ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Υλοποίηση Διαδικτυακού Συστήματος Ανταλλαγής Προϊόντων και Υπηρεσιών The VCommunity

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

των

ΚΑΡΛΗ ΜΑΡΙΑ ΝΙΚΗ
ΚΟΥΡΤΙΔΗ ΑΝΕΣΤΗ

Επιβλέπων : Τιμολέων Σελλής
Καθηγητής Ε.Μ.Π.

Αθήνα, Δεκέμβριος 2007

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Υλοποίηση Διαδικτυακού Συστήματος Ανταλλαγή Προϊόντων και Υπηρεσιών The VCommunity

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

των

ΚΑΡΛΗ ΜΑΡΙΑ ΝΙΚΗ
ΚΟΥΡΤΙΔΗ ΑΝΕΣΤΗ

Επιβλέπων : Τιμολέων Σελλής
Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 17η Δεκεμβρίου 2007.

.....	.....	.....
Ιωάννης Βασιλείου	Τιμολέων Σελλής	Ανδρέας-Γεώργιος Σταφυλοπάτης
Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.

Αθήνα, Δεκέμβριος 2007

.....

ΜΑΡΙΑ ΝΙΚΗ ΚΑΡΛΗ – ΑΝΕΣΤΗΣ ΚΟΥΡΤΙΔΗΣ

Διπλωματούχοι Ηλεκτρολόγοι Μηχανικοί και Μηχανικοί Υπολογιστών Ε.Μ.Π.

Περίληψη

Σκοπός της διπλωματικής εργασίας είναι η ανάλυση των απαιτήσεων, η σχεδίαση και η υλοποίηση του συστήματος VCommunity, που αποτελεί δικτυακή υπηρεσία συναλλαγής αντικειμένων και υπηρεσιών. Η VCommunity είναι μία ηλεκτρονική κοινωνία της οποίας οι χρήστες μπορούν να δημιουργούν και να συμμετέχουν σε κοινότητες σύμφωνα με τα ενδιαφέροντα, τις επιδιώξεις τους ή τα συμφέροντά τους και να αλληλεπιδρούν με άλλα μέλη της κοινωνίας εκτελώντας δοσοληψίες αγαθών. Στα πλαίσια της VCommunity λειτουργεί μία μη νομισματική οικονομία. Συγκεκριμένα, οι συναλλαγές που διεκπεραιώνονται από τα μέλη της VCommunity δεν περιλαμβάνουν χρηματικές δοσοληψίες. Το χαρακτηριστικό της ηλεκτρονικής μας κοινωνίας είναι η υπόληψη των μελών της, η οποία αντικατοπτρίζει την δραστηριοποίηση και την φερεγγυότητα των μελών. Η υπόληψη των μελών προκύπτει ως αποτέλεσμα των συναλλαγών που εκτελούν τα μέλη της ηλεκτρονικής μας κοινωνίας.

Λέξεις Κλειδιά: VCommunity, ηλεκτρονική κοινωνία, κοινότητες, συναλλαγές, μη νομισματική οικονομία, υπόληψη

Abstract

The purpose of the diploma thesis is the requirements' specification, design and implementation of the VCommunity system. The VCommunity system is a community portal to support sharing of goods and services. The users of this e-society can create and participate in communities according to their interests or goals. They can interact with each other by participating in transactions but no money should be involved. The basic feature of this barter network is the reputation system that rates its entities. The reputation reflects the activity and credibility of an entity, as far as the transaction, which this entity completes, are concerned. The reputation is a result of the credits that an entity receives when the entity after the entity's participation in a transaction.

Keywords: VCommunity, e- society, communities, transactions, barter network, reputation

Πίνακας περιεχομένων

1

Εισαγωγή

1.1 Web programming and Internet technologies

Η συνεχής ανάπτυξη του Διαδικτύου καθώς και η ολοένα συχνότερη χρήση των υπηρεσιών που παρέχονται από τον παγκόσμιο ιστό οδήγησε στην ανάγκη μοντελοποίησης των ανθρώπινων δραστηριοτήτων και την απεικόνιση των σύγχρονων κοινωνιών σε ηλεκτρονικές κοινότητες (e-societies). Η αυξανόμενη σημασία του Διαδικτύου επέφερε δραστικές αλλαγές σε πολλές πτυχές της σύγχρονης ζωής και κυρίως στον τρόπο αλληλεπίδρασης των ανθρώπων. Στις ημέρες μας παρατηρούμε άνθιση των συστημάτων διαδικτύου που αποτελούν μικρογραφία των ανθρώπινων κοινοτήτων, μοντελοποιούν την κοινωνική δραστηριότητα και επιδιώκουν να αποτυπώσουν την κοινωνική συμπεριφορά.

Το αποτέλεσμα αυτής της σύγχρονής ροπής προς τα εξελιγμένα κοινωνικά site είναι η στροφή του ενδιαφέροντος προς τον δικτυακό προγραμματισμό και τις σύγχρονες τεχνολογίες διαδικτύου. Στις μέρες μας, λοιπόν, υπάρχουν πολλά εργαλεία για δικτυακό προγραμματισμό που παρέχουν πολλές δυνατότητες και ευκολίες για γρήγορη ανάπτυξη μεγάλων δικτυακών συστημάτων. Η έλευση των Web 2,0 τεχνολογιών, οι οποίες αναφέρονται στην δεύτερη γενιά επικοινωνίας διαμέσω του Διαδικτύου στόχευσε την εξυπηρέτηση της δημιουργικότητας, της συνεργασίας και της αλληλεπίδρασης των χρηστών του Διαδικτύου, ενισχύοντας την ανάπτυξη πολύπλοκων επιχειρηματικών μοντέλων και

προσδίδοντας ένα διαφορετικό χαρακτήρα στον τρόπο με τον οποίο οι προγραμματιστές και οι τελικοί χρήστες αλληλεπιδρούν με τα διαδικτυακά συστήματα.

1.2 Αντικείμενο διπλωματικής

Αντικείμενο της διπλωματικής εργασίας είναι η υλοποίηση ενός διαδικτυακού συστήματος κοινοτήτων με σκοπό την ανταλλαγή προϊόντων και υπηρεσιών. Μία διαδικτυακή κοινότητα (virtual/online community, e-community) είναι ένα σύνολο ανθρώπων, γνωστών και αγνώστων μεταξύ τους, που αλληλεπιδρούν μέσω τεχνολογίας πολυμέσων, όπως μέσω μηνυμάτων, ηλεκτρονικού ταχυδρομείου κτλ. Στις ημέρες μας άνθρωποι που γνωρίζονται στην πραγματική τους ζωή, συνηθίζουν να επιλέγουν την συμμετοχή τους σε ηλεκτρονικές κοινότητες μαζί με τους οικείους τους, αξιοποιώντας τα διαδικτυακά συστήματα ως συμπληρωματική μορφή της μεταξύ τους επικοινωνίας. Το σύστημά μας, ως e-community, υποστηρίζει την ηλεκτρονική αλληλεπίδραση των χρηστών του Διαδικτύου και αποτελείται από επιμέρους κοινότητες τις οποίες διαμορφώνουν και στις οποίες συμμετέχουν ελεύθερα οι χρήστες του Διαδικτύου. Οι κοινότητες αυτές αποκτούν υπόσταση επί ενός συνδετικού κρίκου που ορίζεται στο καταστατικό δημιουργίας τους. Οι χρήστες συμμετέχουν σε αυτές σύμφωνα με τα ενδιαφέροντα, τις επιδιώξεις τους αλλά και για διάφορους άλλους λόγους όπως γεωγραφικούς. Μία κοινότητα του συστήματός μας μπορεί να απεικονίζει και μία πραγματική ομάδα, όπως π.χ. μία ομάδα συνεργατών μίας εταιρείας.

Ο προορισμός του συστήματός μας είναι η υποστήριξη συναλλαγών επί προϊόντων και υπηρεσιών. Το σύστημά που σχεδιάσαμε και υλοποιήσαμε αποτελεί την σύγχρονη απεικόνιση του οικονομικού συστήματος του αντιπραγματισμού. Ο όρος αντιπραγματισμός περιγράφει το είδος εμπορίου που δεν χρησιμοποιεί ως μέσο συναλλαγής νομισματικές μονάδες, αλλά τα αγαθά ή οι υπηρεσίες ανταλλάσσονται με αλλά αγαθά ή / και υπηρεσίες.. Αξίζει να σημειώσουμε ότι, ενώ ο αντιπραγματισμός και το χρήμα είναι διαφορετικά μέσα εξισορρόπησης μιας οικονομικής ανταλλαγής, στις περισσότερες κοινωνίες οι μη νομισματικές αγοραπωλησίες συνυπάρχουν με νομισματικά συστήματα. Η ενσωμάτωση του συστήματός μας στα πλαίσια μίας καπιταλιστικής αγοράς τεκμηριώνεται από το γεγονός ότι οι μη νομισματικές αγοραπωλησίες στις περισσότερες κοινωνίες συνυπάρχουν με νομισματικά συστήματα.

Μια συναλλαγή είναι δυνατή, όταν η σύμπτωση των επιθυμιών των παραγόντων της επιτρέπει την ανταλλαγή μεταξύ ενός κύκλου προσφορών: κάθε μέλος πρέπει να είναι σε θέση να επιδείξει κάτι άλλο το οποίο επιθυμεί το άλλο μέλος. Η συναλλαγή λαμβάνει χώρα όταν υπάρχει αμοιβαίο συμφέρον ή επιθυμία μεταξύ δυο οντοτήτων. Ο αντιπραγματισμός

επιτρέπει την διαπλοκή ανταλλαγών μεταξύ εμπορευμάτων, δεξιοτήτων, διαπολιτισμικών γνώσεων και ικανοτήτων κάθε είδους, που είτε αφορούν τέχνη, ψυχαγωγία, ανάγκη ή πνευματική προσπάθεια. Ιστορικά, ο αντιπραγματισμός λειτούργησε σε προ-καπιταλιστικές κοινωνίες καθώς και σε κοινωνίες που δεν παρουσίαζαν ανεπτυγμένο νομισματικό σύστημα. Κίνητρο για την υποστήριξη του συστήματος του αντιπραγματισμού στις σύγχρονες κοινωνίες αποτελεί η εύστοχη διαχείριση των χρημάτων και η ανάγκη για άρση οικονομικών περιορισμών κυρίως ανάμεσα σε εύπορους και μη οικονομικούς δράστες. Επιπλέον, παρότι τα χρήματα θεωρούνται ως το πιο βολικό μέσο για συναλλαγές, το Διαδίκτυο άλλαξε αυτή την αντίληψη παρέχοντας τη δυνατότητα υποστήριξης ιστοσελίδων που μπορούν εύκολα και γρήγορα να συστηματοποιήσουν συναλλαγές. Έτσι, λοιπόν και το σύστημα που υλοποιούμε αποτελεί ένα άτυπο σύστημα αντιπραγματισμού με συμμετέχοντες χρήστες και κοινότητες του Διαδικτύου, οι οποίοι συναλλάσσονται επί αγαθών των οποίων η αξία καθορίζεται σε βάση «διαχειριστικής καταπίστευσης». Το σύστημά μας παρέχει λοιπόν έναν τρόπο εύρεσης και απόκτησης αγαθών και παρεχόμενων υπηρεσιών που δεν απαιτούν χρηματικό αντάλλαγμα, αλλά εκτελούνται στην βάση της αξιοπιστίας των συναλλασσόμενων οντοτήτων.

Οι αναλυτές των διαδικτυακών κοινοτήτων αναγνωρίζουν ότι η συμμετοχή των ανθρώπων σε ηλεκτρονικές κοινότητες δεν βασίζεται στο αίσθημα συνεισφοράς ή σε απόλυτα ανιδιοτελή συμπεριφορά. Υπογραμμίζουν τέσσερις βασικές παραμέτρους για την ενδυνάμωση των κοινωνικών δικτύων:

- Αναμενόμενη Αμοιβαιότητα

  Ένα άτομο έχει κίνητρο να συμμετάσχει πιο ενεργά σε ένα διαδικτυακό σύστημα, με την προσδοκία ότι θα λάβει ως αντάλλαγμα, προς όφελός του, μεγαλύτερη δραστηριοποίηση από τα άλλα μέλη της κοινότητας. Έχει διαπιστωθεί ότι πιο ενεργοί συμμετέχοντες ενός διαδικτυακού συστήματος ανταλλαγής πληροφοριών (π.χ. forums) τείνουν να λαμβάνουν ταχύτερα, περισσότερες αποκρίσεις έναντι νέων, άγνωστων συμμετεχόντων.

- Αύξηση Αναγνώρισης

  Η αναγνώριση είναι σημαντική για τις οντότητες μίας ηλεκτρονικής κοινότητας, όπως και τα φυσικά πρόσωπα επιδιώκουν την αναγνώριση για την οποιουδήποτε είδους συμβολή τους. Η επιδίωξη γοήτρου είναι αναγνωρισμένη ως ένα από τα βασικότερα κίνητρα για ατομική συνεισφορά σε ομάδα. Η συνεισφορά λοιπόν ενός ατόμου και μάλιστα η ευυπόληπτη δραστηριότητα ενός ατόμου μέσα σε μία ομάδα θα αυξηθεί στο βαθμό που θα είναι ορατή εντός του κοινωνικού πλαισίου στο οποίο το άτομο δραστηριοποιείται. Η δύναμη της φήμης αναδεικνύεται από το ακόλουθο παράδειγμα. Ένας συστηματικά κακόβουλος χρήστης του Διαδικτύου που εμπλέκεται

σε παράνομες δραστηριότητες θα όφειλε να προστατεύσει την προσωπική του ταυτότητα με πολλαπλά ψευδώνυμα. Η επαναλαμβανόμενη χρήση του ίδιου ψευδώνυμου μπορεί να βοηθήσει τις αρχές να εντοπίσουν αυτό το άτομο. Ωστόσο, οι κακόβουλοι χρήστες παρουσιάζονται απρόθυμοι να αλλάξουν το ψευδώνυμό τους σε τακτική βάση, καθώς η κατάσταση που συνδέεται με αυτό το ψευδώνυμό τους θα χαθεί.

- Αίσθημα Αποτελεσματικότητας

  Οι χρήστες του Διαδικτύου τείνουν να αυξήσουν την δραστηριοποίησή τους εντός ηλεκτρονικών κοινοτήτων, όταν νιώθουν ότι οι ενέργειές τους έχουν επίδραση στο περιβάλλον αυτό. Η συχνή και ποιοτική συμβολή στην κοινότητα βοηθά το άτομο να πιστεύει ότι έχει αντίκτυπο στην ομάδα και ότι υποστηρίζει την δική του εικόνα ως αποτελεσματικό μέλος της κοινότητας.

- Αίσθημα Συμμετοχής

  Ο άνθρωπος είναι αδιαμφισβήτητα φύσει κοινωνικό όν και αποτελεί κίνητρο για πολλούς ανθρώπους το γεγονός ότι η συνεισφορά τους σε μία κοινότητα λαμβάνει άμεση απόκριση.

Η δομή του συστήματός μας, το οποίο αποτελείται από κοινότητες χρηστών, και η δραστηριότητα των οντοτήτων εντός του συστήματός μας έρχονται σε συμφωνία με τα παραπάνω κίνητρα.

Ένα βασικό χαρακτηριστικό του συστήματός μας, το οποίο υπηρετεί τα παραπάνω κίνητρα, αποτελεί η μέτρηση της αξιοπιστίας των οντοτήτων του που διαμορφώνει την φήμη και κατ' επέκταση την υπόληψή τους μέσα στα πλαίσια της ηλεκτρονικής μας κοινωνίας. Η αξιοπιστία μίας οντότητας του συστήματος συστήνεται σύμφωνα με τις συναλλαγές που διεκπεραιώνει, αποτελεί κοινωνική αξιολόγηση και εκφράζει την άποψη των υπολοίπων οντοτήτων για αυτή την οντότητα. Αποτελεί σημαντική παράμετρο για την διαδικτυακή μας κοινότητα καθώς είναι ένας άκρως αποτελεσματικός μηχανισμός κοινωνικού ελέγχου. Η επιρροή εντοπίζεται σε ανταγωνιστικές ρυθμίσεις και συνεταιριστικές ρυθμίσεις. Όπως είναι φυσικό, μία οντότητα θα επιλέξει να συναλλαχθεί με μία πιο αξιόπιστη οντότητα όπως επίσης ένας χρήστης θα επιλέξει να συμμετάσχει σε μία αξιόπιστη κοινότητα. Η φήμη είναι ένας πολύ σημαντικός παράγοντας στις ηλεκτρονικές κοινότητες όπου, λόγω απουσίας εικόνας για την φυσική υπόσταση μίας οντότητας, η φήμη αποτελεί το μοναδικό κριτήριο για απόδοση εμπιστοσύνης. Μάλιστα, η οικοδόμηση και η διατήρηση μίας καλής φήμης είναι πολύ σημαντικό κίνητρο για την συμβολή των χρηστών του Διαδικτύου στην ηλεκτρονική κοινότητα που διαμορφώνουμε.

Συμπερασματικά, το αντικείμενο της διπλωματικής μας εργασίας είναι η ανάπτυξη, σχεδίαση και υλοποίηση μίας ηλεκτρονικής κοινότητας που λειτουργεί υπό το σύστημα του

αντιπραγματισμού και η δραστηριοποίηση των μελών της συντελείται υπό το πρίσμα της υπόληψής τους.

1.2.1 Συνεισφορά

Η συνεισφορά της διπλωματικής συνοψίζεται ως εξής:

1. Μελετήσαμε συστήματα ανταλλαγών(barter networks)
2. Μοντελοποιήσαμε το χαρακτηριστικό της υπόληψης.
3. Μελετήσαμε frameworks για web development
4. Σχεδιάσαμε και αναπτύξαμε πρότυπο σύστημα για την αξιολόγηση της υπόληψης
5. Αναπτύξαμε, σχεδιάσαμε και υλοποιήσαμε σύστημα ανταλλαγής προϊόντων και υπηρεσιών μέσα σε μη νομισματικό περιβάλλον

1.3 Οργάνωση κειμένου

Εργασίες σχετικές με το αντικείμενο της διπλωματικής παρουσιάζονται στο Κεφάλαιο 2 . Το Κεφάλαιο 3 αναλύει τις απαιτήσεις του δικτυακού συστήματος ανταλλαγής προϊόντων και υπηρεσιών του συστήματος που υλοποιήσαμε. Στο Κεφάλαιο 4 παρουσιάζουμε τον σχεδιασμό του συστήματος και στο Κεφάλαιο 5 αναπτύσσουμε την υλοποίηση του συστήματος. Το Κεφάλαιο 6 αναλύει τα σενάρια ελέγχου που εφαρμόσαμε προκειμένου να διαπιστώσουμε την συμφωνία της απόκρισης του συστήματός μας με την επιθυμητή συμπεριφορά. Το Κεφάλαιο 7 συνοψίζει την εργασία μας και το Κεφάλαιο 8 παρουσιάζει τις πηγές από όπου αντλήσαμε χρήσιμες πληροφορίες για την εκπόνηση της διπλωματικής μας εργασίας .

2

Σχετικές εργασίες

Στο διαδίκτυο υπάρχουν πολλά διαφορετικά συστήματα, καθένα από τα οποία έχει αναπτύξει μια ιδιότητα που το κάνει πρωτοποριακό, προσδίδοντας ένα χαρακτηριστικό γνώρισμα σε αυτό. Έτσι, υπάρχουν ιστοσελίδες που δίνουν στους χρήστες τη δυνατότητα να αλληλεπιδρούν, ανταλλάσσοντας ή πουλώντας και αγοράζοντας αγαθά. Ακόμα υπάρχουν συστήματα που ενσωματώνουν στις λειτουργίες τους χαρακτηριστικά υπόληψης για κάθε μέλος τους ή που τους επιτρέπουν να κοινωνικοποιούνται μέσω αυτού, προσφέροντας εργαλεία για να το επιτύχουν. Το σύστημα της VCommunity, όπως ονομάζεται το σύστημα που υλοποιούμε, προσπαθεί να συνδυάσει όλα τα παραπάνω στοιχεία παρέχοντας στον χρήστη μια ολοκληρωμένη λύση. Εντάσσεται δηλαδή στην κατηγορία των λεγόμενων Social Barter Networks.

2.1 Συστήματα Υπόληψης

- eBay

  Το *eBay* είναι κατεξοχήν ένας δικτυακός τόπος όπου μπορείς να δημοπρατείς αντικείμενα. Αφού παρέλθει ένας συγκεκριμένος χρόνος, που ορίζει ο χρήστης, το αντικείμενο κατοχυρώνεται σε αυτόν που έχει κάνει την υψηλότερη προσφορά. Το eBay, όμως, για να διαφυλάξει τους "καλούς" χρήστες από τους επιτήδειους και τους

κλέφτες, εφαρμόζει ένα σύστημα υπόληψης, ώστε ο καθένας να έχει μια ιδέα για αυτόν με τον οποίο συναλλάσσεται.

Αναλύοντας το σύστημα υπόληψης που διαθέτει ο συγκεκριμένος ιστοχώρος, βλέπουμε πως βασίζεται σε πόντους υπόληψης που παίρνει κάθε χρήστης μετά από κάθε συναλλαγή. Αφού ολοκληρωθεί δηλαδή η συναλλαγή, κάθε μέλος που πήρε μέρος μπορεί προαιρετικά να κρίνει τον άλλον θετικά, ουδέτερα ή αρνητικά αφήνοντας ταυτόχρονα και κάποιο σχόλιο. Για κάθε θετική συναλλαγή, προστίθεται ένας βαθμός στη βαθμολογία του χρήστη, για κάθε ουδέτερη κανένας ενώ για κάθε αρνητική αφαιρείται ένας βαθμός. Έτσι προκύπτει ο συνολικός βαθμός υπόληψης κάθε χρήστη που αντιστοιχείται και με ένα αστέρι του οποίου το χρώμα διαφέρει ανάλογα με το την συνολική βαθμολογία.

Ένα ιδιαίτερο χαρακτηριστικό του συστήματος αποτελεί το γεγονός πως κάθε χρήστης μπορεί να συνεισφέρει στην βαθμολογία του κρινόμενου μόνο κατά ένα βαθμό, είτε είναι αρνητικός είτε θετικός, ανεξαρτήτως των συναλλαγών που έχουν κάνει μεταξύ τους. Βγαίνει δηλαδή ο μέσος όρος όλων των βαθμών που έχει δώσει ο χρήστης στον κρινόμενο και αν είναι θετικός προσθέτει ένα βαθμό, αν είναι αρνητικός αφαιρεί έναν κτλ.

