FROM Users";
                $result = $connection->prepare($query);
                $result->execute();
                $result->store_result();
                $result->bind_result($this->numberOfUsers);
                $result->fetch();

                $query = "SELECT COUNT(*) AS `items` FROM Products WHERE
`category`=?";
                $result = $connection->prepare($query);
                $result->bind_param('i', $categoryID);
                $result->execute();
                $result->store_result();
                $result->bind_result($this->numberOfItems);
                $result->fetch();

                for ($item1 = 0; $item1 < $this->numberOfItems; $item1++) {
                        for ($item2 = 0; $item2 < $this->numberOfItems; $item2++) {
                                $this->cards[$item1][$item2] = 0;
                                $this->deviations[$item1][$item2] = 0;
                        }
                }
        }

        public static function predict($recommendedUser, $categoryID) {

                $connection = Database::createNewConnection();

                $query = "SELECT Datas.`userID` AS `userID`, Datas.`productID` AS
`productID`, IFNULL(`rating`, -1) AS `rating`
                                        FROM (SELECT Users.`ID` AS `userID`, Products.`ID` AS
`productID`, `category`
                                                FROM Users, Products WHERE `category`=?) AS
Datas LEFT JOIN(SELECT `rating`, `productID`, `userID`

                                                FROM Ratings) AS Ratings

                        ON Ratings.`productID`=Datas.`productID` AND
Ratings.`userID`=Datas.`userID`
```

```
                ORDER BY Datas.`userID`, Datas.`productID`";

$result = $connection->prepare($query);
$result->bind_param('i', $categoryID);
$result->execute();
$result->store_result();
$result->bind_result($userID, $productID, $rating);

$mapping = array();
$products = array();
$AI = 0;
$newProducts = true;
$ratings = array();
$counter = 0;
$previousUser = -1;

$enter = false;
while($result->fetch()) {

        $enter = true;

        if($previousUser == -1){
                $ratings[$counter] = array();
                $mapping[$userID] = $counter;
        } else if($userID != $previousUser) {
                $newProducts = false;
                $counter++;
                $ratings[$counter] = array();
                $mapping[$userID] = $counter;
        }

        if($newProducts == true){
                $products[$AI] = $productID;
                $AI++;
        }

        array_push($ratings[$counter], $rating);

        $previousUser = $userID;
}
```

```
		if($enter == false)
			return null;

		$wso = new WeightedSlopeOne($ratings, $categoryID);
		$res = $wso->predictRatings($mapping[$recommendedUser]);
		arsort($res);

		$query = "SELECT `ID`
				FROM Products
				WHERE `category`=? AND `ID` NOT IN (SELECT `productID`
FROM Purchases, Orders

		 WHERE `orderID`=Orders.`ID` AND `userID`=?)";

		$result = $connection->prepare($query);
		$result->bind_param('ii', $categoryID, $recommendedUser);
		$result->execute();
		$result->store_result();
		$result->bind_result($productID);

		$availableProductList = array();
		while($result->fetch()){
			array_push($availableProductList, $productID);
		}

		$foundRecommenderProducts = 0;
		$recommendedProducts = array();
		foreach($res as $key => $rating){
			if($rating == 0) {
				break;
			}

			if(in_array($products[$key], $availableProductList)){
				$recommendedProducts[$key] = $rating;
				$foundRecommenderProducts++;
				if($foundRecommenderProducts == 3) {
					break;
				}
			}
		}

		$productList = array();
```

```
                foreach($recommendedProducts as $key => $rating){

                        if($rating == -1)
                                break;

                        $query = "SELECT Products.`ID` AS `productID`, Products.`name` AS
`productName`, `description`, `price`,
                                                Categories.`name` AS `categoryName`,
Categories.`ID` AS `categoryID`, Products.`file_path`
                                FROM Products, Categories
                                WHERE Categories.`ID`=Products.`category` AND
Products.`ID`=?";

                        $result = $connection->prepare($query);
                        $result->bind_param('i', $products[$key]);

                        $result->execute();

                        $result->store_result();

                        $result->bind_result($productID, $productName, $description,
$price, $categoryName, $categoryID, $filePath);

                        $result->fetch();

                        $category = new ModelCategory($categoryID, $categoryName,
null);


                        $product = new ModelProduct($productID, $productName,
$description, $price, $category, $filePath);