Διαπιστώνουμε πως το σύστημα που υλοποιεί το eBay, αποτελεί μία αναλογική βαθμολόγηση του χρήστη, λαμβάνοντας υπόψη τον κάθε χρήστη ξεχωριστά και όχι την κάθε συναλλαγή. Αυτό έχει ως συνέπεια κάθε χρήστης να μην μπορεί μεμονωμένα να επηρεάσει την βαθμολογία κάποιου άλλου χρήστη. Επίσης δίνει τη δυνατότητα για επιμέρους βαθμολογίες ή πρόσφατης βαθμολογίας.

- iKarma

  Ο ιστοχώρος *iKarma* δίνει τη δυνατότητα σε κάθε μέλος της να διατηρεί ένα προφίλ μέσω του οποίου να μπορεί να συναλλάσσεται στο διαδίκτυο. Το περιεχόμενο του προφίλ περιορίζεται σε κάποια προσωπικά και επαγγελματικά στοιχεία του χρήστη καθώς και το σημαντικότερο, την υπόληψή του. Μέσω αυτής μπορεί κάποιος να ψάξει για επαγγελματίες με υψηλή υπόληψη ή να εντοπίσει “κακούς” πελάτες, όσον αφορά στις επαγγελματικές τους υποχρεώσεις.

  Το σύστημα υπόληψης, που χρησιμοποιεί, είναι μια απλή εφαρμογή της αναλογικής βαθμολόγησης που εξετάσαμε και προηγουμένως χωρίς κάποιον ιδιαίτερο περιορισμό ως προς το πόσο επηρεάζει κάθε χρηστής ξεχωριστά της υπόληψη κάποιου άλλου. Επίσης η υπόληψη αντιστοιχίζεται απευθείας με ένα σύστημα αστεριών με κλίμακα από μηδέν έως πέντε. Ένα αρνητικό γνώρισμα της

συγκεκριμένης υπηρεσίας, είναι η απαιτούμενη έγκριση ενός χρήστη πριν από τον κρινόμενο πριν ο πρώτος μπορέσει να βαθμολογήσει.

- RapLeaf

  Μια ξεχωριστή υπηρεσία αποκλειστικά αφιερωμένη στην παροχή μιας κλίμακας βαθμολόγησης της υπόληψης ενός ατόμου, είναι το *Rapleaf*. Κάθε άτομο που διαθέτει μια ηλεκτρονική διεύθυνση, έχει αυτόματα και ένα βαθμό στην υπηρεσία αυτή, δεν απαιτείται δηλαδή να είσαι εγγεγραμμένος χρήστης. Αυτό επιτυγχάνεται λόγω του γεγονότος πως οι πηγές τις οποίες λαμβάνει υπόψη του ο συγκεκριμένος ιστοχώρος δεν είναι μόνο οι βαθμολογήσεις των χρηστών που έχουν γίνει από εκεί. Χρησιμοποιεί επιπλέον την κατάταξη που έχει κάποιος σε άλλες ιστοσελίδες social networking, ο οποίος αναγνωρίζεται από την ηλεκτρονική του διεύθυνση. Έτσι βλέπουμε πως η υπόληψη που παρέχει το *Rapleaf* είναι μια συνιστώσα των βαθμολογιών που δίνουν απευθείας οι χρήστες του αλλά και αυτών που έχει κάποιος σε άλλες ιστοσελίδες. Με αυτό τον τρόπο προσπαθεί να ενοποιήσει τον βαθμό υπόληψης που έχει κάθε χρήστης του διαδικτύου σε μία υπηρεσία.

  Το σύστημα που εφαρμόζει για να προκύψει η τελική βαθμολογία, είναι ανάλογη με αυτή του eBay, δηλαδή κάθε θετική αξιολόγηση συνεισφέρει αναλογικά όπως και κάθε αρνητική. Επίσης βλέπουμε πως κι εδώ κάθε χρήστης επηρεάζει κατά μόνο ένα βαθμό την υπόληψη κάποιου άλλου ανεξαρτήτως του αριθμού των φορών που τον έχει βαθμολογήσει. Όπως και στα προηγούμενους δικτυακούς χώρους, υπάρχουν δύο διαφορετικές κατηγορίες υπόληψης, η γενική και η Commerce Score που δείχνει την υπόληψη του χρήστη που έχει με βάση τις εμπορικές συναλλαγές που έχει κάνει.

2.2 Κοινωνικοποίηση

- MySpace

  Ο πιο γνωστός ιστοχώρος κοινωνικοποίησης μέσω του διαδικτύου είναι το *MySpace*. Μέσω αυτού μπορεί κάθε άτομο να αποκτήσει δωρεάν ένα δικό του χώρο έκφρασης και επικοινωνίας με άλλα άτομα. Καταρχάς κάθε χρήστης μπορεί να εισάγει τα προσωπικά του στοιχεία όπως την ηλικία του, που μένει κτλ. Το σημαντικότερο όμως είναι ότι επιτρέπει την δημιουργία περιεχομένου ποικίλου είδος όπως μουσική και βίντεο που μπορεί να ανεβάσει κάποιος ώστε να ακούει και να βλέπει όποιος θέλει, τις επιλογές του σε ταινίες που του άρεσαν, φωτογραφίες δικές του ή άλλες ή ακόμα να αναφέρει τα ενδιαφέροντά του, τα χόμπι του, τις ασχολίες του. Είναι βασικά ένας

χώρος όπου μπορείς να εκφράσεις την προσωπικότητα σου με διάφορους τρόπους και να την κάνεις γνωστή σε άλλους.

Έτσι σου δίνεται η δυνατότητα μέσω μια απλής αναζήτησης να βρεις άτομα με κοινά ενδιαφέροντα και να τους γνωρίσεις. Ακόμα να μοιραστείς με τους φίλους σου απόψεις και πλευρές της προσωπικότητάς σου. Ένα βοηθητικό εργαλείο σου επιτρέπει να ορίσεις μια λίστα με φίλους τους οποίους μπορείς να βλέπεις να είναι την ίδια στιγμή στο διαδίκτυο. Επίσης παρέχει και φόρουμ στο οποίο μπορείς να συζητάς για διάφορα θέματα ου σε απασχολούν με άλλα άτομα που έχουν και εκείνα παρόμοια θέματα.

Ένα ακόμα εργαλείο για περαιτέρω ομαδοποίηση και κοινωνικοποίηση είναι και η δυνατότητα να φτιάχνεις ομάδες με κάποιο κοινό ενδιαφέρον. Έτσι μπορούν να ορίζονται μέλη σε κάθε μια από αυτές καθώς και να έχουν το δικό τους χώρο. Τα δικαιώματα κάθε μέλος περιορίζονται στο τι μπορεί να βλέπει και να τροποποιεί και τα οποία τα ορίζει ο αρχηγός κάθε ομάδας.

- FaceBook

Λίγο διαφορετικό από τον προηγούμενο ιστοχώρο είναι το ανερχόμενο σε δημοτικότητα *FaceBook*, το οποίο ξεκίνησε ως ένας χώρος για δημοσίευση προσωπικών φωτογραφιών και φίλων. Στη συνέχεια όμως εξελίχθηκε σε έναν πολυχώρο ο οποίος ως κεντρικό του σημείο έχει την δικτύωση των μελών του με βάση τις σχέσεις του στην πραγματική τους ζωή. Για το σκοπό αυτό παρέχει πολλά εργαλεία με απώτερο σκοπό να προσελκύει όλο και περισσότερο αριθμό χρηστών.

Όπως είπαμε η κύρια λειτουργία που μπορεί να κάνει κάποιος χρήστης είναι να προσθέσει φωτογραφίες δικές του και φίλων του στις οποίες να ορίσει επακριβώς ποιος απεικονίζεται. Έτσι το αρχικό βήμα για τη δικτύωση επέρχεται με την εγγραφή των φίλων του χρήστη που θέλουν δουν τις φωτογραφίες αυτού καθώς και τις δικές τους. Με την εγγραφή τους όμως και τον ορισμό της σχέσης τους με τον αρχικό χρήστη προχωράνε στο δεύτερο βήμα δικτύωσης αφού πρέπει να ορίσουν από πού ξέρουν τον χρήστη αυτό. Έτσι δημιουργείται ένα πλέγμα με τις προσωπικές σχέσεις κάθε χρήστη το οποίο λειτουργεί ως εργαλείο για την περαιτέρω επέκταση του ιστοχώρου. Μπορούμε εύκολα να καταλάβουμε πως οι χρήση των φωτογραφιών για την δικτύωση με τους γνωστούς κάνει ποιο άμεση την επικοινωνία μεταξύ τους.

Ένα ακόμη σημαντικό στοιχείο αυτού του δικτυακού χώρου είναι η δυνατότητα που δίνει στους χρήστες να δημιουργήσουν και να ενσωματώσουν εφαρμογές οι οποίες χρησιμοποιούν τα στοιχεία καθενός με σκοπό την περαιτέρω επικοινωνία μεταξύ

τους καθώς και την εκμαίευση πληροφοριών για τους άλλους. Μπορεί ο καθένας δηλαδή να «ζητήσει» από κάποιον φίλο του να χρησιμοποιήσει μια εφαρμογή ώστε να μάθει περισσότερα για αυτόν.

Τέλος ο συγκεκριμένος ιστοχώρος παρέχει διαφόρους τρόπους επικοινωνίας με τους υπόλοιπους χρήστες όπως η συζήτηση μέσα από δημοσίευσης μηνυμάτων, τα μηνύματα και τους «χαιρετισμούς» που μπορείς να στείλεις σε κάποιον. Αυτό το χαρακτηριστικό μαζί με την δυνατότητα δημιουργίας ομάδων με μέλη για κάποιον κοινό σκοπό όπως το καταπολέμηση του φαινομένου του θερμοκηπίου, ενδυναμώνουν την αναγκαιότητα κάθε χρήστη να συμμετέχει ενεργά στο *FaceBook*, με αποτέλεσμα να το καθίστα πολύ δημοφιλές.

2.3 Μη νομισματικές συναλλαγές

- Swapthing

  Η υπηρεσία *Swapthing* προσφέρει στους χρήστες της την δυνατότητα να ανταλλάσσουν, πωλούν και αγοράζουν αγαθά. Κάθε χρήστης μπορεί να εισάγει στο σύστημα ένα αγαθό που κατέχει ή που επιθυμεί και να δηλώσει με τι θέλει να το ανταλλάξει ή το χρηματικό του αντίτιμο. Όταν βρεθεί ο κατάλληλος χρήστης η συναλλαγή ολοκληρώνεται με την καταβολή ενός δολαρίου στην υπηρεσία. Το σύστημα δεν παρεμβαίνει καθόλου στη διαδικασία της υλικής συναλλαγής πέρα από την παροχή των απαιτούμενων στοιχείων κάθε συναλλασσόμενου.

  Πέρα από το βασικό αυτό κορμό των υπηρεσιών, η *Swapthing* παρέχει σε κάθε χρηστή τη δυνατότητα να εγκρίνει ή να απορρίψει μια προσφορά ανάλογα αν τον εξυπηρετεί, με αποτέλεσμα να προκρίνεται η πιο συμφέρουσα για εκείνον. Η επιλογή γίνεται μέσω της λίστας με την τρέχουσες προσφορές του κάθε χρήστη που μπορεί να προσπελάσει από το προφίλ του. Στο προφίλ επίσης περικλείονται προσωπικές πληροφορίες του κάθε χρήστη με σημαντικότερο έναν βαθμό με τον οποίο ορίζεται η υπόληψη του χρήστη και ο οποίος δημιουργείται με βάση την βαθμολογία που παίρνει από κάθε συναλλαγή.

  Επίσης κάθε αγαθό πλαισιώνεται από διάφορες πληροφορίες όπως και από φωτογραφίες του αντικειμένου. Για την κατηγοριοποίηση των αγαθών ακολουθείται το σύστημα των ετικετών όπου σε κάθε αντικείμενο ο χρήστης μπορεί να ορίσει πολλές διαφορετικές ετικέτες με κριτήριο να προσδιορίζουν το αντικείμενο αυτό.

  Τέλος στο *Swapthing* μπορεί ο χρήστης να ορίσει έναν δικό του κύκλο από μέλη με σκοπό να διευκολύνεται η ανταλλαγή μεταξύ αυτών των μελών. Κάθε μέλος μπορεί

να βλέπει τα αγαθά που συναλλάσσονται από άτομα μέσα στον κύκλο καθώς και να συμμετέχει σε ένα ξεχωριστό φόρουμ για τον κύκλο αυτό. Ο κύκλος δεν υφίσταται ως οντότητα που συνεπάγεται την μη κατοχή αγαθών.

3

Ανάλυση Απαιτήσεων Συστήματος

Το σύστημα VCommunity, το οποίο υλοποιούμε, αποτελεί μια δικτυακή υπηρεσία παγκοσμίου ιστού για την ανταλλαγή προϊόντων και υπηρεσιών. Οι χρήστες αυτής της δικτυακής υπηρεσίας μπορούν να δομούν και να εντάσσονται σε κοινότητες ανάλογα με τα ενδιαφέροντά τους ή τις επιδιώξεις τους. Κίνητρα για δημιουργία κοινοτήτων ή ένταξη σε αυτές μπορεί να εντοπιστεί και σε διάφορους άλλους λόγους, όπως π.χ. γεωγραφικούς. Οι ανταλλαγές προϊόντων και υπηρεσιών ανάμεσα σε χρήστες ή κοινότητες λαμβάνουν χώρα στα πλαίσια ενός μη νομισματικού περιβάλλοντος, υπό την έννοια ότι οι συναλλαγές που εκτελούνται ανάμεσα στους χρήστες ή τις κοινότητες δεν περιλαμβάνουν χρηματικές δοσοληψίες. Ωστόσο ένα σημαντικό στοιχείο του συστήματος μας είναι η υπόληψη των χρηστών και των κοινοτήτων. Συγκεκριμένα, στους χρήστες/κοινότητες προσδίδεται ένα χαρακτηριστικό φήμης που αφορά στις συναλλαγές που εκτελούν. Όταν ένας χρήστης διεκπεραιώσει μια δοσοληψία (π.χ. ανταλλάξει κάποιο αντικείμενο με έναν άλλο χρήστη) λαμβάνει κάποιο credit για αυτή του την πράξη, το οποίο τον χαρακτηρίζει ως προς την αξιοπιστία του καθώς και την δραστηριοποίησή του ως μέλος του συστήματος.

3.1 Αρχιτεκτονική

Η VCommunity είναι ένα καλώς δομημένο περιβάλλον που απεικονίζει μέσα από την δική του αφαιρετική προοπτική μία πραγματική κοινωνία. Η αρχιτεκτονική της κοινωνίας αυτής έχει τριπλή υπόσταση. Τα στοιχεία της αρχιτεκτονικής της VCommunity είναι:

- Η δομή της κοινωνίας (οντότητες : κοινότητες – χρήστες, αγαθά : προϊόντα-υπηρεσίες)
- Η δραστηριότητά της (ανταλλαγή προϊόντων και υπηρεσιών)
- Το χαρακτηριστικό της (μη-νομισματική δραστηριότητα , υπόληψη)

Ο βασικός χρήστης του συστήματος μας είναι ο μέσος χρήστης του Διαδικτύου. Οι χρήστες μπορούν να δομούν και να εντάσσονται σε κοινότητες (communities). Οι κοινότητες αυτές έχουν ένα συνδετικό κρίκο, τον οποίο ενστερνίζονται τα μέλη τους. Αυτός ο συνδετικός κρίκος τεκμηριώνει την σύσταση της κοινότητας και είτε εκφράζει ενδιαφέροντα και επιδιώξεις των μελών της κοινότητας είτε επιδιώκει να συνδέσει τους χρήστες του συστήματος σε ομάδες για άλλους λόγους όπως γεωγραφικούς. Το σύστημά μας λοιπόν έχει δύο βασικές οντότητες - δράστες, τους χρήστες του Διαδικτύου και τις κοινότητες που αυτοί συστήνουν.

Μία ακόμα παράμετρος της δομής της VCommunity είναι τα αγαθά (commodities). Τα αγαθά αυτά είναι είτε προϊόντα είτε υπηρεσίες. Η ιδιοκτησία ενός αγαθού ανήκει σε μία και μοναδική οντότητα (χρήστης/κοινότητα) και το δικαίωμα χρήσης του αγαθού καθορίζεται από αυτή την οντότητα. Αρχικά, το δικαίωμα χρήσης του αγαθού έχει μόνο ο ιδιοκτήτης του.

Η δραστηριότητα των οντοτήτων εντός της VCommunity έγκειται στις συναλλαγές επί αγαθών. Υπάρχουν τριών ειδών συναλλαγές :

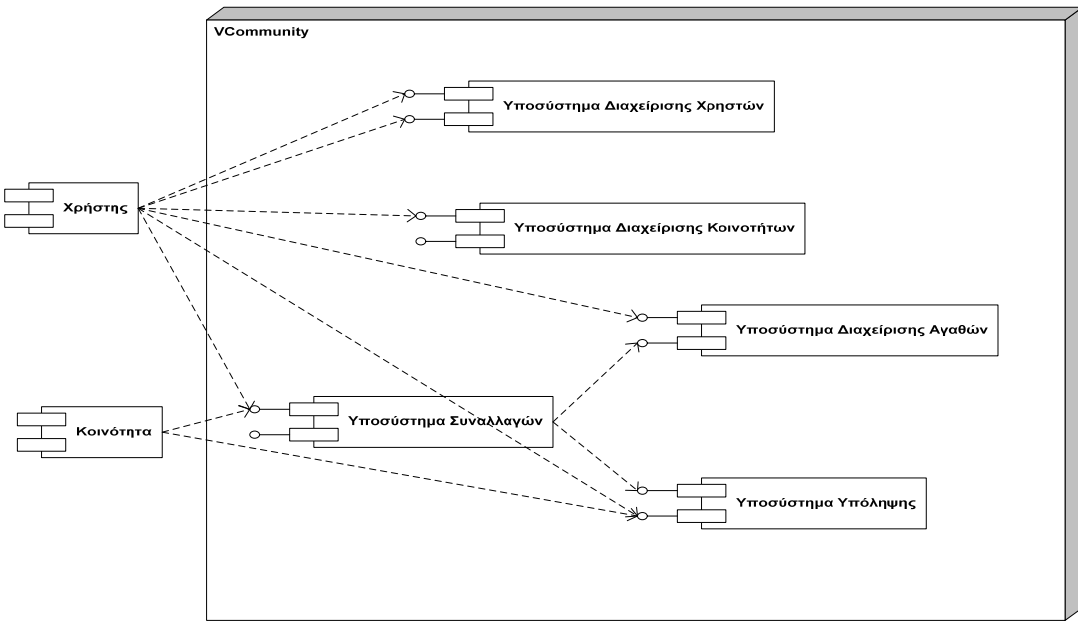
- Ανταλλαγή : Μία οντότητα προσφέρει αγαθό που κατέχει σε άλλη οντότητα, η οποία με τη μεριά της προσφέρει ως αντάλλαγμα ένα δικό της αγαθό.
- Δωρεά : Μία οντότητα προσφέρει ένα αγαθό που κατέχει σε άλλη οντότητα
- Κοινοχρησία : Μία οντότητα επιτρέπει σε μία άλλη οντότητα να χρησιμοποιεί αγαθό που ανήκει στην πρώτη.

Το βασικό χαρακτηριστικό αυτών των συναλλαγών είναι ότι έχουν μη νομισματικό χαρακτήρα. Αυτό σημαίνει ότι δεν περιλαμβάνουν χρηματικές δοσοληψίες μεταξύ των οντοτήτων ή μεταξύ των οντοτήτων και του συστήματός μας. Η οικονομία λοιπόν της ηλεκτρονικής μας κοινωνίας είναι μη νομισματική και δεν έχει τον καπιταλιστικό χαρακτήρα των σύγχρονων κοινωνιών.

Ένα βασικό χαρακτηριστικό της ηλεκτρονικής μας κοινωνίας είναι η υπόληψη των οντοτήτων της. Η υπόληψη αντικατοπτρίζει πόσο «καλό» μέλος της κοινωνίας είναι αυτή η

οντότητα. Δομείται με βάση δύο συνιστώσες , την αξιοπιστία της οντότητας όσον αφορά τις συναλλαγές στις οποίες έχει συμμετάσχει καθώς και τη δραστηριοποίηση της οντότητας.

Το παρακάτω διάγραμμα μας δίνει μία εικόνα για τη δομή του συστήματός μας:



3.2 Περιγραφή Λειτουργιών VCommunity

3.2.1 Δομή VCommunity

Ο μικρόκοσμος της VCommunity συνίσταται από δύο βασικές οντότητες. Οι οντότητες αυτές είναι οι χρήστες, γνωστοί ως users και οι κοινότητες. Ένας χρήστης είναι ο μέσος χρήστης του Διαδικτύου και οι κοινότητες είναι ομάδες χρηστών που συγκροτούνται με βάση ένα θέμα – το συνδετικό κρίκο των μελών της.

3.2.1.1 Πρώτη Εγγραφή Χρήστη στο Σύστημα

Ο χρήστης εισέρχεται στην ιστοσελίδα και επιλέγει την ένδειξη sign up για να εγγραφεί στο σύστημα. Στο σημείο αυτό ορίζει τα βασικά στοιχεία του προφίλ του , όπως όνομα, επίθετο, ηλικία και εργασιακή κατάσταση που συγκροτούν μία πρώτη εικόνα του χρήστη. Επιπλέον, καλείται να εισάγει και στοιχεία επικοινωνίας, όπως την ηλεκτρονική του διεύθυνση (e-mail),

το τηλέφωνό του και την διεύθυνση κατοικίας του. Τα στοιχεία αυτά θα καταστήσουν εφικτή την πραγματική αλληλεπίδραση των χρηστών που όπως θα δούμε παρακάτω ορμώμενοι από το σύστημά μας καταλήγουν σε πραγματική – φυσική επικοινωνία. Ένα ακόμα στοιχείο που συνοδεύει τον χρήστη είναι η ημερομηνία εγγραφής του στο σύστημά μας. Με βάση όλα αυτά τα στοιχεία, διαμορφώνεται μία πρώτη εικόνα του προφίλ του χρήστη και συγκροτείται ο λογαριασμός του. Αφότου δημιουργήσει λογαριασμό, ο χρήστης έχει την δυνατότητα να αλληλεπιδρά με το σύστημά μας

3.2.1.2 Ορισμός Κοινοτήτων

Αφότου ο χρήστης αποκτήσει λογαριασμό, έχει δυνατότητα να δημιουργήσει μία νέα κοινότητας. Ο χρήστης θα ορίσει το όνομα της κοινότητας, το οποίο θα είναι μοναδικό, το θέμα ενδιαφέροντος της κοινότητας, που αποτελεί το συνδετικό κρίκο των χρηστών της και τεκμηριώνει τη συγκρότηση της κοινότητας, και μία περιγραφή της κοινότητας. Αυτό είναι το καταστατικό της κοινότητας.