                        array_push($productList, $product);
                }

                return $productList;
        }


        public function algorithm() {

                $this->computeDeviations();
                $this->makePredictions();
```

```
		return $this->ratings;
	}

	public function predictRatings($user) {

		$this->computeDeviations();
		$this->predictUserRating($user);

		return $this->ratings[$user];
	}

	private function computeDeviations() {

		for($item1=0; $item1<$this->numberOfItems; $item1++) {
			for($item2=$item1 + 1; $item2<$this->numberOfItems; $item2++) {
				for ($user = 0; $user < $this->numberOfUsers; $user++) {
					if (($this->ratings[$user][$item1] != -1) && ($this-
>ratings[$user][$item2] != -1)) {
						$this->cards[$item1][$item2]++;
						$this->cards[$item2][$item1]++;
					}
				}
			}
		}

		for ($item1 = 0; $item1 < $this->numberOfItems; $item1++) {
			for ($item2 = $item1 + 1; $item2 < $this->numberOfItems;
$item2++) {

				if ($item1 == $item2) {
					continue;
				}
				for ($user = 0; $user < $this->numberOfUsers; $user++) {
					if (($this->ratings[$user][$item1] != -1)
						&& ($this->ratings[$user][$item2] != -1)) {

						$this->deviations[$item1][$item2] += ($this-
>ratings[$user][$item1] -

						 $this->ratings[$user][$item2]) / $this-
>cards[$item1][$item2];
```

```
                                        $this->deviations[$item2][$item1] += ($this-
>ratings[$user][$item2] -

                                         $this->ratings[$user][$item1]) / $this-
>cards[$item1][$item2];
                                    }
                                }
                            }
                        }
    }

        private function makePredictions() {

                for ($user = 0; $user < $this->numberOfUsers; $user++) {
                        for ($item1 = 0; $item1 < $this->numberOfItems; $item1++) {

                                if ($this->ratings[$user][$item1] != -1) {
                                        continue;
                                }

                                $prediction = 0;
                                $sumCards = 0;
                                for ($item2 = 0; $item2 < $this->numberOfItems; $item2++)
{
                                        if ($this->ratings[$user][$item2] != -1){
                                                $prediction+=($this-
>ratings[$user][$item2]+$this->deviations[$item1][$item2])*$this->cards[$item1][$item2];
                                                $sumCards+=$this->cards[$item1][$item2];
                                        }
                                }
                                $prediction/=$sumCards;
                                $this->ratings[$user][$item1] = $prediction;
                        }
                }
        }

        private function predictUserRating($user) {

                for ($item1 = 0; $item1 < $this->numberOfItems; $item1++) {

                        if ($this->ratings[$user][$item1] != -1)
                                continue;
```

```
                        $prediction = 0;
                        $sumCards = 0;
                        for ($item2 = 0; $item2 < $this->numberOfItems; $item2++) {
                                if ($this->ratings[$user][$item2] != -1){
                                        $prediction+=($this->ratings[$user][$item2]+$this-
>deviations[$item1][$item2])*$this->cards[$item1][$item2];
                                        $sumCards+=$this->cards[$item1][$item2];
                                }
                        }

                        if($sumCards != 0)
                                $prediction/=$sumCards;
                        else
                                $prediction = -1;

                        $this->ratings[$user][$item1] = $prediction;
                }
        }
}
?>
```

2. Κλάση SimpleRecommender

Στη κλάση αυτή έχει υλοποιηθεί η λειτουργικότητα της σύστασης προτάσεων με βάση τη δημοτικότητα των προϊόντων ανά κατηγορία.

```
<?php


Class SimpleRecommender {

        public static function
getSimpleRecommenderProductsByCategory($categoryID, $userID) {

                $connection = Database::createNewConnection();

                $query = "SELECT `name` FROM Categories WHERE `ID`=?";

                $result = $connection->prepare($query);

                $result->bind_param('i', $categoryID);
```

```
		$result->execute();

		$result->store_result();

		$result->bind_result($categoryName);

		$result->fetch();


		$query = "SELECT products.`id`, products.`name`, `description`,
`price`, products.`file_path`,
					categories.`ID`, categories.`Name`
				FROM products, purchases, categories
				WHERE products.`id`=`productID` AND
`category`=categories.`id` AND categories.`name`=?
					AND Products.`ID` NOT IN (SELECT
DISTINCT `productID` FROM Purchases, Orders, Users