3.2.1.3 Συμμετοχή σε Κοινότητες

Ο χρήστης αφού εγγραφεί στο σύστημά μας έχει δυνατότητα να συμμετέχει σε υπάρχουσες κοινότητες. Ένας χρήστης μπορεί να δει τις κοινότητες της VCommunity και το προφίλ τους, δηλαδή το θέμα ενδιαφέροντός τους και την περιγραφή τους. Επιπλέον, έχει τη δυνατότητα να δει τα μέλη μίας κοινότητας και τον βαθμό της υπόληψής της. Τα κριτήρια αυτά σε συνδυασμό με τα ενδιαφέροντα, τις επιδιώξεις και τα προσωπικά στοιχεία του ένας χρήστης μπορεί ελεύθερα, με δική του βούληση, να αποφασίσει να συμμετάσχει σε μία από τις υπάρχουσες κοινότητες.

3.2.1.4 Τύποι Χρηστών

Κάθε χρήστης που ανήκει στο σύστημά μας μπορεί να συμμετάσχει σε κοινότητες. Δηλώνοντας συμμετοχή σε μία κοινότητα, ένας χρήστης χαρακτηρίζεται ως μέλος της. Ωστόσο, σημειώνουμε ότι ο χρήστης που είναι δημιουργός μίας κοινότητας, αμέσως λαμβάνει συμμετοχή στην κοινότητα με δικαιώματα μέλους – administrator. Ο ρόλος αυτός είναι κεντρικός, μοναδικός ανά κοινότητα και ουσιώδους σημασίας για την διαχείριση των θεμάτων της κοινότητας, για την δραστηριοποίησή της και για τις στρατηγικές τις επιλογές που θα αναλυθούν παρακάτω.

Στο σύστημά μας δεν υπάρχει περιορισμός στο πλήθος των κοινοτήτων στις οποίες ανήκει ένας χρήστης. Σε κάθε κοινότητα ο χρήστης έχει μοναδικό ρόλο, είτε ως administrator είτε ως απλό μέλος αλλά ο ρόλος ενός χρήστη σε μία κοινότητα δεν σχετίζεται με τον ρόλο του

χρήστη σε μία άλλη κοινότητα. Αυτό σημαίνει ότι ένας χρήστης μπορεί να είναι administrator σε μία κοινότητα και απλό μέλος σε μία άλλη.

Ένας χρήστης του συστήματός μας δεν υποχρεούται να συμμετάσχει και να δραστηριοποιείται σε μία κοινότητα. Ένας χρήστης, λοιπόν, που δεν συμμετέχει σε καμία από τις κοινότητες του συστήματός μας χαρακτηρίζεται ως «ανεξάρτητος» χρήστης.

3.2.1.5 Αγαθά – Κατηγορίες Αγαθών

Μια από τις βασικές συνιστώσες του συστήματος αποτελούν τα αγαθά(commodities) του συστήματος. Με τον όρο αγαθά εννοούμε είτε προϊόντα, που έχουν υλική υπόσταση, είτε υπηρεσίες. Κάθε αγαθό έχει έναν ιδιοκτήτη. Ιδιοκτήτης αγαθού είναι ο χρήστης που το εισήγαγε στο σύστημά μας. Ο ιδιοκτήτης του αγαθού είναι υπεύθυνος για την διαχείριση του.

Στο σύστημά μας υπάρχει και μία καθολική ιεραρχία κατηγοριών αγαθών. Οι κατηγορίες αυτές υποδεικνύουν το γενικό χαρακτήρα – φύση του αγαθού, π.χ. γραφική ύλη. Οι κατηγορίες ακολουθούν μία διεπίπεδη ιεραρχική δομή. Μια κατηγορία πρώτου επιπέδου έχει 0 ή περισσότερες υποκατηγορίες ενώ μία κατηγορία δευτέρου επιπέδου δεν έχει άλλες υποκατηγορίες και εντάσσεται σε μία κατηγορία πρώτου επιπέδου. Οι κατηγορίες του συστήματός μας είναι προκαθορισμένες. Η ένταξη ενός αγαθού σε μια κατηγορία καθώς και ο προσδιορισμός της εκάστοτε υποκατηγορίάς στην οποία ανήκει το αγαθό είναι ευθύνη του ιδιοκτήτη του αγαθού.

Τα αγαθά του συστήματός μας διαχωρίζονται με βάση τον τρόπο διάθεσής τους. Έχουμε λοιπόν τα αγαθά που ζητούνται(demanded commodities) και τα αγαθά που προσφέρονται(offered commodities). Ένας επιπλέον διαχωρισμός που γίνεται σε αυτές τις δύο ομαδοποιήσεις υποδεικνύει τον επιθυμητό τρόπο συναλλαγής επί του αγαθού. Στο σύστημά μας οι τρόποι συναλλαγής είναι τρεις:

- Ανταλλαγή (Exchange)
- Δωρεά (Gift)
- Κοινοχρησία (Sharing)

Ο ιδιοκτήτης λοιπόν του αγαθού καλείται να περιγράψει το αγαθό που εισάγει στο σύστημα καθορίζοντας το όνομά του αγαθού, την περιγραφή του, την κατηγορία και ενδεχομένως υποκατηγορία στην οποία ανήκει το αγαθό, τον τρόπο διάθεσής του (demanded/ offered αγαθό) και τον επιθυμητό τρόπο συναλλαγής επί του αγαθού.

3.2.2 Δραστηριότητες Οντοτήτων– Λειτουργίες Συστήματος

Η VCommunity είναι ένα σύστημα προσανατολισμένο στις συναλλαγές προϊόντων και υπηρεσιών υπό το πρίσμα ενός μη νομισματικού περιβάλλοντος. Τα αγαθά δεν έχουν χρηματική αξία, οι δοσοληψίες δεν έχουν χρηματικό χαρακτήρα και το σύστημά μας δεν λαμβάνει προμήθεια ως ενδιάμεσος των συναλλαγών. Οι επιτρεπτές συναλλαγές είναι τριών ειδών :

1. Ανταλλαγή
2. Δωρεά
3. Κοινοχρησία

3.2.2.1 Ανταλλαγή Αγαθών

Η ανταλλαγή αγαθών είναι μία λειτουργία κατά την οποία αλληλεπιδρούν δυο οντότητες. Κάθε μία από αυτές τις οντότητες είναι ιδιοκτήτης ενός αγαθού. Η μία οντότητα προσφέρει στην άλλη οντότητα αγαθό που κατέχει και η δεύτερη οντότητα ως αντάλλαγμα προσφέρει αγαθό της ιδιοκτησίας της. Αφού εκτελεστεί η λειτουργία της ανταλλαγής το ιδιοκτησιακό καθεστώς και τα δικαιώματα χρήσης των αγαθών, που εμπλέκονται στην ανταλλαγή, έχουν αλλάξει.

Τα σενάρια ανταλλαγής αγαθών στο σύστημά μας είναι τα παρακάτω

- Ανταλλαγή Αγαθών ανάμεσα σε ανεξάρτητους χρήστες
- Ανταλλαγή Αγαθών ανάμεσα σε ανεξάρτητο χρήστη και σε χρήστη-μέλος κοινότητας
- Ανταλλαγή Αγαθών ανάμεσα σε ανεξάρτητο χρήστη και σε κοινότητα
- Ανταλλαγή Αγαθών ανάμεσα σε χρήστη που το σύνολο κοινοτήτων στις οποίες συμμετέχει είναι ξένο του συνόλου κοινοτήτων στις οποίες ανήκει ο άλλος χρήστης – μέλος συναλλαγής
- Ανταλλαγή Αγαθών ανάμεσα σε χρήστη που το σύνολο κοινοτήτων στις οποίες ανήκει δεν είναι ξένο του συνόλου κοινοτήτων στις οποίες ανήκει ο άλλος χρήστης – μέλος συναλλαγής
- Ανταλλαγή Αγαθών ανάμεσα σε χρήστη που ανήκει σε κοινότητα C και την κοινότητα C
- Ανταλλαγή Αγαθών ανάμεσα σε χρήστη που ανήκει σε κοινότητα C και την κοινότητα K ,όπου $C \neq K$
- Ανταλλαγή Αγαθών ανάμεσα σε κοινότητα C και κοινότητα K , όπου $C \neq K$

3.2.2.2 Δωρεά Αγαθών

Η λειτουργία της δωρεάς περιλαμβάνει δυο οντότητες που αλληλεπιδρούν και ένα αγαθό που ανήκει στη μια από αυτές τις δύο οντότητες. Η οντότητα Α, που επιθυμεί να κάνει δωρεά στην οντότητα Β ένα αγαθό που κατέχει, αποποιείται της κατοχής αυτού του αγαθού. Μετά την διεκπεραίωση αυτής της συναλλαγής, το αγαθό αλλάζει ιδιοκτησία και δικαιώματα χρήσης. Πλέον ο νέος ιδιοκτήτης είναι η οντότητα Β και αυτή έχει το δικαίωμα ιδιοκτησίας του αγαθού. Η λειτουργία της δωρεάς δεν είναι διμερής λειτουργία, με την έννοια ότι η οντότητα Α που κάνει την δωρεά δεν αξιώνει κανένα αντάλλαγμα από μέρους της οντότητα Β.

Τα σενάρια δωρεάς του συστήματός μας είναι τα παρακάτω :

- Δωρεά Αγαθού από ανεξάρτητο χρήστη σε ανεξάρτητο χρήστη
- Δωρεά Αγαθού από ανεξάρτητο χρήστη σε χρήστη – μέλος κοινότητας
- Δωρεά Αγαθού από χρήστη μέλος κοινότητας σε ανεξάρτητο χρήστη
- Δωρεά Αγαθού από χρήστη που το σύνολο κοινοτήτων στις οποίες συμμετέχει είναι ξένο του συνόλου κοινοτήτων στις οποίες ανήκει ο άλλος χρήστης – μέλος συναλλαγής
- Δωρεά Αγαθού από χρήστη που το σύνολο κοινοτήτων στις οποίες ανήκει δεν είναι ξένο του συνόλου κοινοτήτων στις οποίες ανήκει ο άλλος χρήστης – μέλος συναλλαγής
- Δωρεά Αγαθού από χρήστη που ανήκει σε κοινότητα C στην κοινότητα C
- Δωρεά Αγαθού από κοινότητα C σε χρήστη που ανήκει στην κοινότητα C
- Δωρεά Αγαθού από χρήστη που ανήκει σε κοινότητα C στην κοινότητα K ,όπου C $\neq$ K
- Δωρεά Αγαθού από την κοινότητα K στον χρήστη που ανήκει σε κοινότητα C, όπου C $\neq$ K
- Δωρεά Αγαθού από κοινότητα C σε κοινότητα K , όπου C $\neq$ K

3.2.2.3 Κοινοχρησία Αγαθού

Η λειτουργία της κοινοχρησίας περιλαμβάνει δύο οντότητες που αλληλεπιδρούν και ένα αγαθό. Η λειτουργία αυτή δεν συνοδεύεται από αλλαγή ιδιοκτησιακού καθεστώτος και ο ιδιοκτήτης δεν χάνει το δικαίωμα χρήσης του αγαθού. Τα μέλη της συναλλαγής δεν ζητούν ως ανταλλάγματα άλλο προϊόν για να εκτελέσουν την συναλλαγή.

Το μοναδικό σενάριο κοινοχρησίας που επιτρέπεται από το σύστημά μας είναι :

- Κοινοχρησία αγαθού από χρήστη που ανήκει σε κοινότητα C προς την ίδια κοινότητα C

Σε αυτό το σενάριο, ο χρήστης κατέχει ένα αγαθό και θέτει αίτημα κοινοχρησίας προς την κοινότητα C. Αν η κοινότητα C αποδεχθεί την συναλλαγή, ο χρήστης εξακολουθεί να έχει την ιδιοκτησία του αγαθού αλλά τα δικαιώματα χρήσης του αγαθού διευρύνονται ώστε να περιλαμβάνουν και τα υπόλοιπα μέλη της κοινότητας C. Είναι αξιοσημείωτο ότι η κοινότητες του συστήματός μας δεν αποτελούν αποθήκες αγαθών, αποκτούν όμως αγαθά μέσω της λειτουργίας κοινοχρησίας.

3.2.3 Υπόληψη Οντοτήτων

Ο βαθμός της υπόληψης είναι ένα χαρακτηριστικό φήμης ,το οποίο προσδίδεται από το σύστημά μας σε κάθε οντότητα (χρήστη ή κοινότητα), η οποία συμμετέχει στο σύστημά μας. Ο βαθμός αυτός εκφράζει την αξιοπιστία της οντότητας, όσον αφορά στη διεκπεραίωση συναλλαγών καθώς και τη δραστηριοποίηση της οντότητας εντός της VCommunity, δηλαδή το πόσο συχνά συμμετέχει σε δοσοληψίες. Όσο πιο αξιόπιστη και όσο πιο δραστήρια είναι μία οντότητα τόσο μεγαλύτερο βαθμό υπόληψης έχει.

Η δομή της κοινωνίας του συστήματός μας υποδεικνύει την αλληλεπίδραση των βαθμών υπόληψης των οντοτήτων μας. Ο βαθμός υπόληψης ενός χρήστης είναι θεμιτό να επηρεάζεται από τους βαθμούς υπόληψης των κοινοτήτων στις οποίες συμμετέχει και αντίστοιχα ο βαθμός υπόληψης μίας κοινότητας επηρεάζει τους βαθμούς υπόληψης των μελών της. Μια οντότητα λοιπόν μπορεί να αξιολογηθεί με βάση το βαθμό που έχει λάβει εξαιτίας της ίδιας της δραστηριότητάς της και με βάση το βαθμό που της προσδίδει το γεγονός ότι ανήκει σε κοινότητες. Για παράδειγμα, η εικόνα ενός ιδιαιτέρως αξιόπιστου χρήστη μπορεί να αμαυρωθεί από την κακή εικόνα κοινοτήτων στις οποίες ανήκει. Αντίστοιχα, μία κοινότητα που έχει λειτουργήσει με ιδιαίτερη αξιοπιστία κατά τις συναλλαγές της, μπορεί να παρουσιάζει κακή εικόνα σε περίπτωση που οι χρήστες τις έχουν αναξιόπιστο προφίλ. Βέβαια, υπάρχουν και οι «ανεξάρτητοι» χρήστες, των οποίων ο βαθμός υπόληψης προκύπτει αμιγώς εξαιτίας των προσωπικών τους συναλλαγών και δεν επηρεάζεται από βαθμούς υπόληψης άλλων οντοτήτων

Ο βαθμός της υπόληψης μίας οντότητας προκύπτει από την βαθμολογία που κερδίζει η οντότητα, αφότου εκτελέσει μία συναλλαγή. Συγκεκριμένα, κατά την εκτέλεση των επιτρεπτών συναλλαγών του συστήματός μας έχουμε τις εξής βαθμολογήσεις :

- Ανταλλαγή : Η οντότητα Α δίνει έναν βαθμό υπόληψης στην οντότητα Β και η

οντότητα Β δίνει έναν βαθμό υπόληψης στην οντότητα Α.

- Κοινοχρησία : Η οντότητα Α που προσφέρει το αγαθό που κατέχει προς κοινοχρησία με την οντότητα Β λαμβάνει βαθμό υπόληψης για την συγκεκριμένη συναλλαγή από την οντότητα Β. Η οντότητα Α δεν δίνει βαθμό στην οντότητα Β για αυτή τη συναλλαγή.
- Δωρεά : Η οντότητα Α που δωρίζει αγαθό που κατέχει στην οντότητα Β λαμβάνει βαθμό υπόληψης από την οντότητα Β. Η οντότητα Β δεν λαμβάνει βαθμό για αυτή τη συναλλαγή στην οποία συμμετέχει..

Σημειώνουμε ότι κάθε οντότητα έχει ένα βαθμό υπόληψης για κάθε κατηγορία αγαθών. Οι επιμέρους αυτοί βαθμοί προκύπτουν λόγω των βαθμολογήσεων που λαμβάνει μία οντότητα με τους παραπάνω τρόπους. Οι βαθμοί της υπόληψης ανά κατηγορία συνδυαστικά διαμορφώνουν τον συνολικό βαθμό υπόληψης της οντότητας.

3.2.3.1 Μοντελοποίηση Προβλήματος Υπόληψης Οντότητας

Λαμβάνοντας υπόψη τις πράξεις συναλλαγής μεταξύ των οντοτήτων και τις αλληλεπιδράσεις των βαθμών υπόληψης προκύπτει ότι :

- Βαθμός Υπόληψης Χρήστη :

  Ο βαθμός υπόληψης χρήστη ανά κατηγορία επηρεάζεται από δύο παραμέτρους:

  α. τους βαθμούς που λαμβάνει ο χρήστης λόγω συναλλαγών στις οποίες μετέχει ο ίδιος

  β. τους βαθμούς των κοινοτήτων στις οποίες συμμετέχει

- Βαθμός Υπόληψης Κοινότητας :

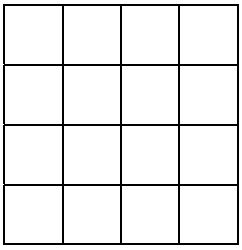
  Ο βαθμός υπόληψης κοινότητας ανά κατηγορία επηρεάζεται από δύο παραμέτρους:

  α. τους βαθμούς που λαμβάνει η κοινότητα λόγω συναλλαγών στις οποίες μετέχει η ίδια

  β. τους βαθμούς των μελών της κοινότητας.

Όπως προκύπτει από τα παραπάνω το χαρακτηριστικό της υπόληψης αποτελεί ένα από τα βασικά υποσυστήματα της VCommunity. Θα αναλύσουμε τις απαιτήσεις του υποσυστήματος υπόληψης των οντοτήτων με τη βοήθεια πινάκων.

Πίνακας Υπόληψης Χρηστών:	***Πίνακας Υπόληψης Κοινοτήτων:***
k categories	k categories

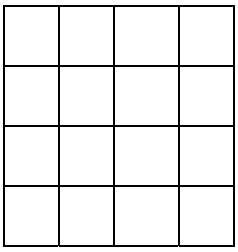
n users  v communities

Έστω ότι στο σύστημά μας υπάρχουν y χρήστες. Θεωρούμε ότι n από αυτούς τους χρήστες (όπου $n \leq y$) είναι «ενεργοί», δηλαδή n χρήστες έχουν συμμετάσχει έστω και μία φορά σε μία συναλλαγή. Επίσης θεωρούμε ότι υπάρχουν k κατηγορίες αγαθών. Χρησιμοποιούμε τους παρακάτω πίνακες για να μοντελοποιήσουμε το σύστημα της υπόληψης που θα επιδιώξουμε να σχεδιάσουμε σύμφωνα με τις απαιτήσεις που του χρήστη.

- ***Πίνακας U_grade***

Ο βαθμός u_grade(i, j) ,που κρατάμε στο κελί του πίνακα U_grade μεγέθους n x k , είναι ο μέσος όρος των βαθμών που έχει λάβει χρηστής i ($1 \leq i \leq n$) για συναλλαγές που έχει κάνει επί αγαθών κατηγορίας j ($1 \leq j \leq k$). Η ανανέωση δηλαδή των τιμών u_grade(i, j) γίνεται όταν ο χρήστης i λάβει έναν βαθμό g, για δοσοληψία στην οποία συμμετείχε αυτός καθαυτός, και μάλιστα η ανανέωση γίνεται ως εξής :

u_grade(i, j) = (u_grade(i, j) x N + g) / (N+1)

N : το πλήθος των δοσοληψιών επί αγαθού της  κατηγορίας j από τον χρήστη i

- ***Πίνακας U_Trans***

Με βάση τα παραπάνω, αλλά και όπως θα δούμε παρακάτω, προκύπτει η ανάγκη να κρατάμε το πλήθος των δοσοληψιών που έχει κάνει ο χρήστης i δίνοντας αγαθά της κατηγορίας j. Για αυτό το λόγο, έχουμε τον πίνακα U_Trans, μεγέθους n x k. Το κελί U_Trans(i, j) υποδεικνύει πόσες δοσοληψίες έχει κάνει ο χρήστης i δίνοντας αγαθά της κατηγορίας j. Η τιμή U_Trans(i, j) αυξάνεται κατά 1 όταν χρήστης i εκτελέσει συναλλαγή επί αγαθού κατηγορίας j.

- ***Πίνακας U_FinalGrade***

Ο πίνακας U_FinalGrade είναι μεγέθους n x k. Το κελί U_FinalGrade(i, j) φέρει τον βαθμό υπόληψης του χρήστη i όσον αφορά στις συναλλαγές της κατηγορίας αγαθών j. Η διαμόρφωση αυτού του βαθμού συντελείται με βάση τον βαθμό υπόληψης που έχει ο χρήστης

λόγω συναλλαγών που έχει κάνει ο ίδιος αλλά και τους βαθμούς υπόληψης των κοινοτήτων στις οποίες ανήκει, σε περίπτωση που δεν είναι «ανεξάρτητος» χρήστης. Είναι προφανές ότι ο βαθμός U_FinalGrade(i, j) είναι ίσος με τον βαθμό U_grade(i, j) στην περίπτωση του «ανεξάρτητου» χρήστη.

Η ανανέωση του U_FinalGrade(i, j) γίνεται όταν ο χρήστης λάβει βαθμό για συναλλαγή που έκανε επί αγαθού κατηγορίας j ή όταν κοινότητα στην οποία συμμετέχει εκτελέσει συναλλαγή επί αγαθού κατηγορίας j. Ανανέωση βέβαια έχουμε και στην περίπτωση εγγραφής/διαγραφής μέλους σε κοινότητα που συμμετέχει ο χρήστης I, υπό την προϋπόθεση ότι το μέλος αυτό έχει συναλλαχθεί επί αγαθού κατηγορίας j. Η ανανέωση αυτή λαμβάνει χωρά καθώς αυτή η ενέργεια εγγραφής/διαγραφής σε κοινότητα επηρεάζει τον βαθμό υπόληψης της κοινότητας και κατ' επέκταση και τον βαθμό κάθε μέλους της κοινότητας.