		WHERE `orderID`=Orders.`id` AND Orders.userID=Users.`ID`
AND Users.`ID`=?)
				GROUP BY products.`id`
				ORDER BY count(*) DESC
				LIMIT 3";

		$result = $connection->prepare($query);

		$result->bind_param('si', $categoryName, $userID);

		$result->execute();

		$result->store_result();

		$result->bind_result($productID, $productName, $description,
$price, $filePath, $categoryID, $categoryName);

		$products = array();
		while($result->fetch()){

			$category = new ModelCategory($categoryID,
$categoryName, null);
```

```
                        $product = new ModelProduct($productID,
$productName, $description, $price, $category, $filePath);

                        array_push($products, $product);
                }

                return $products;
        }



}


?>
```

3. Κλάση Exporter

Η κλάση αυτή υλοποιεί τη λειτουργία του profiler της εφαρμογής

```
<?php

require_once('Database.php');
require_once('User.php');

Class Exporter {

        public $userID;
        public $connection;

        public function __construct($userID) {
                $this->userID = $userID;
                $this->connection = Database::createNewConnection();
        }

        public function exportParent(){

                $query = "SELECT COUNT(*)
                                FROM Products, Orders, Purchases, Categories
                                WHERE `userID` = ? AND
Orders.`ID`=Purchases.`orderID` AND Purchases.`productID`=Products.`ID`
```

```
                                AND category=Categories.`ID` AND
                                (Categories.`name`='Baby Food' OR
Categories.`name`='Baby Products' OR Categories.`name`='Snack')";

                $result = $this->connection->prepare($query);
                $result->bind_param('i', $this->userID);
                $result->execute();
                $result->store_result();
                $result->bind_result($count);
                $result->fetch();

                if($count > 0)
                        return "yes";
                else{


                        $query = "SELECT SUM(`quantity`) AS `quantity`
                                        FROM Orders, Purchases, Products,
Categories
                                        WHERE `userID`=? AND
`orderID`=Orders.`ID` AND `productID`=Products.`ID`
                                        AND `category`=Categories.ID AND
Categories.`name`='Milks'
                        GROUP BY `orderID`";

                                $result = $this->connection->prepare($query);
                                $result->bind_param('i', $this->userID);
                                $result->execute();
                                $result->store_result();
                                $result->bind_result($count);
                                $result->fetch();

                                if($count >= 4)
                                        return "yes";
                                else
                                        return "no";
                }
        }

        public function exportVegetarian(){
```

```
		$buyMeat = false;
		$buyVegetarianProducts = false;

		$query = "SELECT COUNT(*)
				FROM Products, Orders, Purchases, Categories
				WHERE `userID` = ? AND
Orders.`ID`=Purchases.`orderID` AND Purchases.`productID`=Products.`ID`
				AND category=Categories.`ID` AND
				(Categories.`name`='Meats' OR
Categories.`name`='Milks' OR Categories.`name`='Yogurts')";

		$result = $this->connection->prepare($query);
		$result->bind_param('i', $this->userID);
		$result->execute();
		$result->store_result();
		$result->bind_result($count);
		$result->fetch();

		if($count > 0)
			$buyMeat = true;

		$query = "SELECT COUNT(*)
				FROM Products, Orders, Purchases, Categories
				WHERE `userID` = ? AND
Orders.`ID`=Purchases.`orderID` AND Purchases.`productID`=Products.`ID`
				AND category=Categories.`ID` AND
				(Categories.`name`='Vegetables' OR
Categories.`name`='Legumes' OR Categories.`name`='Snack')";

		$result = $this->connection->prepare($query);
		$result->bind_param('i', $this->userID);
		$result->execute();
		$result->store_result();
		$result->bind_result($count);
		$result->fetch();

		if($count > 0)
			$buyVegetarianProducts = true;

		if($buyVegetarianProducts == false && $buyMeat == false)
			return null;
```

```
		else if($buyMeat == false)
			return "yes";
		else
			return "no";

	}

	public function exportSex(){

		$sex = null;

		$query = "SELECT COUNT(*)
				FROM Products, Orders, Purchases, Categories
				WHERE `userID` = ? AND
Orders.`ID`=Purchases.`orderID` AND Purchases.`productID`=Products.`ID`
				AND category=Categories.`ID` AND
(Categories.`name`='Kitchen Items'
				OR Categories.`name`='Decorative House' OR
Categories.`name`='Cosmetics' OR Categories.`name`='Toiletries'
				OR Categories.`name`='Women Products')";


		$result = $this->connection->prepare($query);
		$result->bind_param('i', $this->userID);
		$result->execute();
		$result->store_result();
		$result->bind_result($count);
		$result->fetch();

		if($count > 0)
			$sex = "female";

		$query = "SELECT COUNT(*)
				FROM Products, Orders, Purchases, Categories
				WHERE `userID` = ? AND
Orders.`ID`=Purchases.`orderID` AND Purchases.`productID`=Products.`ID`
				AND category=Categories.`ID` AND
(Categories.`name`='Mens Products' OR Categories.`name`='Frozen Food'
				OR Categories.`name`='Automotive Products')";

		$result = $this->connection->prepare($query);
		$result->bind_param('i', $this->userID);
```

```
                $result->execute();
                $result->store_result();
                $result->bind_result($count);
                $result->fetch();

                if($count > 0){
                        if($sex == null)
                                $sex = "male";
                        else
                                $sex = null;
                }

                return $sex;
        }

        public function exportAge(){


                $query = "SELECT COUNT(*)
                                FROM Products, Orders, Purchases, Categories
                                WHERE `userID` = ? AND
Orders.`ID`=Purchases.`orderID` AND Purchases.`productID`=Products.`ID`
                                AND category=Categories.`ID` AND
Categories.`name`='Elderly Products'";

                $result = $this->connection->prepare($query);
                $result->bind_param('i', $this->userID);
                $result->execute();
                $result->store_result();
                $result->bind_result($count);
                $result->fetch();

                if($count > 0)
                        return 70;

                $query = "SELECT COUNT(*)
                                FROM Products, Orders, Purchases, Categories
                                WHERE `userID` = ? AND
Orders.`ID`=Purchases.`orderID` AND Purchases.`productID`=Products.`ID`
                                AND category=Categories.`ID` AND
```

```
                                (Categories.`name`='Snack' OR
Categories.`name`='Baby Food' OR Categories.`name`='Baby Products')";


                $result = $this->connection->prepare($query);
                $result->bind_param('i', $this->userID);
                $result->execute();
                $result->store_result();
                $result->bind_result($count);
                $result->fetch();

                if($count > 0)
                        return 25;
                else{

                        $query = "SELECT SUM(`quantity`) AS `quantity`
                                        FROM Orders, Purchases, Products,
Categories
                                        WHERE `userID`=? AND
`orderID`=Orders.`ID` AND `productID`=Products.`ID`
                                        AND `category`=Categories.ID AND
Categories.`name`='Milks'
                         GROUP BY `orderID`";

                        $result = $this->connection->prepare($query);
                        $result->bind_param('i', $this->userID);
                        $result->execute();
                        $result->store_result();
                        $result->bind_result($count);
                        $result->fetch();

                        if($count >= 4)
                                return 25;
                }

                $query = "SELECT COUNT(*)
                                FROM Products, Orders, Purchases, Categories
                                WHERE `userID` = ? AND
Orders.`ID`=Purchases.`orderID` AND Purchases.`productID`=Products.`ID`
                                AND category=Categories.`ID` AND
Categories.`name`='Decorative House'";
```

```
		$result = $this->connection->prepare($query);
		$result->bind_param('i', $this->userID);
		$result->execute();
		$result->store_result();
		$result->bind_result($count);
		$result->fetch();

		if($count > 0)
			return 20;

		$query = "SELECT COUNT(*)
				FROM Products, Orders, Purchases, Categories
				WHERE `userID` = ? AND
Orders.`ID`=Purchases.`orderID` AND Purchases.`productID`=Products.`ID`
				AND category=Categories.`ID` AND
				(Categories.`name`='Kitchen Items' OR
Categories.`name`='Cleaning Items' OR Categories.`name`='Vegetables'
				OR Categories.`name`='Legumes' OR
Categories.`name`='Automotive Products'
				OR Categories.`name`='Erotic Accessories' OR
Categories.`name`='Frozen Food')";

		$result = $this->connection->prepare($query);
		$result->bind_param('i', $this->userID);
		$result->execute();
		$result->store_result();
		$result->bind_result($count);
		$result->fetch();

		if($count > 0)
			return 18;
		else{
			$query = "SELECT MAX(`quantity`) AS `quantity` FROM
(
					SELECT
IFNULL(SUM(`quantity`),0) AS `quantity`
					FROM Orders,
Purchases, Products, Categories
					WHERE `userID`=?
AND `orderID`=Orders.`ID` AND `productID`=Products.`ID`
						AND
`category`=Categories.ID AND Categories.`name`='Nuts'
```