Αναφορικά με τις κοινότητες του συστήματός μας, θεωρούμε ότι υπάρχουν y κοινότητες και ότι ν από αυτές τις κοινότητες (όπου $\nu \leq y$) είναι «ενεργές», δηλαδή ν κοινότητες έχουν συμμετάσχει έστω και μία φορά σε μία συναλλαγή ή τουλάχιστον ένας χρήστης-μέλος τους έχει διεκπεραιώσει τουλάχιστον μία συναλλαγή.

- ***Πίνακας C_grade***

Ο βαθμός c_grade(i, j) ,που κρατάμε στο κελί του πίνακα C_grade μεγέθους v x k , είναι ο μέσος όρος των βαθμών που έχει λάβει κοινότητα i ($1 \leq i \leq v$) για συναλλαγές που έχει κάνει επί αγαθών κατηγορίας j ($1 \leq j \leq k$). Η ανανέωση δηλαδή των τιμών c_grade(i, j) γίνεται όταν ο κοινότητα i λάβει έναν βαθμό g, για δοσοληψία στην οποία συμμετείχε αυτή καθαυτή. Η ανανέωση γίνεται ως εξής :

$$c\_grade(i, j) = (c\_grade(i, j) \times N + g) / (N+1)$$

N : το πλήθος των δοσοληψιών επί αγαθού της κατηγορίας j από την κοινότητα i

- ***Πίνακας C_Trans***

Το κελί C_Trans(i, j) του πίνακα C_Trans, μεγέθους v x k, φέρει έναν θετικό ακέραιο αριθμό που δηλώνει το πλήθος συναλλαγών που έχει εκτελέσει η κοινότητα αυτή καθεαυτή προσφέροντας αγαθά κατηγορίας j. Η τιμή του αυξάνεται κατά 1 κάθε φορά που η κοινότητα προσφέρει σε συναλλαγή αγαθό που ανήκει στην κατηγορία j.

- ***Πίνακας C_FinalGrade***

Ο πίνακας C_FinalGrade είναι μεγέθους v x k. Το κελί C_FinalGrade(i, j) φέρει τον βαθμό υπόληψης της κοινότητας i όσον αφορά στις συναλλαγές επί της κατηγορίας αγαθών j. Ο

βαθμός αυτός διαμορφώνεται από τους βαθμούς υπόληψης που έχουν τα μέλη της κοινότητας και από τον βαθμό που έχει λάβει η κοινότητα για συναλλαγές επί αγαθών κατηγορίας j.

Η ανανέωση αυτού του βαθμού λαμβάνει χώρα κάθε φορά που η κοινότητα ή κάποιο μέλος της συναλλάσσεται προσφέροντας αγαθό κατηγορίας j. Ανανέωση έχουμε και κατά την εγγραφή/διαγραφή χρήστη-μέλους από την κοινότητα.

3.2.3.1 Διαμόρφωση Βαθμού Υπόληψης Οντότητας

- ***Βαθμός Υπόληψης Κοινότητας***

Όταν μία κοινότητα i ($1 \le i \le \nu$), που έχει n_i ($n_i \le n$) το πλήθος χρήστες, εκτελέσει συναλλαγή προσφέροντας αγαθό από την κατηγορία j ($1 \le j \le k$) λαμβάνει βαθμό υπόληψης g ($0 \le g \le 10$). Ο βαθμός αυτός επηρεάζει τον μέσο όρο των βαθμών υπόληψης που έχει λάβει η κοινότητα λόγω συναλλαγών που έχει εκτελέσει αυτή καθεαυτή επί της συγκεκριμένης κατηγορίας αγαθών. Άρα, η εικόνα της υπόληψης της κοινότητας λόγω των δικών της ενεργειών διαμορφώνεται ως εξής :

$$c\_grade(i, j) = [\ c\_grade(i, j\ ) \times c\_trans(i, j)\ + g\ ] / [c\_trans(i, j) +1]$$

$$c\_trans(i, j) = c\_trans(\ i, j) + 1$$

Δηλαδή ο μέσος όρος των βαθμών που έχει λάβει η κοινότητα i λόγω συναλλαγών που έχει διεκπεραιώσει καθώς και το πλήθος συναλλαγών που έχει κάνει η ίδια η κοινότητα για την κατηγορία αγαθών j ανανεώνεται.

Η εικόνα της υπόληψης μίας κοινότητας δε διαμορφώνεται μόνο από την αξιοπιστία που έχει επιδείξει λόγω των δικών της συναλλαγών και μόνο. Στην εικόνα της υπόληψης της κοινότητας συμβάλλει και η εικόνα υπόληψης των μελών της. Σε αυτό το σημείο πρέπει να εισάγουμε την επίδραση των βαθμών υπόληψης των μελών της κοινότητας. Εκφράζουμε λοιπόν το συνολικό βαθμό υπόληψης της κοινότητας ως συνισταμένη δύο παραγόντων, του αμιγώς δικού της βαθμού υπόληψης λόγω των συναλλαγών που έχει η ίδια εκτελέσει και τον μέσο όρο βαθμών υπόληψης των μελών της :

$$\text{C\_FinalGrade(i,j)} = w \cdot C\_grade(i, j) + (1 - w) \cdot [\sum_{\substack{1 \le l \le n \\ user\_l \in community\_i}} U\_grade(l, j)] / m$$

Όπου w : ποσοστό επίδρασης του βαθμού που έχει η κοινότητα λόγω συναλλαγών

που έχει κάνει αυτή καθεαυτή ($0 \le w \le 1$)

m : πλήθος ενεργών μελών της κοινότητας i ($0 \le m \le n_i \le n$)

Ο παράγοντας $\sum_{\substack{1 \le l \le n \\ user\_l \in community\_i}} U\_grade(l, j) / m$ αποτελεί τον μέσο όρο των βαθμών υπολήψεων των χρηστών - μελών της κοινότητας i. Οι βαθμοί αυτοί είναι οι βαθμοί που έλαβαν αυτοί οι χρήστες για συναλλαγές που έκαναν αυτοί καθαυτοί. Όπως παρατηρούμε αυτή η συνιστώσα επιδρά κατά ένα ποσοστό (1-w) στο συνολικό βαθμό της υπόληψης της κοινότητας ενώ ο βαθμός λόγω των συναλλαγών που έχει κάνει αυτή καθαυτή επιδρά κατά ένα ποσοστό w.

Τίθεται λοιπόν το θέμα προσδιορισμού του ποσοστού w. Σε μία πρώτη προσέγγιση, το ποσοστό αυτό θα μπορούσε να είναι σταθερό, έστω π.χ. w = 0,5. Η επιλογή όμως αυτή δεν παρουσιάζει καμία προσαρμοστικότητα στα δεδομένα και θα μπορούσε να οδηγήσει σε μία όχι και τόσο δίκαιη εικόνα της υπόληψης της κοινότητας. Ας αναλογιστούμε την περίπτωση κατά την οποία τα μέλη μίας κοινότητας βρίσκονται σε καλή συνεργασία μεταξύ τους και αποφασίζουν να ενεργήσουν ως επί το πλείστον ως ολότητα. Τα μέλη τα οποία ενεργούν αυτόνομα είναι ελάχιστα, έστω 2. Θεωρούμε ότι η κοινότητα έχει εκτελέσει πολλές συναλλαγές για την κατηγορία αγαθών j, π.χ. 50 συναλλαγές, με μέσο όρο βαθμών υπόληψης c_grade = 10. Αντίστοιχα, τα 2 ενεργά μέλη της κοινότητας έχουν εκτελέσει 2 συνολικά συναλλαγές με μέσο όρο βαθμών συναλλαγών 1. Είναι προφανές ότι η κοινότητα αυτή παρουσιάζει αξιόπιστη συμπεριφορά που δεν είναι δίκαιο να αμαυρωθεί από την συμπεριφορά των δυο μη αξιόπιστων χρηστών της. Ας δούμε πως διαμορφώνεται η εικόνα της υπόληψης της κοινότητας , με σταθερό ποσοστό w = 0,5. Ισχύει :

$$\text{C\_FinalGrade(i, j)} = w \cdot C\_grade(i, j) + (1 - w) \cdot [\sum_{\substack{1 \le l \le n \\ user\_l \in community\_i}} U\_grade(l, j)] / m =$$

$$= 0.5 \text{ x } 10 + 0.5 \text{ x } 1 = 5 + 0.1 = 5.1$$

Από το παραπάνω αποτέλεσμα παρατηρούμε ότι τα δυο μη ευυπόληπτα μέλη της κοινότητας, με τις λίγες το πλήθος συναλλαγές τους, αμαύρωσαν την εικόνα υπόληψης της κατά τα άλλα αξιόπιστης κοινότητας. Αυτό δεν είναι θεμιτό. Προσεγγίζοντας γενικότερα το θέμα έχουμε να παρατηρήσουμε ότι μία κοινότητα θα πρέπει να έχει βαθμό που να ανταποκρίνεται περισσότερο στην επίδραση του παράγοντα που έχει την πιο έντονη παρουσία όσον αφορά στην δραστηριοποίηση της κοινότητας και των μελών της. Δηλαδή, μία κοινότητα πρέπει να έχει υψηλό βαθμό υπόληψης αν η ίδια πραγματοποιεί πολλές συναλλαγές αξιόπιστα, ενώ έχει σχετικά μη δραστήρια μέλη που έχουν εκτελέσει λίγες μη αξιόπιστες συναλλαγές. Αντίστοιχα, μία κοινότητα θεωρείται εξίσου αξιόπιστη αν έχει πολλές καλές συναλλαγές που

εκτελούνται από τα μέλη της αυτά καθαυτά, αλλά η ίδια παραμένει ανενεργή ή με χαμηλή δραστηριότητα. Προκύπτει λοιπόν η ανάγκη διαμόρφωσης του ποσοστού επίδρασης των συνιστωσών υπόληψης, με τρόπο που να αποκρίνεται στην πιο δίκαιη εικόνα υπόληψης της κοινότητας.

Έστω MeanU_Trans(i , j) το μέσο πλήθος συναλλαγών ενός ενεργού μέλους της κοινότητας i για την κατηγορία αγαθών j. Η τιμή αυτή υποδεικνύει πόσες συναλλαγές έχει εκτελέσει ο «μέσος» ενεργός χρήστης της κοινότητας, προσφέροντας αγαθά της κατηγορίας j. Το ποσοστό w διαμορφώνεται ως εξής :

$$w = \frac{C\_Trans(i,j)}{MeanU\_Trans(i,j) + C\_Trans(i,j)}$$

όπου MeanU_Trans(i, j) = $\sum_{\substack{1 \le l \le n \\ user\_l \in community\_i}} U\_Trans(l,j) / m$

m : το πλήθος των ενεργών χρηστών της κοινότητας i ($0 \le m \le n_i \le n$)

Με αυτό τον τρόπο ο βαθμός της υπόληψης της κοινότητας εκφράζει τις συναλλαγές τις οποίες έχει εκτελέσει η ίδια και επηρεάζεται καθοριστικά και από το προφίλ υπόληψης των μελών που την συνθέτουν. Η συνδυαστική εικόνα που προκύπτει είναι μία δίκαιη εικόνα που αποκρίνεται στην πραγματική υπόληψης της κοινότητας καθώς δίνει μεγαλύτερο βάρος στην συνιστώσα που, λόγω πιο έντονης δραστηριότητας, πρέπει να χαρακτηρίσει περισσότερο την κοινότητα.

Με βάση την παραπάνω διαμόρφωση του βαθμού υπόληψης της κοινότητας, στο παράδειγμα που αναφέραμε παραπάνω, θα έχουμε τα παρακάτω αποτελέσματα:

MeanU_Trans(i, j) = $\sum_{\substack{1 \le l \le n \\ user\_l \in community\_i}} U\_Trans(l,j) / m$ = (1 + 1) / 2 = 1

$$w = \frac{C\_Trans(i,j)}{MeanU\_Trans(i,j) + C\_Trans(i,j)} = \frac{50}{1+50} = 0{,}98$$

C_FinalGrade(i, j) = $w \cdot C\_grade(i,j) + (1-w) \cdot (\sum_{\substack{1 \le l \le n \\ user\_l \in community\_i}} U\_grade(l,j)) / m =$

= 0.98 x 10 + (1 - 0.98) x 1 = 9.8 + 0.02 = 9.82

Είναι προφανές ότι η εικόνα αυτή της υπόληψης της κοινότητας ανταποκρίνεται πολύ περισσότερο στην πραγματική εικόνα της.

- ***Βαθμός Υπόληψης Χρήστη***

Ο βαθμός υπόληψης χρήστη επηρεάζεται από δύο συνιστώσες, την βαθμολόγηση που έχει λάβει ο χρήστης λόγω των συναλλαγών που έχει εκτελέσει ο ίδιος και την υπόληψη που έχουν οι κοινότητες στις οποίες συμμετέχει. Αυτό σημαίνει ότι η διαμόρφωση της υπόληψης ενός χρήστη είναι ανάλογη με τις κατηγορίες στις οποίες ανήκει ο χρήστης. Επομένως μπορούμε να διακρίνουμε τις εξής περιπτώσεις:

a. «Ανεξάρτητος» Χρήστης
b. Χρήστης που ανήκει σε κοινότητες οι οποίες παρουσιάζονται ανενεργές για τη συγκεκριμένη κατηγορία αγαθών
c. Χρήστης που ανήκει σε m το πλήθος ενεργές κοινότητες όσον αγορά στην συγκεκριμένη κατηγορία αγαθών ($0 < m \le v$)

Στις πρώτες δύο περιπτώσεις, είναι προφανές ότι ο βαθμός υπόληψης του χρήστη i για την κατηγορία αγαθών j διαμορφώνεται μόνο με βάση τους βαθμούς που έχει λάβει ο χρήστης για συναλλαγές που έχει κάνει ο ίδιος για την συγκεκριμένη κατηγορία αγαθών. Οι βαθμοί υπόληψης άλλων οντοτήτων του συστήματός μας δεν επηρεάζουν το βαθμό υπόληψης του χρήστη. Επομένως, στις πρώτες δύο περιπτώσεις ισχύει:

$$\text{U\_FinalGrade(i, j)} = U\_grade(i, j)$$

Στην τρίτη περίπτωση, στον βαθμό υπόληψης του χρήστη επιδρά και η συνιστώσα που εκφράζει τους βαθμούς υπόληψης των κοινοτήτων στις οποίες συμμετέχει ο χρήστης. Συγκεκριμένα, για την περίπτωση που ο χρήστης i ανήκει σε κοινότητες που έχουν βαθμό υπόληψης για την κατηγορία αγαθών j, εκφράζουμε την υπόληψη του χρήστη μας για κατηγορία αγαθών j με την παρακάτω τιμή :

$$\text{U\_FinalGrade(i, j)} = w \cdot U\_grade(i, j) + (1 - w) \cdot \Big(\sum_{\substack{1 \le l \le v \\ user\_i \in community\_l}} C\_FinalGrade(l, j) \Big) / m$$

Όπου w : ποσοστό επίδρασης του βαθμού που έχει ο χρήστης ανεξαρτήτως των κοινοτήτων στις οποίες μετέχει($0 \le w \le 1$)

m : πλήθος των ενεργών κοινοτήτων στις οποίες μετέχει ο χρήστης i ($0 < m \le v$)

Το ποσοστό w θα προσδιοριστεί αντίστοιχα όπως και για τον βαθμό υπόληψης κοινότητας, που αναλύσαμε παραπάνω. Αρχικά ας αναλογιστούμε την επίδραση ενός σταθερού ποσοστού w στην εικόνα υπόληψης του χρήστη. Έστω ότι ο χρήστης i έχει εκτελέσει 20 συναλλαγές προσφέροντας αγαθά από την κατηγορία j και έχει λάβει καλή βαθμολογία, π.χ. 10, σε όλες τις συναλλαγές. Αντίστοιχα ο χρήστης ανήκει σε 1 κοινότητα η οποία δεν παρουσιάζεται εξίσου ευυπόληπτη για αυτή την κατηγορία αγαθών, αν και έχει μικρή δραστηριότητα στην συγκεκριμένη κατηγορία αγαθών. Συγκεκριμένα, ο μέσος όρος των τελικών βαθμών υπόληψής της κοινότητας στην συγκεκριμένη κατηγορία είναι 2 και προκύπτει επειδή οι ίδιες έχουν εκτελέσει 3 συναλλαγές. Τότε, ο βαθμός υπόληψης του χρήστη στην κατηγορία αγαθών j, θεωρώντας σταθερό ποσοστό w =0.5, θα είναι:

$$\text{U\_FinalGrade(i, j)} = w \cdot U\_grade(i, j) + (1 - w) \cdot \Big(\sum_{\substack{1 \le l \le v \\ user\_i \in community\_l}} C\_FinalGrade(l, j) \Big) / m$$

= 0.5 x 10 + 0.5 x 2 = 5 + 1 = 6

Από το παραπάνω αποτέλεσμα προέκυψε ότι ενώ ο χρήστης έχει πολύ αξιόπιστη παρουσία κατά τις συναλλαγές του, η εικόνα των κοινοτήτων στις οποίες ανήκει αμαύρωσε κατά πολύ την εικόνα του. Άρα, μια σταθερή τιμή του ποσοστού w δεν είναι αποδεκτή.

Για τον προσδιορισμό του ποσοστού w, θα πρέπει να καθορίσουμε τις συνθήκες υπό τις οποίες η μία συνιστώσα της υπόληψης του χρήστη είναι πιο βαρύνουσα έναντι της άλλης. Είναι προφανές ότι όταν ένας χρήστης έχει κάνει περισσότερες συναλλαγές για μία κατηγορία αγαθών, ενώ οι κοινότητες στις οποίες ανήκει δεν έχουν έντονη δραστηριότητα για αυτή την κατηγορία, οι βαθμοί που έχει λάβει ο χρήστης για αυτή την κατηγορία είναι αυτοί που θα προσδιορίσουν με μεγαλύτερη βαρύτητα την εικόνα υπόληψης του χρήστη. Έτσι, το ποσοστό w διαμορφώνεται ως εξής:

$$w = \frac{U\_Trans(i, j)}{MeanC\_Trans(i, j) + U\_Trans(i, j)}$$

Η παράμετρος MeanC_Trans(i, j) υποδεικνύει την δραστηριότητα επί κατηγορίας αγαθών j των ενεργών κοινοτήτων στις οποίες ανήκει ο χρήστης i. Η παράμετρος αυτή θα πρέπει να φέρει την πληροφορία για το πλήθος των συναλλαγών που έχουν εκτελέσει οι κοινότητες στις οποίες ανήκει ο χρήστη. Όμως, για να προσδιορίζουμε πόσες συναλλαγές έχει κάνει μία κοινότητα, θα πρέπει εκτός από τις συναλλαγές που έχει εκτελέσει η ίδια η κοινότητα να συνυπολογίσουμε και τον παράγοντα του πλήθους των συναλλαγών που έχουν εκτελέσει τα μέλη των κοινοτήτων αυτών. Άλλωστε, οι συναλλαγές των μελών την κοινότητας συμβάλλουν στην διαμόρφωση της εικόνας υπόληψης της κοινότητας και επομένως οι

ενέργειες ενός μέλους μίας κοινότητας επηρεάζει έμμεσα τον βαθμό υπόληψης των υπολοίπων μελών της κοινότητας.

Η δραστηριότητα λοιπόν μίας κοινότητας μπορεί να εκφραστεί από την παρακάτω τιμή:

$$\text{CommunityActivity(i, j)} = \frac{TotalTransOfActiveUsers(i,j) + C\_Trans(i,j)}{NoOfActiveUsers(i,j) + 1}$$

Όπου TotalTransOf ActiveUsers(i, j) : το συνολικό πλήθος συναλλαγών που έχουν εκτελέσει τα μέλη της κοινότητας i προσφέροντας αγαθά κατηγορίας j

C_Trans : το πλήθος συναλλαγών που έχουν εκτελέσει η ίδια η κοινότητα i προσφέροντας αγαθά κατηγορίας j

NoOf ActiveUsers :το πλήθος των ενεργών μελών της κοινότητας i για την κατηγορία j

Ο τρόπος με τον οποίο προσδιορίσαμε την δραστηριότητα της κοινότητας υποδεικνύει ότι οι συναλλαγές που κάνουν τα μέλη της κοινότητας έχουν την ίδια βαρύτητα με τις συναλλαγές που κάνει η ίδια η κοινότητα.

Μια διαφορετική προσέγγιση θα έδινε για την δραστηριότητα της κοινότητας την τιμή :

$$\text{CommunityActivity(i, j)} = \frac{MeanU\_Trans(i,j) + C\_Trans(i,j)}{2}$$

Όπου MeanU_Trans : το μέσο πλήθος συναλλαγών των ενεργών χρηστών της κοινότητας i για την κατηγορία j

C_Trans : το πλήθος συναλλαγών που έχουν εκτελέσει η ίδια η κοινότητα i προσφέροντας αγαθά κατηγορίας j

Η προσέγγιση αυτή θεωρεί ότι για τον προσδιορισμό της δραστηριότητας της κοινότητας μεγαλύτερη βαρύτητα έχουν οι συναλλαγές της ίδιας της κοινότητας καθώς η δραστηριότητα εξαρτάται με ποσοστό 50% από τον αριθμό των συναλλαγών που έχει κάνει η ίδια η κοινότητα και 50% από τον μέσο αριθμό συναλλαγών των μελών της.

Για τις ανάγκες του δικού μας συστήματος υπόληψης θα υιοθετήσουμε την πρώτη προσέγγιση που υποδεικνύει ισότιμο υπολογισμό του αριθμού των συναλλαγών της κοινότητας και τον αριθμό συναλλαγών των μελών της.

Έχουμε λοιπόν εκφράσει την δραστηριότητα μίας κοινότητας i επί αγαθών κατηγορίας j με την βοήθεια της παραμέτρου CommunityActivity(i, j). Χρειάζεται λοιπόν σε αυτό το σημείο να λάβουμε μία εικόνα για την μέση δραστηριότητα των κοινοτήτων στις οποίες ανήκει ο χρήστης μας. Οπότε εισάγουμε και την παρακάτω παράμετρο:

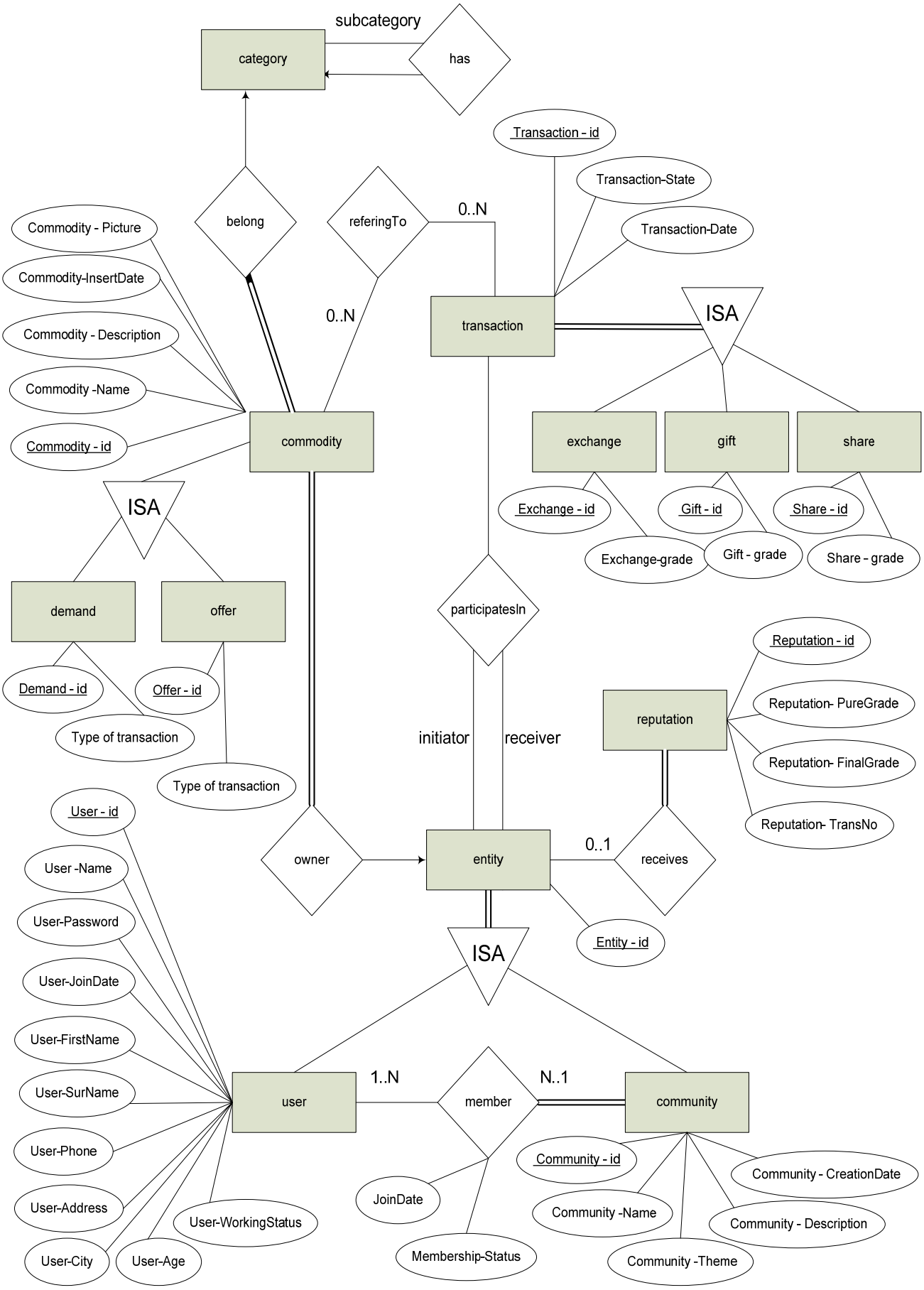
$$\text{MeanC\_Trans}(i, j) = \left(\sum_{\substack{1 \le c \le v \\ user\_i \in community\_c}} \text{CommunityActivity}(c, j) \right) / n$$

Όπου n : το πλήθος των ενεργών κοινοτήτων όσον αφορά στην κατηγορία αγαθών j στις οποίες ανήκει ο χρήστης i

Παρατήρηση : Οι τύποι υπολογισμού της υπόληψης χρήστη και της υπόληψης κοινότητας είναι διαμορφωμένοι για να εκφράζουν τις εγγενείς αλληλεξαρτήσεις που πρέπει να υφίστανται στο σύστημα της υπόληψής μας. Η κοινότητα θεωρείται ως το σύνολο που αποτελείται από χρήστες και έτσι ο βαθμός της διαμορφώνεται από αυτούς, ενώ ο χρήστης αποτελεί ένα μέρος της κοινότητας και γι' αυτό επηρεάζεται από τον βαθμό υπόληψής της. Με αυτή τη λογική, στον τύπο υπολογισμού της υπόληψης της κοινότητας , η συνιστώσα των βαθμών υπόληψης των χρηστών της αναφέρεται στις συναλλαγές που έχουν κάνει οι χρήστες και τους βαθμούς που έχουν λάβει ανεξαρτήτως άλλων οντοτήτων. Από την άλλη μεριά, στον τύπο υπολογισμού της υπόληψης χρήστη, η συνιστώσα που εκφράζει την επίδραση των κοινοτήτων στις οποίες ανήκει ο χρήστης δεν περιέχει μόνο την επίδραση των βαθμών που έχουν λάβει οι κοινότητες λόγω των δικών τους συναλλαγών. Περιλαμβάνει και την επίδραση των βαθμών που έχουν λάβει τα υπόλοιπα μέλη της κοινότητας. Αυτό είναι θεμιτό, καθώς η συμπεριφορά ενός μέλους μίας κοινότητας θα πρέπει να επηρεάζει και την εικόνα των υπόλοιπων μελών της ίδιας κοινότητας.

3.3 Μοντέλο Οντοτήτων Συσχετίσεων

Στο εδάφιο αυτό θα παρουσιάσουμε τη λογική διάταξη του συστήματός μας μέσω του μοντέλου οντοτήτων συσχετίσεων.


subcategory
category
has
belong
referingTo
0..N
0..N
Transaction - id
Transaction-State
Transaction-Date
Commodity - Picture
Commodity-InsertDate
Commodity - Description
Commodity -Name
Commodity - id
transaction
ISA
commodity
exchange
gift
share
Exchange - id
Gift - id
Share - id
Exchange-grade
Gift - grade
Share - grade
ISA
demand
offer
participatesIn
Demand - id
Offer - id
Type of transaction
Type of transaction
initiator
receiver
Reputation - id
reputation
Reputation- PureGrade
Reputation- FinalGrade
Reputation- TransNo
User - id
User -Name
User-Password
User-JoinDate
User-FirstName
User-SurName
User-Phone
User-Address
User-City
User-Age
User-WorkingStatus
owner
entity
0..1
receives
Entity - id
ISA
user
1..N
member
N..1
community
JoinDate
Membership-Status
Community - id
Community -Name
Community -Theme
Community - CreationDate
Community - Description

4

Σχεδίαση Συστήματος.

4.1 Αρχιτεκτονική

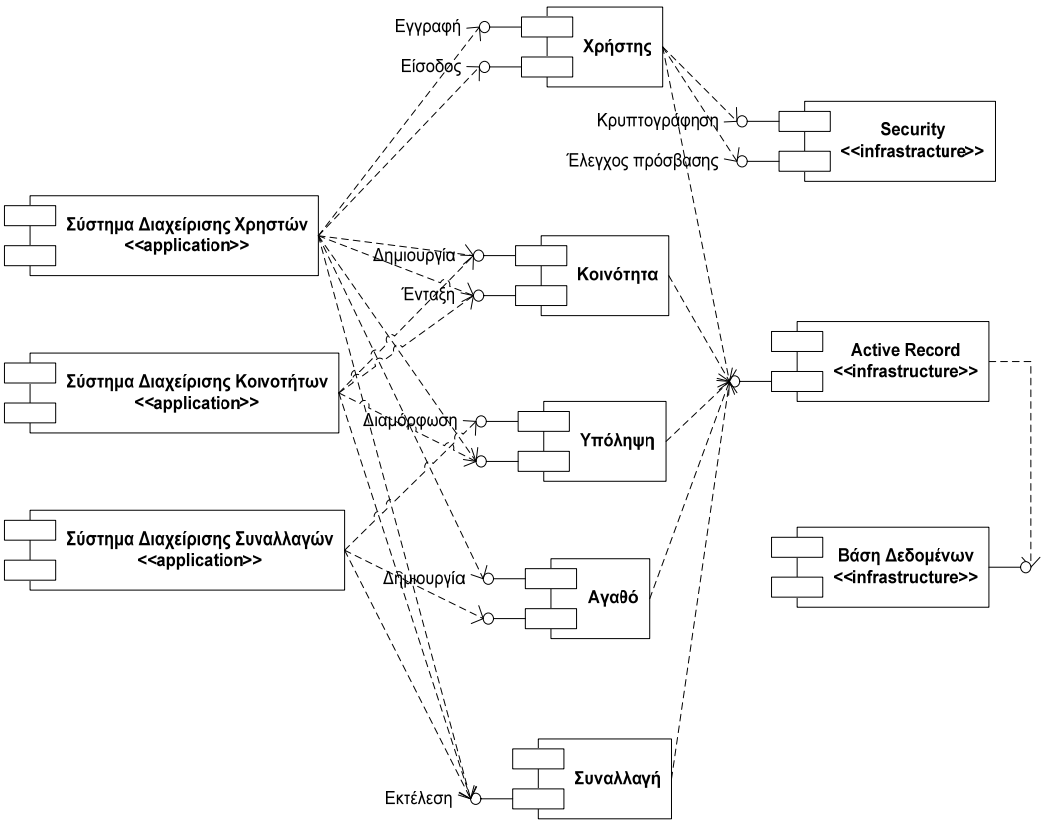
Η τεχνοτροπία που ακολουθούμε είναι της αρχιτεκτονικής MVC. Θα αναπτύξουμε τις υπομονάδες με βάση το αντικειμενοστραφές μοντέλο και τον λογικό διαχωρισμό τους παρουσιάζοντας ως πυρήνα το Model.

1. Model : οι πίνακες της βάσης δεδομένων, η λογική αυτών των πινάκων σύμφωνα με όσα περιγράψαμε στην προδιαγραφή απαιτήσεων
2. View : οι προβολή των δεδομένων και γενικά όλες οι σελίδες που παρουσιάζονται στον χρήστη
3. Controller : η διαχείριση των εισόδων και η διαμόρφωση της παρουσίασης της πληροφορίας του χρήστη.

Σύμφωνα λοιπόν με τα παραπάνω, κάθε πίνακας της βάσης μας αποτελεί υπομονάδα του μοντέλου. Συγκεκριμένα, οι πίνακες της βάσης μας αντιστοιχούν σε κλάσεις του μοντέλου, και τα attributes των πινάκων της βάσης μας είναι μεταβλητές της κλάσεις. Κάθε εγγραφή του πίνακα μοντελοποιείται από στιγμιότυπο της αντίστοιχης κλάσης.

Οι views αποτελούνται από τα html που παρουσιάζονται στον χρήστη και οι controllers χειρίζονται κάθε input (στην περίπτωσή μας κάθε ενέργεια του χρήστη η οποία παράγει HTTP μήνυμα που αποστέλλει ο browser στον server) και παρέχουν στα views τα κατάλληλα δεδομένα αλληλεπιδρώντας και με την βάση δεδομένων.

Το παρακάτω διάγραμμα αποτελεί το ψηφιδικό διάγραμμα του συστήματός μας:



4.2 Περιγραφή Κλάσεων

Επειδή όπως προαναφέραμε ο κώδικάς μας ακολούθησε το αντικειμενοστραφές μοντέλο, παρακάτω παρουσιάζουμε τις κλάσεις ομαδοποιημένες σύμφωνα με τις βασικές κλάσεις του μοντέλου μας. Η ιδιαιτερότητα που παρουσιάζεται στο σύστημά μας είναι ότι ο σχεδιασμός του θα ακολουθήσει την αρχιτεκτονική τεχνοτροπία MVC που σημαίνει ότι μία κλάση του μοντέλου δεν περιέχει όλες τις μεθόδους που αφορούν άμεσα το μοντέλο αυτό, καθώς η διαχείριση του μοντέλου συντελείται από τους controllers. Παρά την παραπάνω ιδιομορφία, παρουσιάζουμε τις κλάσεις του συστήματός μας, με πυρήνα το μοντέλο μας, και με αυτό τον άξονα περιγράφουμε και τη σχεδίαση των βασικών μεθόδων του συστήματός μας.

4.2.1 Οντότητα Χρήστης

Η οντότητα χρήστης μοντελοποιείται από την κλάση User του μοντέλου μας. Στο εδάφιο αυτό θα παρουσιάσουμε τις βασικές λειτουργίες που αφορούν την οντότητα χρήστη.

4.2.1.1 Sign up – Εγγραφή χρήστη

Προϋποθέσεις : Ο χρήστης έχει επιλέξει sign up στην αρχική σελίδα

Περιγραφή : Είσοδος σε κατάλληλη φόρμα/σελίδα που θα συμπληρώσει ο χρήστης για να γίνει μέλος του συστήματός μας

- ***Ο χρήστης καλείται να συμπληρώσει τα στοιχεία του***
  - Ο χρήστης εισάγει στοιχεία όπως π.χ. username, password, first name, surname, e-mail, phone-number (*), address (*), ιδιότητα (*), age (*) κλπ, όπου τα πεδία με * είναι προαιρετικά.
  - Ο χρήστης κάνει submit την φόρμα με τα στοιχεία του.
- ***Αστοχία φόρμας***

  Αν η φόρμα δεν είναι πλήρης ή δεν είναι αποδεκτή ο χρήστης ενημερώνεται με κατάλληλο μήνυμα και επανέρχεται στην σελίδα συμπλήρωσης των στοιχείων του.
- ***Ενημέρωση Βάσης Δεδομένων***

  Σε περίπτωση επιτυχούς υποβολής της φόρμας συμπλήρωσης στοιχείων χρήστη, η Βάση Δεδομένων του συστήματος ενημερώνεται κατάλληλα.
- ***Επιβεβαίωση μέσω email***

  Ο χρήστης λαμβάνει κατάλληλο mail που τον ενημερώνει για την επιτυχή του εγγραφή στο σύστημα καθώς και για τα στοιχεία λογαριασμού του.
- ***Πλοήγηση στην Login Page***

  Αυτόματα, μετά το submit, στον χρήστη παρουσιάζεται σελίδα στην οποία οδηγείται στην Login Page που τον παροτρύνει να κάνει sign in

4.2.1.2 Sign In – Είσοδος χρήστη

Προϋποθέσεις : Ο χρήστης έχει ήδη εισαχθεί στην πρώτη σελίδα του συστήματος

Περιγραφή : Ο χρήστης εισέρχεται στο σύστημα

- ***Εισαγωγή Στοιχείων Χρήστη***

Ο χρήστης εισάγει το username του και το password του στα κατάλληλα πεδία της Login Page.

- ***Έλεγχος Στοιχείων Χρήστη***

  Εκτελείται κατάλληλος έλεγχος στην Β.Δ. για την εγκυρότητα του username του και το password του.

- ***Αποτέλεσμα Εγκυρότητας Στοιχείων Χρήστη***

  Αν έχει εισάγει έγκυρα στοιχεία εισέρχεται στο σύστημα.

  Εναλλακτικά : Αν δεν έχει εισάγει έγκυρα στοιχεία ενημερώνεται κατάλληλα και παραμένει στην αρχική σελίδα.

4.2.1.3 Enter Site – Προβολή Profile Χρήστη

Προϋποθέσεις :

- Ο χρήστης έχει ήδη κάνει sign up στο σύστημα
- Ο χρήστης έχει εισάγει σωστά το username και password του στην αρχική ιστοσελίδα.

Περιγραφή : Ο χρήστης εισέρχεται στην σελίδα που παρουσιάζει το προφίλ του και τις δυνατότητες αλληλεπίδρασής του με το σύστημα. Ο χρήστης, λοιπόν, αφού εισάγει σωστά τα στοιχεία του εισέρχεται στο σύστημα και αρχικά βλέπει μία σελίδα η οποία του παρουσιάζεται διαμορφωμένη με βάση προσωπικά του στοιχεία. Τέτοια στοιχεία είναι :

- Συνολικός Βαθμός Υπόληψης Χρήστη
- Κοινότητες στις οποίες συμμετέχει

Ο χρήστης πλέον έχει δυνατότητα να αλληλεπιδράσει πλήρως με το σύστημά μας. Οι δυνατότητες που του παρέχονται είναι οι παρακάτω:

- Δυνατότητα να εισάγει ένα αγαθό στο σύστημα.
- Δυνατότητα να αναζητήσει αγαθό.
- Δυνατότητα να εκτελέσει μία συναλλαγή.
- Δυνατότητα να δει τις συναλλαγές που εκκρεμούν προς εκτέλεση και τον αφορούν.
- Δυνατότητα να διαμορφώσει το προφίλ του.
- Δυνατότητα να δει τις υπάρχουσες κοινότητες του συστήματος και τα μέλη της καθεμιάς
- Δυνατότητα να γίνει μέλος μίας κοινότητας

- Δυνατότητα να διαγραφεί από μία κοινότητα
- Δυνατότητα να ορίσει μία κοινότητα

  Ο χρήστης τότε εισέρχεται σε κατάλληλη σελίδα όπου του ζητείται να δώσει όνομα στην κοινότητα, μια περιγραφή της καθώς και το συνδετικό κρίκο των χρηστών-μελών της. Ορίζοντας την κοινότητα παίρνει δικαιώματα administrator. Στο σύστημά μας, λοιπόν, ο administrator μιας κοινότητας είναι ο δημιουργός της. Αυτό σημαίνει ότι μόλις ένας χρήστης δημιουργήσει μία κοινότητα, αποτελεί μέλος της με δικαιώματα administrator. Ο administrator μίας κοινότητας μπορεί :

  - να παρέμβει στο προφίλ της κοινότητας
  - να εκτελέσει συναλλαγές εκ μέρους της
  - να προτείνει έναν χρήστη προς αποχώρηση

Όσον αφορά το προφίλ του, ο χρήστης έχει τη δυνατότητα να ορίσει ως προσωπικά του στοιχεία τα παρακάτω:

- φωτογραφίες
- ένα πεδίο myMoto που αποτελεί ένα προσωπικό μήνυμα που τον εκφράζει
- highlight συναλλαγή στην οποία έχει συμμετάσχει
- μία περιγραφή του εαυτού του

Το προφίλ αυτό, μαζί με την επιπρόσθετη πληροφορία των βαθμών υπόληψης του χρήστη, που προκύπτει από το υποσύστημα υπόληψης, και τους οποίους δεν μπορεί να αλλάξει ο χρήστης, θα βλέπουν οι υπόλοιποι χρήστες αν κάνουν αναζήτηση των στοιχείων του .

4.2.2 Οντότητα Κοινότητα

Η οντότητα χρήστης μοντελοποιείται από την κλάση Community του μοντέλου μας. Στο εδάφιο αυτό θα παρουσιάσουμε τις βασικές λειτουργίες που αφορούν την οντότητα κοινότητα.

4.2.2.1 Δημιουργία κοινότητας – Create a Community

Προϋποθέσεις :

- Ο χρήστης έχει ήδη κάνει sign up στο σύστημα

- Ο χρήστης έχει εισάγει σωστά το username και password του στην αρχική ιστοσελίδα.

Περιγραφή : Είσοδος σε κατάλληλη φόρμα/σελίδα που θα συμπληρώσει ο χρήστης για να δημιουργήσει την κοινότητα που επιθυμεί

- ***Ο χρήστης καλείται να συμπληρώσει τα στοιχεία που θέλει να έχει η κοινότητα***
  - Ο χρήστης εισάγει στοιχεία όπως το όνομα της κοινότητας, ένα θέμα ενδιαφέροντος με το οποίο να ασχολείται και μια περιγραφή.
  - Ο χρήστης κάνει submit την φόρμα με τα στοιχεία του.
- ***Αστοχία φόρμας***

  Αν η φόρμα δεν είναι πλήρης ή δεν είναι αποδεκτή ο χρήστης ενημερώνεται με κατάλληλο μήνυμα και επανέρχεται στην σελίδα συμπλήρωσης των στοιχείων του.
- ***Ενημέρωση Βάσης Δεδομένων***

  Η Βάση Δεδομένων του συστήματος ενημερώνεται κατάλληλα.

4.2.2.2 Συμμετοχή σε κοινότητα – Join a Community

Προϋποθέσεις :

- Ο χρήστης έχει ήδη κάνει sign up στο σύστημα
- Ο χρήστης έχει εισάγει σωστά το username και password του στην αρχική ιστοσελίδα.
- Υπάρχει η κοινότητα στην οποία θέλει να εγγραφεί ο χρήστης

Περιγραφή : Ο χρήστης επιλέγει μέσα από μία λίστα των κοινοτήτων που υπάρχουν, αυτή στην οποία επιθυμεί και περιμένει την έγκριση από τον moderator της.

- ***Ο χρήστης καλείται να επιλέξει την επιθυμητή κοινότητα***
  - Παρουσιάζεται στον χρήστη μια λίστα με τις διαθέσιμες κοινότητες.
  - Ο χρήστης πατάει το κουμπί join στην κοινότητα της επιλογής του.
- ***Ο administrator εγκρίνει / απορρίπτει την αίτηση του χρήστη***

  Μέσα από μια λίστα με αιτήσεις, ο administrator εγκρίνει ή απορρίπτει κάθε μία ξεχωριστά.
- ***Ενημέρωση Βάσης Δεδομένων***

  Η Βάση Δεδομένων του συστήματος ενημερώνεται κατάλληλα.

4.2.2.3 Προβολή μελών κοινότητας – Show Members

Προϋποθέσεις :

- Ο χρήστης έχει ήδη κάνει sign up στο σύστημα
- Ο χρήστης έχει εισάγει σωστά το username και password του στην αρχική ιστοσελίδα.
- Υπάρχει η κοινότητα της οποίας ο χρήστης θέλει να δει τα μέλη

Περιγραφή : Ο χρήστης επιλέγει να δει τα μέλη μιας κοινότητας.