```
                                                    GROUP BY `orderID`
                                                ) AS T ";

                $result = $this->connection->prepare($query);
                $result->bind_param ('i', $this->userID);
                $result->execute();
                $result->store_result();
                $result->bind_result($count);
                $result->fetch();

                if($count >=4)
                        return 18;
        }

        $query = "SELECT COUNT(*)
                        FROM Products, Orders, Purchases, Categories
                        WHERE `userID` = ? AND
Orders.`ID`=Purchases.`orderID` AND Purchases.`productID`=Products.`ID`
                        AND category=Categories.`ID` AND
(Categories.`name`='Drinks' OR Categories.`name`='Mens Products')";


        $result = $this->connection->prepare($query);
        $result->bind_param('i', $this->userID);
        $result->execute();
        $result->store_result();
        $result->bind_result($count);
        $result->fetch();

        if($count > 0)
                return 16;

        return 0;
}

public function exportRich(){

        $isRich = false;
        $isPoor = false;

        $query = "SELECT COUNT(*)
```

```
                        FROM Products, Orders, Purchases, Categories
                        WHERE `userID` = ? AND
Orders.`ID`=Purchases.`orderID` AND Purchases.`productID`=Products.`ID`
                        AND category=Categories.`ID` AND
Categories.`name`='Chinese Cuisine'";

        $result = $this->connection->prepare($query);
        $result->bind_param('i', $this->userID);
        $result->execute();
        $result->store_result();
        $result->bind_result($count);
        $result->fetch();

        if($count > 0)
                $isRich = true;
        else{
                $query = "SELECT MAX(`total_price`)
                                FROM orders
                                WHERE `userid`=?";

                $result = $this->connection->prepare($query);
                $result->bind_param('i', $this->userID);
                $result->execute();
                $result->store_result();
                $result->bind_result($maxPrice);
                $result->fetch();

                if($maxPrice > 150)
                        $isRich = true;
                else{

                        $query = "SELECT COUNT(*)
                                        FROM Purchases, Orders
                                        WHERE `userID`=? AND
Orders.`ID`=`orderID` AND `productID` IN (SELECT `ID`
                                                        FROM Products
                                                        WHERE `price` =
(SELECT MAX(`price`) FROM Products i WHERE i.`category` =
Products.`category`))";

                        $result = $this->connection->prepare($query);
                        $result->bind_param('i', $this->userID);
```

```
                        $result->execute();
                        $result->store_result();
                        $result->bind_result($count);
                        $result->fetch();

                        if($count > 0)
                                $isRich = true;
                        else{

                                $query = "SELECT MAX(`quantity`) AS
`quantity` FROM (
                                                SELECT
IFNULL(SUM(`quantity`),0) AS `quantity`
                                                FROM Orders,
Purchases, Products, Categories
                                                WHERE `userID`=?
AND `orderID`=Orders.`ID` AND `productID`=Products.`ID`
                                                        AND
`category`=Categories.ID AND (Categories.`name`='Yogurts'
                                                        OR
Categories.`name`='Meats' OR Categories.`name`='Nuts')
                                                GROUP BY `orderID`
                                        ) AS T ";

                                $result = $this->connection->prepare
($query);
                                $result->bind_param('i', $this->userID);
                                $result->execute();
                                $result->store_result();
                                $result->bind_result($count);
                                $result->fetch();

                                if($count >=4)
                                        $isRich = true;
                        }
                }
        }


        $query = "SELECT COUNT(*)
                        FROM Purchases, Orders
```

```
                        WHERE `userID`=? AND Orders.`ID`=`orderID` AND
`productID` IN (SELECT `ID`
                                                FROM Products
                                                WHERE `price` = (SELECT
MIN(`price`) FROM Products i WHERE i.`category` = Products.`category`))";

                $result = $this->connection->prepare($query);
                $result->bind_param('i', $this->userID);
                $result->execute();
                $result->store_result();
                $result->bind_result($count);
                $result->fetch();

                if($count > 0)
                                $isPoor = true;

                if($isRich == false && $isPoor == true)
                        return "no";
                else if($isRich == true && $isPoor == false)
                        return "yes";
                else
                        return null;