- ***Επιλογή Κοινότητας***

  Ο χρήστης μέσα από ένα μενού διαλέγει την κοινότητα, της οποίας θέλει να δει τα μέλη. Αφού κάνει την επιλογή του εμφανίζεται μια λίστα με τα μέλη της επιλεγμένης κοινότητας

4.2.2.4 Προβολή Κοινοτήτων στις οποίες ανήκει ο χρήστης – Show My Membership

Προϋποθέσεις :

- Ο χρήστης έχει ήδη κάνει sign up στο σύστημα
- Ο χρήστης έχει εισάγει σωστά το username και password του στην αρχική ιστοσελίδα.

Περιγραφή : Ο χρήστης επιλέγει να δει τις κοινότητες στις οποίες ανήκει. Ο χρήστης πλοηγείται σε σελίδα όπου εμφανίζεται λίστα με τις κοινότητες στις οποίες ανήκει.

4.2.3 Αγαθό

Η οντότητα αγαθό μοντελοποιείται από την κλάση Commodity του μοντέλου μας. Οι βασικές λειτουργίες που έχουν ως πυρήνα τα αγαθά του συστήματος είναι οι παρακάτω:

4.2.3.1 Δημοσίευση Αγαθού - Post a commodity

Προϋποθέσεις :

- Ο χρήστης έχει εισαχθεί στο σύστημα με sign in
- Ο χρήστης έχει επιλέξει να δημοσιεύσει ένα αγαθό που κατέχει

Περιγραφή : Ο χρήστης θέλει να δημοσιεύσει ένα αγαθό το οποίο κατέχει. Το αγαθό αυτό τίθεται προς συναλλαγή.

Φόρμα Δημοσίευσης Αγαθού

Ο χρήστης εισέρχεται σε κατάλληλη ιστοσελίδα που αποτελεί την φόρμα δημοσίευσης αγαθού. Σε αυτή την φόρμα καλείται να συμπληρώσει τα παρακάτω πεδία:

- Όνομα αγαθού
- Τύπος (Αντικείμενο ή υπηρεσία)
- Κατηγορία
- Υποκατηγορία
- Περιγραφή (ενδεχομένως και φωτογραφία)
- Επιθυμητός Τρόπος Συναλλαγής
- Επιθυμητός Τρόπος Διάθεσης Αγαθού(Ζητείται/Προσφέρεται)

Αφού συμπληρώσει τα πεδία υποβάλλει την φόρμα. Αν είναι πλήρης και σωστή η καταχώρηση περνάει στην Β.Δ. Ο χρήστης πλοηγείται στην κεντρική σελίδα του συστήματος και ενημερώνεται με μήνυμα για την επιτυχή εισαγωγή αγαθού στο σύστημά μας,

Στην περίπτωση κατά την οποία η φόρμα δημοσίευσης αγαθού έχει ελλιπή στοιχεία ή μη έγκυρα στοιχεία, δεν έχουμε καταχώρηση στην βάση δεδομένων και στον χρήστη παρουσιάζεται ξανά η φόρμα με κατάλληλο μήνυμα λάθους.

4.2.3.2 Αναζήτηση Αγαθού – Search a commodity

Προϋποθέσεις **:** Ο χρήστης έχει επιλέξει να αναζητήσει αγαθό, μέσω κατάλληλου GUI item του interface του συστήματός μας.

Περιγραφή : Ο χρήστης αναζητεί ένα αγαθό στο σύστημά μας. Την λειτουργία αυτή μπορεί να την επιτελέσει είτε αν βρίσκεται στην πρώτη σελίδα του συστήματός μας είτε αν βρίσκεται στην κεντρική σελίδα του συστήματος.

Φόρμα Αναζήτησης Αγαθού

Ο χρήστης θα έχει τη δυνατότητα να κάνει προχωρημένη αναζήτηση για ένα αγαθό. Αφού κάνει αυτή την επιλογή, παρουσιάζεται μία φόρμα που πρέπει να συμπληρώσει με στοιχεία που περιγράφουν το αγαθό. Η φόρμα περιλαμβάνει :

- Εισαγωγή λέξης κλειδί για το αγαθό
- Προσδιορισμό της κατηγορίας που ο χρήστης προσδοκεί ότι θα βρει το αγαθό
- Προσδιορισμό είδους της συναλλαγής που επιθυμεί να επιτελέσει ο χρήστης επί αυτού του αγαθού (π.χ. αν το αγαθό διατίθεται για ανταλλαγή, αν το αγαθό χαρίζεται κλπ)
- Προσδιορισμό του τρόπου με τον οποίο διατίθεται το αγαθό (π.χ. αν κάποιος το ζητάει ή αν κάποιος το προσφέρει).

Ο χρήστης αφού συμπληρώσει την φόρμα την υποβάλλει. Η φόρμα απαιτείται να έχει συμπληρωμένα τα πεδία που είναι υποχρεωτικά για την αναζήτηση. Σε περίπτωση που ο χρήστης δεν έχει συμπληρώσει όλα τα υποχρεωτικά πεδία, η φόρμα εμφανίζεται ξανά με κατάλληλο μήνυμα για την συμπλήρωση όλων των υποχρεωτικών πεδίων.

Αναζήτηση

- Η αναζήτηση γίνεται στην βάση δεδομένων σύμφωνα με την κατηγορία που έχει προσδιορίσει ο χρήστης.
- Με βάση αυτή την κατηγορία, προκύπτουν συγκεκριμένες εγγραφές από τις οποίες γίνεται σύγκριση του προσδιοριστικού ονόματος του αγαθού με την λέξη κλειδί.
- Έπειτα ακολουθεί η επιλογή εγγραφών με βάση άλλα πεδία που έχει προσδιορίσει ο χρήστης στην φόρμα εγγραφής του.
- Με αυτό τον τρόπο προκύπτει μία σειρά εγγραφών που αφορούν αγαθά που αντιστοιχούν στο query αναζήτησης που εκτελέσαμε

Παρουσίαση αποτελεσμάτων

Μετά από την υποβολή της φόρμας αναζήτησης, παρουσιάζεται στον χρήστη μία νέα σελίδα με τα αποτελέσματα της αναζήτησης. Η σελίδα αυτή περιέχει ένα πίνακα αγαθών που βρέθηκαν ο οποίος παρουσιάζει τα αγαθά αυτά και επιπλέον πληροφορία (π.χ. ποιος χρήστης τα παρέχει και ποια η υπόληψή του στην συγκεκριμένη κατηγορία, με ποιον τρόπο τα παρέχει κλπ)

4.2.3.3 Αναζήτηση Χρήστη/Κοινότητας – Search user/community

Προϋποθέσεις : Ο χρήστης έχει εισαχθεί στο σύστημα με sign in

Περιγραφή : Ο χρήστης επιθυμεί να δει το προφίλ μίας οντότητας. Εισάγει το όνομα του χρήστη ή της κοινότητας. Καθορίζει επίσης τον τύπο της οντότητας (αν πρόκειται για χρήστη ή κοινότητα). Έπειτα εκτελεί την αναζήτηση.

Αναζήτηση

Η αναζήτηση εκτελείται σε κατάλληλο πίνακα στην βάση μας. Το όνομα της οντότητας που χρησιμοποιήσαμε θα μας δώσει ενδεχομένως μία πληθώρα οντοτήτων που θα έχουν όνομα που μοιάζει με το όνομα που δόθηκε προς αναζήτηση.

Παρουσίαση Αποτελεσμάτων

Τα αποτελέσματα που προκύπτουν παρουσιάζονται στον χρήστη σε πίνακα σε κατάλληλη σελίδα. Με κατάλληλα hyperlinks μπορεί να δει τα προφίλ των οντοτήτων.

4.2.4 Συναλλαγή

Προϋποθέσεις :

- Η οντότητα Α έχει εισαχθεί στο σύστημα(π.χ. έχει κάνει sign in στο σύστημα). Με την έννοια οντότητα Α εννοούμε χρήστη ή κοινότητα (η κοινότητα εκπροσωπείται από κάποιο moderator της)
- Η οντότητα Α έχει αναζητήσει αγαθό Ι και έχει βρει αγαθό που επιθυμεί να αποκτήσει
- Το αγαθό Ι που επιθυμεί ο Α να αποκτήσει ανήκει στην οντότητα Β

Περιγραφή : Η οντότητα Α μέσα από μία λίστα με αγαθά , η οποία έχει προκύψει είτε μέσω αναζήτησης, είτε από λίστες που ενημερώνουν τους χρήστες για αγαθά (π.χ. η λίστα της πρώτης σελίδας που αφορά τα αγαθά που μπήκαν πιο πρόσφατα στο σύστημα), επιλέγει ένα αγαθό που θέλει να αποκτήσει δίνοντας ως αντάλλαγμα αγαθό Ι' που κατέχει.

Φόρμα Αίτησης Συναλλαγής

Η οντότητα Α επιλέγει το αγαθό το αγαθό που θέλει να αποκτήσει και με κατάλληλο hyperlink μεταβαίνει σε επόμενη κατάλληλα διαμορφωμένη σελίδα που αποτελεί την φόρμα αίτησης συναλλαγής.

Στην φόρμα αυτή ένα από τα βασικά στοιχεία που θα συμπληρώσει ο χρήστης αφορά τον τρόπο συναλλαγής (στην περίπτωσή μας ανταλλαγή) και το αντάλλαγμα που θα δώσει για την απόκτηση του αγαθού.

Εδώ υπάρχουν δύο δυνατότητες :

1. επιλογή αγαθού μέσα από λίστα με αγαθά τα οποία έχει ήδη εισάγει ο χρήστης στην βάση ως αγαθά που κατέχει και η οποία παρουσιάζεται στον χρήστη ως drop down list. Εδώ ο χρήστης μπορεί να καθορίσει σε κατάλληλο πεδίο την κατηγορία αγαθού και η λίστα θα διαμορφωθεί ανάλογά.
2. εισαγωγή ενός νέου αγαθού στην βάση και επιλογή αυτού του αγαθού για ανταλλαγή .Για εισαγωγή νέου αγαθού μεταβαίνει σε κατάλληλη σελίδα και αφού εισάγει το αγαθό στην βάση, επιστρέφει στην σελίδα φόρμας συναλλαγής.

Αποστολή Φόρμας Αίτησης Συναλλαγής

Αφού ο χρηστής συμπληρώσει την φόρμα συναλλαγής:

- υποβάλλει την φόρμα.
- η βάση δεδομένων ενημερώνεται για την συναλλαγή του εκκρεμεί στον αντίστοιχο πίνακα
- το άλλο μέλος της συναλλαγής ενημερώνεται μέσω του προφίλ του ή/και με mail για την συναλλαγή αυτή

Επιβεβαίωση/Απόρριψη Συναλλαγής

- Η οντότητα Β επικυρώνει ή απορρίπτει την συναλλαγή και ο χρήστης Α ενημερώνεται κατάλληλα ή μέσω του προφίλ του ή/και μέσω mail.
- Η Β.Δ. ενημερώνεται κατάλληλα. Σημειώνουμε, ωστόσο, ότι η συναλλαγή ακόμα βρίσκεται στον πίνακα με τις συναλλαγές που εκκρεμούν.

Εκτέλεση Συναλλαγής

Αν το άλλο μέλος της συναλλαγής επικυρώσει την συναλλαγή, τα δύο μέλη της συναλλαγής επικοινωνούν και εκτελούν την συναλλαγή. Αναπόσπαστο κομμάτι της εκτέλεσης της συναλλαγής είναι η ανταλλαγή βαθμών. Άρα, ο κάθε χρήστης κάνει τις εξής ενέργειες :

- Αναζητά την συναλλαγή που εκτέλεσε μέσα στο πλήθος των συναλλαγών που εκκρεμούν για αυτόν.
- Επιλέγει την συναλλαγή και δίνει βαθμό υπόληψης στο άλλο μέλος της συναλλαγής
- Οι πίνακες των συναλλαγών ενημερώνονται κατάλληλα στην βάση μας.
- Επίσης κατάλληλα ενημερώνονται οι βαθμοί υπόληψης που επηρεάζονται στην βάση μας

Για να σιγουρέψουμε ότι οι συναλλαγές που εκτελούνται συνοδεύονται από ανταλλαγή βαθμών, μπορούμε να απαγορέψουμε να εκκρεμούν περισσότερες από Ν συναλλαγές ανά χρήστη, παρότι το άλλο μέλος της συναλλαγής έχει δώσει βαθμό σε αυτές τις ν συναλλαγές. Με αυτό τον τρόπο ο χρήστης σίγουρα θα υποχρεωθεί σε κάποια στιγμή να δώσει βαθμό.

Απόρριψη Συναλλαγής

- Ο χρήστης Α ενημερώνεται κατάλληλα μέσω mail ή/και του προφίλ του.
- Η εγγραφή που αφορούσε την συναλλαγή διαγράφεται από τον κατάλληλο πίνακα που αφορά τις ανταλλαγές που εκκρεμούν, στην βάση δεδομένων

4.2.5 Υπόληψη

4.2.5.1 Βαθμολόγηση συναλλαγής – Rate transaction

Προϋποθέσεις :

- Οι οντότητες που λαμβάνουν μέρος σε μία συναλλαγή έχουν εισαχθεί στο σύστημα. Με τον όρο οντότητα εννοούμε χρήστη ή κοινότητα, η οποία εκπροσωπείται από τον administrator της.
- Η συναλλαγή έχει ολοκληρωθεί και από τα δύο μέλη και είναι στο στάδιο της βαθμολόγησης.

Περιγραφή : Ανάλογα με το είδος της συναλλαγής, η μία ή και οι δυο οντότητες βαθμολογούν την συναλλαγή με βάση την ποιότητα της και την αξιοπιστία που επέδειξε το άλλος μέρος.

- ***Ο χρήστης καλείται να επιλέξει την συγκεκριμένη συναλλαγή.***

  Μέσα από μια λίστα με τις τρέχουσες συναλλαγές, ο χρήστης επιλέγει την συναλλαγή που θέλει να βαθμολογήσει.

- ***Ο χρήστης ή οι χρήστες βαθμολογεί/ούν την συναλλαγή***

  Μέσα από μια φόρμα όπου παρουσιάζονται οι βαθμοί στην κλίμακα του δέκα, ο κάθε χρήστης επιλέγει την κατάλληλη καταυτόν βαθμολογία. Η κλίμακα παρουσιάζεται γραφικά.

- ***Ενημέρωση Βάσης Δεδομένων***

  Η Βάση Δεδομένων του συστήματος ενημερώνεται κατάλληλα.

4.2.5.1 Προβολή βαθμολογίας – Show rating

Προϋποθέσεις :

- Ο χρήστης Α έχει συμμετάσχει σε τουλάχιστον μία συναλλαγή.
- Μία συναλλαγή του χρήστη Α έχει ολοκληρωθεί επιτυχώς έχοντας βαθμολογηθεί.

Περιγραφή **:** Οποιοσδήποτε επισκέπτης της εφαρμογής μπορεί να δει το μέτρο της αξιοπιστίας κάποιου μέλους χωρίς να απαιτείται η εγγραφή του. Η βαθμολογία μπορεί να εμφανίζεται σε διάφορα σημεία του συστήματός μας.

- ***Ο χρήστης/επισκέπτης επιλέγει να δει το προφίλ του χρήστη Α***.

  Μέσα από το παράθυρο της εφαρμογής ο επισκέπτης διαλέγει ποιον χρήστη θέλει και πατάει πάνω στο ψευδώνυμο του χρήστη.

- ***Ο επισκέπτης βλέπει το προφίλ του χρήστη Α και τη βαθμολογία του.***

  Εμφανίζεται στον επισκέπτη το προφίλ του χρήστη Α στο οποίο εμφανίζεται αναλυτικά η βαθμολογία του(π.χ. η ατομική του υπόληψη, η υπόληψη στις κοινότητες που ανήκει).

4.3 Βάση Δεδομένων

Για να κάνουμε κατανοητές τις λειτουργίες και τις ανάγκες που εξυπηρετεί το σχήμα της Βάσης μας, θα παραθέσουμε και θα αναλύσουμε κάθε πίνακα του σχήματός μας ξεχωριστά. Για να το κάνουμε αυτό θα δώσουμε τις εντολές δημιουργίας των πινάκων σε sql

- Πίνακας **Entity**

```
CREATE TABLE  `entities` (
        `id` int(11) NOT NULL auto_increment,
        `actor_id` int(11) default NULL,
        `actor_type` varchar(255) default NULL,
        PRIMARY KEY  (`id`)
)
```

Ο πίνακας *entity* δημιουργεί την πρωταρχική μας οντότητα πάνω στην οποία θα δομηθούν οι άλλες δύο, του Χρήστη και της Κοινότητας. Αυτό μας διευκολύνει στη σύνδεση των οντοτήτων μας με τις υπόλοιπες κλάσεις της εφαρμογής. Υλοποιούμε δηλαδή μια σχέση γενίκευσης μεταξύ των οντοτήτων ώστε να υλοποιηθεί ο απαιτούμενος πολυμορφισμός σε επίπεδο κλάσεων. Την σχέση πολυμορφισμού θα αναλύσουμε σε παρακάτω εδάφιο. Έτσι στις

περιπτώσεις που απαιτείται η αναφορά χωρίς διάκριση και στις δύο οντότητες, χρησιμοποιούμε την entity.

Το πεδίο *actor_id* αναφέρεται στο id του αντίστοιχου πίνακα των χρηστών ή των κοινοτήτων ενώ το πεδίο *actor_type* καθορίζει τι τύπου είναι η οντότητα και άρα σε ποιον από τους δυο πίνακες αναφέρεται.

- Πίνακες **Users - Communities**

```
CREATE TABLE  `users` (
        `id` int(11) NOT NULL auto_increment,
        `name` varchar(255) NOT NULL,
        `hashed_password` varchar(255) NOT NULL,
        `salt` varchar(255) NOT NULL,
        `firstname` varchar(255) NOT NULL,
        `surname` varchar(255) NOT NULL,
        `email` varchar(255) NOT NULL,
        `address` varchar(255) default NULL,
        `city` varchar(255) default NULL,
        `insert_date` date NOT NULL,
        `age` int(11) default NULL,
        `phone_number` varchar(255) default NULL,
        `working_status` varchar(255) default NULL,
        PRIMARY KEY  (`id`)
)

CREATE TABLE  `communities` (
        `id` int(11) NOT NULL auto_increment,
        `name` varchar(255) NOT NULL,
        `theme_of_interest` varchar(255) NOT NULL,
        `description` text,
        `user_id` int(11) NOT NULL,
        `date` varchar(255) NOT NULL,
```

```
    PRIMARY KEY (`id`),
    KEY `fk_community_users` (`user_id`),
    CONSTRAINT `fk_community_users` FOREIGN KEY (`user_id`) REFERENCES `users` (`id`)
)
```

Οι πίνακες *users* και *communities* δημιουργούν και περιέχουν τα χαρακτηριστικά των βασικών μας οντοτήτων, των χρηστών και των κοινοτήτων. Όπως έχουμε προαναφέρει κάθε εγγραφή στον πίνακα entity αντιστοιχεί σε μία εγγραφή ενός από τους δυο παραπάνω πίνακες επιτρέποντας μας εξειδικεύσουμε τις χαρακτηριστικές ιδιότητες κάθε πίνακα ξεχωριστά.

Έτσι βλέπουμε ότι τα πεδία *name, first_name, surname, email, address, city, age, phone_number* και *working_status* αποτελούν τα άμεσα χαρακτηριστικά του χρήστη έτσι όπως τα έχει υποβάλλει στην εφαρμογή. Τα πεδία *hashed_password* και *salt* αποθηκεύουν τον κωδικό του χρήστη μέσα από έναν αλγόριθμο, τον οποίο αναπτύσσουμε στο επόμενο κεφάλαιο, ενώ στο πεδίο *insert_date* αποθηκεύεται αυτόματα από το σύστημά μας η ημερομηνία της εγγραφής του χρήστη.

Αντίστοιχα, για τις κοινότητες της εφαρμογής, στα πεδία *name, theme_of_interest* και *description* αποθηκεύονται τα στοιχειά της κοινότητας έτσι όπως τα υποβάλλει ο δημιουργός της. Το πεδίο *user_id* χρησιμεύει στην αντιστοίχιση της κάθε κοινότητας με την εγγραφή του χρήστη που την δημιούργησε. Αυτό ενισχύεται και από τον τελευταίο περιορισμό που έχουμε εισάγει στον κώδικα.

- Πίνακας **Memberships**

```
CREATE TABLE `memberships` (
    `id` int(11) NOT NULL auto_increment,
    `user_id` int(11) NOT NULL,
    `community_id` int(11) NOT NULL,
    `join_date` datetime NOT NULL,
    `membership_status` char(1) default NULL,
    PRIMARY KEY (`id`),
    KEY `fk_membership_users` (`user_id`),
    KEY `fk_membership_communities` (`community_id`),
```

```
    CONSTRAINT `fk_membership_communities` FOREIGN KEY (`community_id`)
REFERENCES `communities` (`id`),
    CONSTRAINT `fk_membership_users` FOREIGN KEY (`user_id`)
REFERENCES `users` (`id`)
)
```

Στον πίνακα *memberships* αποθηκεύεται η πληροφορία για τα μέλη κάθε κοινότητας μαζί με κάποια επιπλέον στοιχεία που ταυτοποιούν την κάθε εγγραφή. Έτσι βλέπουμε πως τα πεδία *user_id* και *community_id* χρησιμεύουν για την αντιστοίχιση ενός χρήστη μέλος μιας κοινότητας με την κοινότητα αυτή. Ακόμα στα πεδία *join_date* και *memberships_status* αποθηκεύουμε την ημερομηνία εγγραφής των χρηστών στην κοινότητα καθώς και τα δικαιώματα που έχει ο χρήστης σε αυτή, αν είναι δηλαδή απλό μέλος ή έχει διαχειριστικά δικαιώματα. Οι δύο περιορισμοί δεν κάνουν τίποτε άλλο από το να εξασφαλίζουν την σωστή αντιστοίχιση.

- Πίνακες **Commodities - Pictures**

```
CREATE TABLE `commodities` (
        `id` int(11) NOT NULL auto_increment,
        `name` varchar(255) NOT NULL,
        `description` text,
        `entity_id` int(11) NOT NULL,
        `date` datetime NOT NULL,
        PRIMARY KEY (`id`),
        KEY `fk_commodity_entities` (`entity_id`),
        CONSTRAINT `fk_commodity_entities` FOREIGN KEY (`entity_id`)
        REFERENCES `entities` (`id`)
)

CREATE TABLE `pictures` (
        `id` int(11) NOT NULL auto_increment,
        `name` varchar(255) default NULL,
        `commodity_id` int(11) default NULL,
```

```
    `content_type` varchar(255) default NULL,
    `data` mediumblob,
    PRIMARY KEY (`id`)
)
```

Ο πίνακας όπου αποθηκεύονται οι πληροφορίες για τα αγαθά που υπάρχουν γενικά στον σύστημά μας είναι ο *commodities*. Σε αυτόν δεν αποθηκεύουμε καμία πληροφορία για την κατάσταση των αγαθών παρά μόνο τα στοιχεία που έχει εισάγει ο χρήστης και χαρακτηρίζουν το αγαθό. Έτσι, τα πεδία *name, description* και *date* περιέχουν τιε πληροφορίες που έχει εισάγει ο χρήστης για το συγκεκριμένο αντικείμενο καθώς και την ημερομηνία εισαγωγής του στο σύστημα. Με το πεδίο *entity_id* αντιστοιχείται ο ιδιοκτήτης του αντικειμένου που μπορεί να είναι είτε χρήστης είτε κοινότητα.

Ακόμα, κάθε αγαθό μπορεί να έχει και κάποιες φωτογραφίες που το χαρακτηρίζουν και οι οποίες αποθηκεύονται στον πίνακα *pictures*. Τα πεδία *name, content_type* και *data* περιέχουν το όνομα της φωτογραφίας, τον τύπο της (gif, jpeg, κτλ) και τα δεδομένα της αντίστοιχα. Τέλος το πεδίο *commodity_id* αντιστοιχεί την κάθε φωτογραφία με ένα αντικείμενο του πίνακα των *commodities*. Επιλέξαμε να βάλουμε το πεδίο εδώ αφού κάθε αγαθό μπορεί να έχει πάνω από μια φωτογραφία.

- Πίνακες **Demands - Offers**

```
CREATE TABLE `demands` (
    `id` int(11) NOT NULL auto_increment,
    `commodity_id` int(11) NOT NULL,
    `type_of_transaction` varchar(255) default NULL,
    PRIMARY KEY (`id`),
    KEY `fk_demand_commodities` (`commodity_id`),
    CONSTRAINT `fk_demand_commodities` FOREIGN KEY (`commodity_id`) REFERENCES `commodities` (`id`)
)

CREATE TABLE `offers` (
    `id` int(11) NOT NULL auto_increment,
```

```
    `commodity_id` int(11) NOT NULL,
    `type_of_transaction` varchar(255) default NULL,
    PRIMARY KEY  (`id`),
    KEY `fk_offer_commodities` (`commodity_id`),
    CONSTRAINT `fk_offer_commodities` FOREIGN KEY (`commodity_id`) REFERENCES `commodities` (`id`)
)
```

Οι πίνακες *demands* και *offers* αποθηκεύουν την πληροφορία της διάθεσης και της ζήτησης των αγαθών κάθε στιγμή. Στον πίνακα *offers* δηλαδή υπάρχουν τα αγαθά που διατίθενται για συναλλαγή ενώ στον πίνακα *demands* εκείνα που ζητούνται. Τα πεδία *commodity_id* και στους δύο πίνακες αντιστοιχούν στο αγαθό που διατίθεται/ζητείται ενώ το πεδίο *type_of_transaction* δείχνει τον τύπο της συναλλαγής που χαρακτηρίζει το αγαθό. Το τελευταίο πεδίο ορίζεται από τον χρήστη/κοινότητα και μπορεί να πάρει διάφορες τιμές, από απλή ανταλλαγή μέχρι όλα τα είδη.

- Πίνακας **Xaxtions**

```
CREATE TABLE  `xactions` (
    `id` int(11) NOT NULL auto_increment,
    `commodity1_id` int(11) default NULL,
    `commodity2_id` int(11) default NULL,
    `entity1_id` int(11) default NULL,
    `entity2_id` int(11) default NULL,
    `type_of_transaction` varchar(255) default NULL,
    `date` datetime default NULL,
    `state` char(1) default NULL,
    PRIMARY KEY  (`id`)
)
```

Στον πίνακα *Xactions* φυλάσσονται οι πληροφορίες για κάθε ενεργή συναλλαγή που βρίσκεται στο σύστημά μας. Με τον όρο ενεργή εννοούμε κάθε συναλλαγή που βρίσκεται σε εξέλιξη, σε οποιαδήποτε φάση. Τα πεδία *commodity1_id, commodity2_id, entity1_id* και

entity2_id επιτρέπουν την συσχέτιση των αγαθών που παίρνουν μέρος σε κάθε συναλλαγή ανάλογα με τον τύπο της, καθώς και των οντοτήτων που λαμβάνουν μέρος σε αυτή είτε ως κάτοχοι των αγαθών είτε ως αιτούντες αυτών. Όπως στους πίνακες *demands* και *offers* το πεδίο *type_of_transaction* δηλώνει τον τύπο της συναλλαγής μου μπορεί να είναι ένας από τους τρείς βασικούς. Το πεδίο *date* αποθηκεύει την πληροφορία της ώρας και μέρας που δημιουργήθηκε η συναλλαγή έτσι ώστε ο χρήστης να μπορεί να ταξινομεί και να επιλέγει και με βάση την χρονολογική σειρά.

Τέλος όπως έχουμε πει, κάθε συναλλαγή μπορεί να βρίσκεται σε διαφορετική κατάσταση ανάλογα με το στάδιο στο οποίο βρίσκεται. Έτσι το πεδίο *state* αποθηκεύει την κατάσταση στην οποία είναι κάθε στιγμή η συναλλαγή.

- Πίνακες **Vactivities – Reputations**

```
CREATE TABLE  `reputations` (
        `id` int(11) NOT NULL auto_increment,
        `entity_id` int(11) default NULL,
        `grade` decimal(6,4) default NULL,
        `transNo` int(11) default NULL,
        `final_grade` decimal(6,4) default NULL,
        PRIMARY KEY  (`id`),
        KEY `fk_reputation_entities` (`entity_id`),
        CONSTRAINT `fk_reputation_entities` FOREIGN KEY (`entity_id`) REFERENCES `entities` (`id`)
)
CREATE TABLE  `vactivities` (
        `id` int(11) NOT NULL auto_increment,
        `community_id` int(11) default NULL,
        `avg` decimal(6,4) default NULL,
        `transNo` int(11) default NULL,
        `activeUsers` int(11) default NULL,
        PRIMARY KEY  (`id`),
        KEY `fk_vactivity_communities` (`community_id`),
```

```
    CONSTRAINT `fk_vactivity_communities` FOREIGN KEY (`community_id`)
REFERENCES `communities` (`id`)
)
```

Στους πίνακες *reputations* και *vactivities* αποθηκεύουμε όλη την απαιτούμενη πληροφορία για το σύστημα της υπόληψης. Ο πρώτος περιέχει τις εγγραφές με τους βαθμούς της υπόληψης για κάθε οντότητα καθώς και όλες τις άλλες απαιτούμενες πληροφορίες με τις από τις οποίες προκύπτει ο τελικός βαθμός. Έτσι στα πεδία *grade, transNo* και *final_grade* αποθηκεύουμε τους βαθμούς καθώς και των αριθμών των συναλλαγών κάθε οντότητας ενώ το πεδίο *entity_id* αντιστοιχεί κάθε εγγραφή με μία οντότητα.

Τον δεύτερο πίνακα τον χρησιμοποιούμε για να μεταχειριζόμαστε τις επιπλέον πληροφορίες που χρειάζονται για την υπόληψη των κοινοτήτων. Έτσι τα πεδία *transNo* και *avg* μας βοηθάνε στον υπολογισμό της υπόληψη των κοινοτήτων καθώς και το πεδίο *activeUsers* που αποθηκεύει των αριθμό των μελών κάθε κοινότητας που έχουν κάνει τουλάχιστον μία συναλλαγή. Τέλος το πεδίο *community_id,* όπως ήταν φυσικό, αντιστοιχεί κάθε εγγραφή του πίνακα με μία κοινότητα.

- Πίνακας **Sessions**

```
CREATE TABLE `sessions` (
    `id` int(11) NOT NULL auto_increment,
    `session_id` varchar(255) default NULL,
    `data` text,
    `updated_at` datetime default NULL,
    PRIMARY KEY (`id`),
    KEY `index_sessions_on_session_id` (`session_id`),
    KEY `index_sessions_on_updated_at` (`updated_at`)
)
```

Τα sessions είναι ένας τρόπος να αποθηκεύσουμε προσωρινά, ανάμεσα στις HTTP αιτήσεις, την κατάσταση του χρήστη. Μας επιτρέπουν να αποθηκεύσουμε αντικείμενα στην μεριά του server και με αυτό τον τρόπο επιτυγχάνουμε να διατηρήσουμε την κατάσταση του χρήστη, δουλεύοντας πάνω σε HTTP, το οποίο δεν διαθέτει μηχανισμό για την διατήρηση της κατάστασης του χρήστη ανάμεσα στις HTTP αιτήσεις. Συγκεκριμένα, η δημιουργία ενός session στην μεριά του server συνοδεύεται από την γένεση ενός τυχαίου μοναδικού αριθμού,

του sessionID το οποίο και στέλνει ο server στον browser. Η πληροφορία αυτού του sessionID αποθηκεύεται στην cache του browser και αποστέλλεται στον server μέσα σε κάθε αίτηση HTTP που στέλνει ο client στον server. Με αυτό τον τρόπο, η εφαρμογή μας στην μεριά του server επιτυγχάνει την αντιστοίχηση της πληροφορίας που κρατάει με τον client τον οποίο αφορά. Ο ρόλος των sessions στην εφαρμογή μας θα αναλυθεί στο επόμενο κεφάλαιο.

- Πίνακες **Gifts – Sharings – Swaps**

```
CREATE TABLE  `gifts` (
        `id` int(11) NOT NULL auto_increment,
        `commodity_id` int(11) default NULL,
        `entity1_id` int(11) default NULL,
        `entity2_id` int(11) default NULL,
        `grade` decimal(6,4) default NULL,
        `date` datetime default NULL,
        PRIMARY KEY  (`id`)
)

CREATE TABLE  `sharings` (
        `id` int(11) NOT NULL auto_increment,
        `commodity_id` int(11) default NULL,
        `entity1_id` int(11) default NULL,
        `entity2_id` int(11) default NULL,
        `grade` decimal(6,4) default NULL,
        `date` datetime default NULL,
        PRIMARY KEY  (`id`)
)

CREATE TABLE  `swaps` (
        `id` int(11) NOT NULL auto_increment,
```

```
    `commodity1_id` int(11) default NULL,
    `commodity2_id` int(11) default NULL,
    `entity1_id` int(11) default NULL,
    `entity2_id` int(11) default NULL,
    `grade1` decimal(6,4) default NULL,
    `grade2` decimal(6,4) default NULL,
    `date` datetime default NULL,
    PRIMARY KEY (`id`)
)
```

Οι τελευταίοι πίνακες της βάσης μας, *gifts, sharings* και *swaps*, αποτελούν το μέσο για την καταγραφή της ιστορίας του συστήματός μας αφού σε αυτούς γράφουμε κάθε συναλλαγή όταν αυτή έχει πλέον ολοκληρωθεί. Με τον τρόπο αυτό μπορούμε να έχουμε μια πλήρη εικόνα για την πορεία της εφαρμογής μας αλλά και διασφάλιση της ακεραιότητάς της.

Βλέπουμε λοιπόν πως τα στοιχεία που αποθηκεύουμε σε αυτούς τους τρεις πίνακες είναι παρόμοια με αυτά που κρατάμε για τον πίνακα των ενεργών συναλλαγών με το πρόσθετο πεδίο *grade(grade1,grade2)* στο οποίο φυλάμε το βαθμό που καταχώρησε κάθε χρήστης της συναλλαγής. Έτσι έχουμε ακόμα τα πεδία *commodity(commodity1, commodity2)* και *entity(entity1, entity2)* με τα οποία αντιστοιχούμε κάθε αγαθό και οντότητα που πήρε μέρος στη συναλλαγή καθώς και το πεδίο *date* που είναι η χρονική στιγμή που ολοκληρώθηκε η συναλλαγή.

5

Υλοποίηση

Για την υλοποίηση του συστήματος της VCommunity κάναμε χρήση του Ruby on Rails framework. Η Ruby είναι μία αντικειμενοστραφής γλώσσα που με την προσθήκη των Rails και τις αυξημένες δυνατότητες που αυτά παρέχουν, αποτελεί ένα ισχυρό εργαλείο για structured web programming. Τα Rails προσφέρουν ένα σκελετό για την ανάπτυξη της εφαρμογής μας όπου κάθε στοιχείο του κώδικά μας έχει συγκεκριμένη θέση και ρόλο. Η αλληλεπίδραση των υπομονάδων της εφαρμογής που δημιουργούμε ενορχηστρώνεται από τα Rails με συγκεκριμένο τρόπο.

Η Ruby on Rails υποδεικνύει την ανάπτυξη εφαρμογών υπό την αρχιτεκτονική τεχνοτροπία MVC (Model – View – Controller). Τα Rails έχουν τρεις βασικές υπομονάδες, το Active Record, το Action View και το Action Controller, που διαχειρίζονται αντίστοιχα τις διακριτές μονάδες της εφαρμογές μας (Model – View – Controller) . Οι τρεις αυτές υπομονάδες αποτελούν το ενδιάμεσο ανάμεσα στην εφαρμογή μας και τη Ruby και επιτρέπουν στο framework να ενεργοποιεί αυτόβουλα πολλαπλές λειτουργίες που διαφορετικά θα έπρεπε να εκτελέσει λεπτομερώς ο προγραμματιστής. Σημειώνουμε ότι οι Controllers και οι Views παρουσιάζουν υψηλή συνάφεια, καθώς ο ρόλος του Controller είναι να παρέχει δεδομένα στις Views και να λαμβάνει γεγονότα από τις ιστοσελίδες που διαμορφώνουν οι Views. Λόγω της ισχυρής αλληλεπίδρασής τους, η υποστήριξη αυτών των υπομονάδων ομαδοποιείται από την υπομονάδα των Rails που καλείται Action Pack. Ωστόσο, η ύπαρξη του Action Pack δεν σηματοδοτεί την ενοποίηση του κώδικα του Controller και της View. Αντίθετα, τα Rails υποστηρίζουν τον διαχωρισμό που χρειάζεται για την διαμόρφωση web εφαρμογών με

ξεκάθαρη διάκριση του κώδικα που αφορά την διαχείριση των δεδομένων και την λογική παρουσίασης αυτών.

Το Active Record αποτελεί object – relational mapping (ORM) επίπεδο του framework. Παρέχει σύνδεση με την βάση δεδομένων μας, κατασκευάζει το σχήμα της βάσης δεδομένων και διαχειρίζεται τα δεδομένα μας. Ακολουθεί το κλασσικό ORM μοντέλο σύμφωνα με το οποίο οι πίνακες της βάσης δεδομένων αντιστοιχούν σε κλάσεις της εφαρμογής, οι εγγραφές των πινάκων αντιστοιχούν σε αντικείμενα και οι στήλες των πινάκων σε ιδιότητες των αντικειμένων. Η διαφορά που παρουσιάζει το Active Record έναντι των περισσοτέρων ORM βιβλιοθηκών εντοπίζεται στην παραμετροποίηση του(configuration). Χρησιμοποιεί σύνολο συμβάσεων(π.χ. συμβάσεις ονομάτων, προσθήκη default πρωτεύοντος κλειδιού κτλ) και με αυτό τον τρόπο ελαχιστοποιεί το μέγεθος της παραμετροποίησης που πρέπει να εκτελέσει ο προγραμματιστής. Επιπλέον, διαχειρίζεται την βάση δεδομένων μας, επιτρέποντας την εκτέλεση sql ερωτημάτων ή statements σε προγραμματιστικά υψηλότερο επίπεδο. Η δημιουργία των πινάκων της βάσης δεδομένων του συστήματός μας και η διαχείριση τους υλοποιείται από τις κλάσεις και τις μεθόδους που παρέχει το Active Record.

Στα Rails, η view ευθύνεται για την δημιουργία ενός μέρους ή και ολόκληρης της σελίδας που θα παρουσιαστεί στον browser. Στην πιο απλή της μορφή, μία view είναι ένα κομμάτι από HTML κώδικα που παρουσιάζει κείμενο σε δομημένη μορφή. Συνήθως, ωστόσο, συμπεριλαμβάνει δυναμικό περιεχόμενο που δημιουργείται από μέθοδο ενός controller.

Στα Rails, δυναμικό περιεχόμενο σε view παράγεται από τριών ειδών φόρμες. Η πιο γνωστή φόρμα καλείται rhtml και εισάγει κώδικα Ruby μέσα σε HTML όψη, χρησιμοποιώντας το εργαλείο Erg(Embedded Ruby). Αυτή η προσέγγιση είναι ιδιαιτέρως ευέλικτη παρ' ότι τυπικά παραβιάζει την έννοια του MVC. Η εισαγωγή κώδικα σε μία view ενέχει τον κίνδυνο να εισάγουμε λογική που θα έπρεπε να βρίσκεται είτε στο μοντέλο είτε στον controller. Μία ακόμα φόρμα καλείται rxml και μας επιτρέπει την κατασκευή XML αρχείων χρησιμοποιώντας κώδικα Ruby. Η δομή του XML που δημιουργείται αυτόματα ακολουθεί την δομή του κώδικα σε Ruby. Τα Rails παρέχουν και rjs views που επιτρέπουν την δημιουργία JavaScript κομματιών κώδικα στον server, τα οποία θα εκτελεστούν στον browser. Αυτό το στοιχείο επιτρέπει και την δημιουργία δυναμικών Ajax interfaces.

Το τρίτο component μίας Ruby on Rails εφαρμογής είναι ο controller, ο οποίος αποτελεί το λογικό κέντρο της εφαρμογής. Ο controller συντονίζει την αλληλεπίδραση του χρήστη, της view και του μοντέλου. Επειδή τα Rails διαχειρίζονται αυτή την αλληλεπίδραση παρασκηνιακά, ο κώδικας που χρειάζεται ένας προγραμματιστής επικεντρώνεται στην υλοποίηση της λειτουργικότητας σε επίπεδο της εφαρμογής. Με αυτό τον τρόπο, ο προγραμματισμός ενός controller στα Rails είναι πολύ εύκολος. Ένας controller περιλαμβάνει μεθόδους οι οποίες ενεργοποιούνται από τις HTTP αιτήσεις που κάνουν οι χρήστες μέσω του

browser. Η διαχείριση αυτών των αιτήσεων και η ενεργοποίηση της κατάλληλης μεθόδου βασίζεται στον φορμαλισμό και τις συμβάσεις που επιβάλλει η Ruby on Rails αναφορικά με τα URL που παρέχει ο χρήστης και στα ονόματα των controller και των μεθόδων τους.

Η ισχυρή συνάφεια ανάμεσα σε controllers και views εκφράζεται κατά την δημιουργία και διαγραφή ενός controller. Συγκεκριμένα, η Action Pack υπομονάδα της Ruby on Rails μεριμνεί ώστε η κατασκευή ή διαγραφή ενός controller να συνοδεύεται από την κατασκευή ή την διαγραφή της αντίστοιχης view.

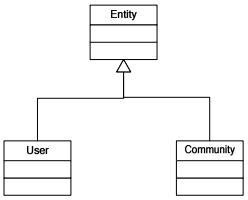
Παρακάτω παρουσιάζουμε την υλοποίηση των λειτουργικών μονάδων του συστήματός μας.

5.1 Λεπτομέρειες υλοποίησης

5.1.1 Σύσταση των οντοτήτων

Το σύστημά μας έχει ως βασικούς δράστες τους χρήστες και τις κοινότητες. Οι χρήστες και οι κοινότητες αποτελούν τις οντότητες του συστήματός μας. Οι χρήστες είναι φυσικά πρόσωπα σε αντίθεση με τις κοινότητες οι οποίες δεν έχουν φυσική υπόσταση αλλά αποτελούν ομάδες χρηστών. Οι οντότητες αυτές και η αλληλεπίδρασή τους είναι ουσιώδης για την λειτουργία του συστήματός μας. Στο εδάφιο αυτό θα αναλύσουμε το κομμάτι της υλοποίησης που αντιστοιχεί στις οντότητες του συστήματός μας και την αλληλεπίδρασή τους.

Αρχικά, παρουσιάζουμε τους πίνακες της βάσης δεδομένων μας, που υλοποιήσαμε για να κρατάμε την πληροφορία για τις οντότητες του συστήματός μας. Οι χρήστες (users) και οι κοινότητες (communities) αποτελούν ειδίκευση της οντότητας (entities). Η συσχέτιση αυτή σε ένα αντικειμενοστραφές μοντέλο εκφράζει μία σχέση κληρονομικότητας, όπου η κλάση πατέρας είναι η κλάση που αντιστοιχεί στην οντότητα (entity) και τις δύο υποκλάσεις της είναι η κλάση user για τους χρήστες και η κλάση community για τις κοινότητες.



Σε μια πρώτη προσέγγιση στην υλοποίηση του παραπάνω σχήματος κληρονομικότητας, θα είχαμε τρείς πίνακες:

- entities(INTEGER id)
- users (INTEGER id, STRING name, STRING hashed_password, STRING salt, STRING firstname, STRING surname, STRING email, STRING address, STRING city, DATE insert_date, INTEGER age, INTEGER phone_number, STRING working_status)
- communities (INTEGER id, STRING name, STRING theme_of_interest, TEXT description, INTEGER user_id, DATE date)

Ο πίνακας users θα είχε ως κύριο κλειδί το πεδίο id, το οποίο θα ήταν εξωτερικό κλειδί με αναφορά στο πεδίο id του πίνακα entities. Αντίστοιχα, ο πίνακας communities θα είχε ως κύριο κλειδί το πεδίο id, το οποίο θα ήταν εξωτερικό κλειδί με αναφορά στο πεδίο id του πίνακα entities.

Όμως, το Active Record δεν επιτρέπει σε ένα πρωτεύον κλειδί να αποτελεί συγχρόνως και εξωτερικό κλειδί. Κατά την δημιουργία ενός πίνακα της βάσης δεδομένων, το Active Record προσθέτει από μόνο του ένα πεδίο που αποτελεί το πρωτεύον κλειδί. Το πεδίο αυτό καλείται πάντα id και είναι ένας ακέραιος αριθμός που εκτός από την ταυτοποίηση της εγγραφής του πίνακα δεν έχει καμία άλλη σημειολογία. Το γεγονός αυτό μας οδήγησε στην αναζήτηση μίας διαφορετικής προσέγγισης για την υλοποίηση της κληρονομικότητας.