        }

        public function exportCountry(){

                $query = "SELECT COALESCE(`founded_country`, '') FROM
Users WHERE `ID`=?";

                $result = $this->connection->prepare($query);
                $result->bind_param('i', $this->userID);
                $result->execute();
                $result->store_result();
                $result->bind_result($country);
                $result->fetch();

                return $country;
        }

        public function export(){
```

```
                $sex = $this->exportSex();
                $isVegetarian = $this->exportVegetarian();
                $isParent = $this->exportParent();
                $age = $this->exportAge();
                $isRich = $this->exportRich();
                $country = $this->exportCountry();

                $user = new ExtractUser(null, null, null, null, null,
                                                        $sex, null, null, $age,
$isRich, $isVegetarian, $isParent);
                $user->founded_country = $country;
                return $user;
        }
}
```

4. Αρχείο Index

Πρόκειται για το πρώτο αρχείο που ενεργεί, όταν γίνεται ένα request.

```
<?php

        require_once('utility.php');
        require_once('controller/User.php');
        require_once('controller/BaseAction.php');
        require_once('controller/Category.php');
        require_once('controller/Product.php');
        require_once('controller/Order.php');

        session_start();

        //include all model here
        require_once('model/Include.php');
        //include Exception Class
        require_once('Exception.php');

        //XSS Security, reject html special characters
        array_walk_recursive($_POST, "my_XSS_SECURE");
        array_walk_recursive($_GET, "my_XSS_SECURE");

        //Load the User object in session
        if(checkArr($_SESSION, "user"))
```

```
        $User = unserialize($_SESSION['user']);
else
        $User = null;

$page = checkArr($_GET, "page");
if($page == null)
        $page = checkArr($_POST, "page");

if(isset($_GET['method']) && $_GET['method'] == "ajax")
        require_once('controller/ajax.php');
else
        switch($page){
                case 'index':
                        require_once('view/header.php');
                        require_once('view/index.php');
                        require_once('view/footer.php');
                        break;
                case 'registration':
                        $action = new BaseAction();
                        $action->registration();
                        break;
                case 'create_user':
                        $user = new ControllerUser();
                        $user->register();
                        break;
                case 'login':
                        $action = new BaseAction();
                        $action->login();
                        break;
                case 'logout':
                        $action = new BaseAction();
                        $action->logout();
                        break;
                case 'login_data':
                        $user = new ControllerUser();
                        $user->login();
                        break;
                case 'view_users':
                        $action = new ControllerUser();
                        $action->viewUsers();
                        break;
```

```
case 'visit_profile':
        $user = new ControllerUser();
        $user->visitProfile();
        break;
case 'view_categories':
        $action = new BaseAction();
        $action->viewCategories();
        break;
case 'category':
        $action = new ControllerCategory();
        $action->viewProductList();
        break;
case 'create_category':
        $action = new BaseAction();
        $action->createCategory();
        break;
case 'add_category':
        $action = new ControllerCategory();
        $action->add_category();
        break;
case 'create_product':
        $action = new BaseAction();
        $action->createProduct();
        break;
case 'add_product':
        $action = new ControllerProduct();
        $action->add_product();
        break;
case 'product':
        $action = new BaseAction();
        $action->viewProductInfo();
        break;
case 'view_cart':
        $action = new BaseAction();
        $action->viewCart();
        break;
case 'make_order':
        $action = new ControllerOrder();
        $action->make();
        break;
case 'submit_order':
```

```
                $action = new ControllerOrder();
                $action->submit();
                break;
            case 'view_profile':
                $user = new ControllerUser();
                $user->viewProfile();
                break;
            case 'not_found':
                require_once('view/header.php');
                require_once('view/403.php');
                require_once('view/footer.php');
                break;
            case 'about':
                require_once('view/header.php');
                require_once('view/about.php');
                require_once('view/footer.php');
                break;
            case 'communicate':
                require_once('view/header.php');
                require_once('view/communicate.php');
                require_once('view/footer.php');
                break;
            case 'send_email':
                $action = new BaseAction();
                $action->sendEmail();
                break;
            case 'view_orders':
                $action = new ControllerOrder();
                $action->viewOrders();
                break;
            case 'view_order':
                $action = new ControllerOrder();
                $action->viewOrder();
                break;
            default:
                require_once('view/header.php');
                require_once('view/404.php');
                require_once('view/footer.php');
                break;
        }
?>
```