Το πρόβλημα που αντιμετωπίζουμε στην ουσία αφορά την απεικόνιση ενός αντικειμενοστραφούς μοντέλου σε ένα σχεσιακό μοντέλο, καθώς στο σχεσιακό μοντέλο των βάσεων δεδομένων δεν υπάρχει η έννοια της κληρονομικότητας. Τα Rails παρέχουν δύο μηχανισμούς που επιτελούν την αντιστοίχηση αντικειμενοστραφούς μοντέλου σε σχεσιακό. Οι δύο αυτοί μηχανισμοί είναι η κληρονομικότητα μέσω μοναδικού πίνακα (single-table inheritance) και η πολυμορφική συσχέτιση (polymorphic association).

Η κληρονομικότητα μέσω μοναδικού πίνακα (single-table inheritance - STI) υποδεικνύει την χρήση ενός και μόνο πίνακα στην βάση δεδομένων μας για να εκφράσει την κληρονομικότητα. Ο πίνακας αυτός περιέχει μία στήλη για κάθε ιδιότητα των κλάσεων που εμπλέκονται στην σχέση. Οι στήλες που αντιστοιχούν σε ιδιότητες των υποκλάσεων είναι προφανές ότι δεν πρέπει να έχουν περιορισμό NOT NULL. Επιπλέον, ο πίνακας περιλαμβάνει και μία στήλη που λέγεται κατά σύμβαση type και δηλώνει σε ποια συγκεκριμένη κλάση ανήκει η εγγραφή του πίνακα. Κάθε εγγραφή που αντιστοιχεί σε οντότητα ή χρήστη ή κοινότητα εισέρχεται σε αυτό τον πίνακα, με αντίστοιχα NULL τιμές στις στήλες που δεν αντιστοιχούν στο χαρακτήρα του δράστη που εκπροσωπεί.

Όμως η λύση αυτή οδηγεί στην δημιουργία ενός μεγάλου πίνακα με πολλές στήλες και πολλές εγγραφές. Επιπλέον, στην δική μας περίπτωση κάθε εγγραφή θα έχει πολλά NULL πεδία καθώς οι ιδιότητες της κλάσης χρήστη και την κλάσης κοινότητα έχουν πολλές διαφορετικές και λίγες κοινές ιδιότητες. Επομένως, η λύση που προσφέρει ο μηχανισμός STI δεν είναι αποδοτική στην δική μας περίπτωση.

Η πολυμορφική συσχέτιση από την άλλη μεριά προσφέρει μία εντελώς διαφορετική λύση. Στην επιστήμη των υπολογιστών, ο πολυμορφισμός είναι ο μηχανισμός που μας επιτρέπει να ορίσουμε την αφαιρετική προσέγγιση μιας διπροσωπίας, ανεξαρτήτως της υλοποίησής της. Ένα απλό παράδειγμα πολυμορφισμού είναι η μέθοδος της πρόσθεσης η οποία εκφράζεται μέσω ενός συμβόλου και λειτουργεί με ακέραιους, δεκαδικούς αλλά ακόμα και strings. Στην Ruby on Rails, μία πολυμορφική συσχέτιση επιτρέπει τη σύνδεση αντικειμένων διαφορετικού τύπου. Η υπόθεση υποδεικνύει ότι αυτά τα αντικείμενα μοιράζονται κάποια κοινά χαρακτηριστικά αλλά έχουν διαφορετικές αναπαραστάσεις. Στην πολυμορφική συσχέτιση, για κάθε κλάση που εμπλέκεται στην κληρονομική σχέση του αντικειμενοστραφούς μοντέλου αντιστοιχούμε έναν πίνακα στο σχεσιακό μοντέλο. Οι πολυμορφικές συσχετίσεις στην Ruby on Rails βασίζονται στο γεγονός ότι ένα ξένο κλειδί ενός πίνακα της βάσης δεδομένων μας δεν είναι αναγκαίο να είναι απλά ένας ακέραιος, σε αντίθεση με την αυστηρή σύμβαση που υπάρχει για τα πρωτεύοντα κλειδιά. Ο πίνακας που αντιστοιχεί στην κλάση πατέρα έχει δύο στήλες για το ξένο κλειδί. Η μία στήλη διατηρεί το id της εγγραφής του πίνακα στόχου (πίνακα υποκλάσης) και η δεύτερη στήλη υποδεικνύει στο Active Record σε ποιο μοντέλο ανήκει αυτό το κλειδί. Η ιδιαιτερότητα αυτή περιλαμβάνει και αυστηρή σύμβαση για την ονομασία του ξένου κλειδιού του πίνακα πατέρα. Συγκεκριμένα, αν για παράδειγμα το ξένο κλειδί ονομαστεί resource, τότε ο πίνακας πατέρας έχει μία στήλη με όνομα resource_id και μία στήλη με όνομα resource_type.

Ο μηχανισμός της πολυμορφικής συσχέτισης προσφέρει μία κομψή λύση και το Active Record παρέχει αυξημένη λειτουργικότητα για την διαχείριση των δεδομένων των πινάκων που εμπλέκονται στην κληρονομική μας σχέση που έχει εκφραστεί με τον μηχανισμό της πολυμορφικής συσχέτισης. Την λύση αυτή λοιπόν υιοθετήσαμε και εμείς για να εκφράσουμε την σχέση κληρονομικότητας που υφίσταται ανάμεσα στην κλάση πατέρα – entity και στις υποκλάσεις της user και community.

Αρχικά, πρέπει να δημιουργήσουμε τους πίνακες που αντιστοιχούν στις οντότητες του συστήματος. Η παρακάτω κλάση δημιουργεί στην βάση δεδομένων μας τον πίνακα

- entities (INTEGER id, INTEGER actor_id, STRING actor_type).

Σημειώνουμε ότι το ξένο κλειδί ονομάζεται actor.

```
class CreateEntities < ActiveRecord::Migration
  def self.up

    create_table :entities, :force => true do |t|
        t.column :actor_id, :integer
        t.column :actor_type, :string
    end
  end

  def self.down
    drop_table :entities
  end
end
```

Οι επόμενες κλάσεις δημιουργούν αντίστοιχα τους πίνακες της βάσης δεδομένων μας που αφορούν τις υποκλάσεις. Οι πίνακες είναι :

- users (INTEGER id, STRING name, STRING hashed_password, STRING salt, STRING firstname, STRING surname, STRING email, STRING address, STRING city, DATE insert_date, INTEGER age, INTEGER phone_number, STRING working_status)
- communities (INTEGER id, STRING name, STRING theme_of_interest, TEXT description, INTEGER user_id, DATE date)

και οι κλάσεις μας είναι :

```
class CreateUsers < ActiveRecord::Migration

  def self.up

      create_table :users, :force => true do |t|
      t.column :name, :string, :null => false
      t.column :hashed_password, :string, :null => false
      t.column :salt, :string,  :null => false
      t.column :firstname, :string, :null => false
      t.column :surname, :string, :null => false
      t.column :email, :string, :null => false
      t.column :address, :string
      t.column :city, :string
      t.column :insert_date, :date,  :null => false
      t.column :age, :integer
      t.column :phone_number, :integer
      t.column :working_status, :string

    end
  end


  def self.down
    drop_table :users
```

```
  end
end


class CreateCommunities < ActiveRecord::Migration
  def self.up
    create_table :communities, :force => true  do |t|
       t.column :name, :string, :null => false
       t.column :theme_of_interest, :string, :null =>
                                                 false
       t.column :description, :text
       t.column :user_id, :integer, :null => false
       t.column :date, :date, :null => false
    end

    execute "alter table communities add constraint
            fk_community_users foreign key (user_id)
            references users(id)"

  end


  def self.down
    drop_table :communities
  end
end
```

Παρατηρούμε ότι οι παραπάνω κλάσεις περιλαμβάνουν η καθεμία δύο μεθόδους, την μέθοδο up και την μέθοδο down. Επιπλέον, αποτελούν υποκλάσεις της κλάσης Migration του ActiveRecord. Επεκτείνοντας λοιπόν την λειτουργικότητα της κλάσης Migration, οι παραπάνω κλάσεις ενεργούν επί την βάση δεδομένων μας και πραγματοποιούν τη διαχείριση της βάσης δεδομένων. Συγκεκριμένα, η μέθοδος up περιέχει τις δηλώσεις των πεδίων, των τύπων των πεδίων και των περιορισμών των πινάκων και δημιουργεί πίνακα της βάσης δεδομένων μας ενώ η μέθοδος down διαγράφει τον πίνακα από την βάση δεδομένων μας. Με αυτό τον τρόπο, για την δημιουργία των πινάκων της βάσης μας δεν χρειάζεται να διαμορφώσουμε τα κατάλληλα sql statements για την διαχείριση των πινάκων της βάσης δεδομένων, αφού το ActiveRecord τα έχει υλοποιήσει σε πιο υψηλό επίπεδο. Το μοναδικό sql statement που διαμορφώσαμε στον κώδικά μας αφορά την δήλωση του ξένου κλειδιού για τον πίνακα communities:

```
execute "alter table communities add constraint
          fk_community_users foreign key (user_id)
          references users(id)"
```

Η δήλωση αυτή υποδεικνύει ότι το πεδίο user_id του πίνακας communities έχει εξωτερικό κλειδί που αναφέρεται στο πρωτεύον κλειδί του πίνακα users.

Εκτός από την διαμόρφωση των πινάκων της βάσης δεδομένων μας, διαμορφώνουμε και το μοντέλο μας. Οι αντίστοιχες κλάσεις του μοντέλου μας είναι οι :

```
class Entity < ActiveRecord::Base

      belongs_to :actor, :polymorphic => true

end


class User < ActiveRecord::Base

      has_one :entity, :as => :actor

end


class Community < ActiveRecord::Base

      has_one :entity, :as => :actor

end
```

Οι κλάσεις αυτές είναι υποκλάσεις της κλάσης Base του ActiveRecord και έχουν απόλυτη απόκριση προς τους πίνακες της βάσης δεδομένων μας. Όπως έχουμε αναφέρει, ένας πίνακας στην βάση δεδομένων μας αντιστοιχεί σε μία κλάση του μοντέλου μας και οι εγγραφές του πίνακα αντιστοιχούν σε αντικείμενα της κλάσης του μοντέλου μας. Στις κλάσεις του μοντέλου έχουμε τη δυνατότητα να προσθέσουμε περιορισμούς, όπως περιορισμούς ξένου κλειδιού, και να διαμορφώσουμε μεθόδους που επενεργούν άμεσα πάνω στις εγγραφές του πίνακά μας, χωρίς να καταφύγουμε στην χρήση της sql.

Σημειώνουμε ότι κατά σύμβαση το όνομα της κλάσης του μοντέλου είναι ίδιο με το όνομα του αντίστοιχου πίνακα της βάσης δεδομένων στον ενικό. Οι δηλώσεις που περιλαμβάνονται στις παραπάνω κλάσεις εκφράζουν την πολυμορφική συσχέτιση και κατ' επέκταση υλοποιούν την σχέση κληρονομικότητας ανάμεσα στην κλάση entity και τις κλάσεις user και community.

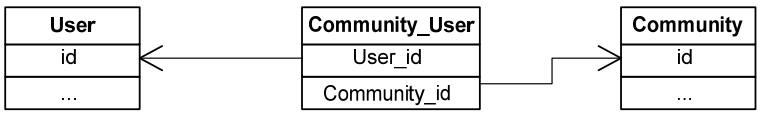
Μια ακόμα σημαντική σχέση που υφίσταται ανάμεσα στους δράστες του συστήματός μας αφορά την συμμετοχή των χρηστών σε κοινότητες. Ένας χρήστης μπορεί να συμμετέχει σε πολλές κοινότητες και μία κοινότητα μπορεί να έχει πολλούς χρήστες. Επιπλέον, ένας χρήστης μπορεί να έχει μοναδική συμμετοχή σε μία κοινότητα, που σημαίνει ότι συμμετέχει με συγκεκριμένο και μοναδικό ρόλο στην κοινότητα (simple user / administrator) και η συμμετοχή του σε μία κοινότητα καταγράφεται στην βάση μας σε μία και μοναδική εγγραφή.

Μια πρώτη προσέγγιση υποδεικνύει την ύπαρξη ενός επιπλέον πίνακα:

memberships (INTEGER user_id, INTEGER community_id, DATE join_date, STRING status)

Το πεδίο user_id του πίνακα memberships είναι ξένο κλειδί που αναφέρεται στο πρωτεύον κλειδί του πίνακα user. Αντίστοιχα, το πεδίο community_id είναι ξένο κλειδί που αναφέρεται στο πρωτεύον κλειδί του πίνακα community. Το πρωτεύον κλειδί του πίνακα memberships είναι σύνθετο και αποτελείται από τα δύο αυτά πεδία. Η Ruby on Rails όμως δεν υποστηρίζει σύνθετο πρωτεύον κλειδί (composite primary key). Άρα, σε αυτή την περίπτωση οδηγούμαστε στην αναζήτηση διαφορετικής λύσης.

Η σχέση συμμετοχής ανάμεσα στις κλάσεις User και Community μπορεί να εκφραστεί με την συσχέτιση "N-N"("πολλά-προς-πολλά"), που δηλώνει ακριβώς την σχέση που υφίσταται ανάμεσα στις κλάσεις User και Community, δηλαδή ότι ένας χρήστης μπορεί να είναι μέλος πολλών κοινοτήτων και μία κοινότητα μπορεί να έχει πολλούς χρήστες. Η συσχέτιση αυτή υποστηρίζεται από την Ruby on Rails με την δήλωση has_and_belongs_to_many.




```
Class User<ActiveRecord:Base
    has_and_belongs_to _many :communities
end
```

```
Class Community<ActiveRecord:Base
    has_and_belongs_to _many :users
end
```

Προσθέτουμε την δήλωση has_and_belong_to_many και στα δύο μοντέλα, καθώς η συσχέτιση "N-N" είναι συμμετρική. Στην ουσία, με αυτό τον τρόπο δημιουργούμε την ένωση των δύο πινάκων-στόχων, καθώς ως γνωστόν στις βάσεις δεδομένων οι συσχετίσεις "N-N" υλοποιούνται με τη βοήθεια intermediate join tables. Ένας intermediate join table περιέχει ζεύγη ξένων κλειδιών μέσω των οποίων συνδέονται οι δύο πίνακες-στόχοι. Το ActiveRecord υποθέτει ότι το όνομα αυτού του πίνακα είναι η συνένωση των ονομάτων των δύο πινάκων-στόχων σε αλφαβητική σειρά.

Ένα πρόβλημα που προκύπτει από την παραπάνω προσέγγιση αφορά την ανάγκη να διατηρήσουμε περισσότερες πληροφορίες για την συσχέτιση των χρηστών και των κοινοτήτων. Συγκεκριμένα, θέλουμε να διατηρούμε την πληροφορία σχετικά με το ρόλο του χρήστη στην κοινότητα καθώς και την ημερομηνία που έγινε μέλος της κοινότητας ο

χρήστης. Μία λύση που πρότεινε μία αρχική έκδοση της Ruby on Rails ήταν η μέθοδος push_with_attributes. Η λύση αυτή όμως δεν προτείνεται πλέον από την νεότερη έκδοση της Ruby on Rails.

Ένα ακόμα πρόβλημα εντοπίζεται στο γεγονός ότι επιδιώκουμε ένας χρήστης να έχει μοναδική εγγραφή που αφορά την συμμετοχή του σε μία συγκεκριμένη κοινότητα. Η παραπάνω όμως προσέγγιση δεν εγγυάται σε καμία περίπτωση την ύπαρξη μοναδικής συμμετοχής ενός χρήστη σε συγκεκριμένη κοινότητα.

Η τελευταία έκδοση της Ruby on Rails υποδεικνύει ότι τα join tables πρέπει να περιέχουν μόνο ζεύγη πεδίων που αποτελούν ξένα κλειδιά. Αν υπάρχει ανάγκη να διατηρούνται περισσότερα δεδομένα στο join table, η Ruby on Rails υποδεικνύει την δημιουργία ενός νέου μοντέλου. Σε αντίθεση με την απλή υλοποίηση της has_and_belong_to_many συσχέτισης, όπου ο πίνακας που αποτελεί την ένωση των δύο αρχικών πινάκων περιείχε γραμμές που δεν είχαν ανεξάρτητη υπόσταση, η ανάγκη προσθήκης πληροφορίας στον πίνακα ένωσης προσδίδει ανεξάρτητη υπόσταση στον πίνακα ο οποίος αξιώνει την ύπαρξη της δικής του κλάσης στο μοντέλο μας. Δημιουργούμε, λοιπόν, έναν ενδιάμεσο πίνακα, γιατί επιδιώκουμε να κρατήσουμε επιπλέον πληροφορίες για την συμμετοχή του χρήστη σε μία κοινότητα. Ο πίνακας της βάσης δεδομένων είναι :

memberships (INTEGER id, INTEGER user_id, INTEGER community_id, DATE join_date, STRING status)

Η παρακάτω κλάση δημιουργεί τον πίνακα.

```
class CreateMemberships < ActiveRecord::Migration
   def self.up
      create_table :memberships do |t|
           t.column :user_id, :integer ,:null => false
           t.column :community_id, :integer, :null => false
           t.column :join_date, :datetime, :null =>false
           t.column :membership_status, :char
      end


    execute "alter table memberships add constraint
fk_membership_users foreign key (user_id) references
users(id)"

    execute "alter table memberships add constraint
fk_membership_communities foreign key (community_id)
references communities(id)"

  end


  def self.down
    drop_table :memberships
  end


end
```

Ο πίνακας εμπεριέχει δύο περιορισμούς ξένου κλειδιού. Συγκεκριμένα, το πεδίο user_id του πίνακα είναι ξένο κλειδί που αναφέρεται στο id του πίνακα users. Αντίστοιχα, το πεδίο community_id είναι ξένο κλειδί που αναφέρεται στο id του πίνακα communities.

Στο μοντέλο της εφαρμογής μας προσθέτουμε την παρακάτω κλάση:

```
class Membership < ActiveRecord::Base

  belongs_to :user
  belongs_to :community


end
```

Οι δηλώσεις belongs_to δηλώνουν τους περιορισμούς εξωτερικού κλειδιού που αναφέραμε. Στο μοντέλο, πρέπει να προσθέσουμε κατάλληλες δηλώσεις στις κλάσεις user και community για να ολοκληρώσουμε την συσχέτιση:

```
class User < ActiveRecord::Base

  has_many :memberships, :dependent => true
  has_many :communities, :through => :memberships

end


class Community < ActiveRecord::Base

  has_many :memberships, :dependent => true
  has_many :users, :through => :memberships

end
```

Με την παραπάνω υλοποίηση, καταγράφουμε την εισαγωγή ενός χρήστη σε μία κοινότητα προσθέτοντας την κατάλληλη εγγραφή στον πίνακα memberships. Επιπλέον, δεν έχουμε χάσει τις δυνατότητες που μας προσφέρει η συσχέτιση has_and_belong_to_many, καθώς εξακολουθούμε να μπορούμε να ζητάμε με εύκολο τρόπο σε ποιες κοινότητες ανήκει ένας χρήστης όπως και ποιοι χρήστες ανήκουν σε μία κοινότητα. Σε αυτό το σημείο, βασικό ρόλο παίζει ο προσδιορισμός :through που συνοδεύει την δήλωση has_many. Ο προσδιορισμός :through κάνει εφικτή την πλοήγηση από το μοντέλο του χρήστη στο μοντέλο της κοινότητας διαμέσω του μοντέλου membership και αντίστοιχα. Ο κώδικάς μας για την εύρεση των κοινοτήτων στις οποίες ανήκει ένας χρήστης user είναι :

```
communities = user.communities
```

Παρασκηνιακά, η Ruby on Rails δημιουργεί τα απαραίτητα SQL queries για να επιστρέψει όλες τις γραμμές του πίνακα communities για τις οποίες ο χρήστης user έχει αντίστοιχη γραμμή στον πίνακα memberships. Συγκεκριμένα, η :through => :memberships παράμετρος υποδεικνύει στο Active Record να εντοπίσει την κλάση membership του μοντέλου για να

φτάσει τελικά στην κλάση communities του μοντέλου και να αναζητήσει τις εγγραφές από τον πίνακα communities. Η πλοήγηση αυτή εκτελείται με την βοήθεια των εξωτερικών κλειδιών του πίνακα memberships και συγκεκριμένα το user_id και το community_id.

Υιοθετώντας την παραπάνω λύση, δεν έχουμε κατοχυρώσει την ύπαρξη μοναδικής εγγραφής ενός χρήστη σε μία κοινότητα στον πίνακα membership. Δηλαδή ο πίνακας membership μπορεί να περιέχει πολλές εγγραφές που να υποδηλώνουν ότι ένας χρήστης ανήκει σε μία συγκεκριμένη κοινότητα. Για να κατοχυρώσουμε την ύπαρξη μοναδικής εγγραφής ενός χρήστη σε μία κοινότητα, θα πρέπει να προσθέσουμε έναν επιπλέον έλεγχο. Ο έλεγχος αυτός θα γίνεται κατά την προσπάθεια ενός χρήστη να γίνει μέλος μίας κοινότητας. Προτού λοιπόν εισάγουμε εγγραφή στον πίνακα memberships για να εκφράσουμε την συμμετοχή του χρήστη στην κοινότητα, ελέγχουμε τον πίνακα membership για να διαπιστώσουμε αν υπάρχει αντίστοιχη εγγραφή. Αν υπάρχει τέτοια εγγραφή, δεν θα γίνει νέα καταχώρηση και αρκεί να ενημερώσουμε τον χρήστη ότι ήδη αποτελεί μέλος αυτής της κοινότητας. Ωστόσο, αν δεν υπάρχει τέτοια εγγραφή, προσθέτουμε στον πίνακα membership την εγγραφή που δηλώνει την συμμετοχή του χρήστη στην κοινότητα. Ο έλεγχος αυτός είναι άρρηκτα συνδεδεμένος με την διαχείριση της βάσης δεδομένων και επομένως του μοντέλου μας. Γι' αυτό η μέθοδος που υλοποιεί αυτόν τον έλεγχο εισάγεται στην κλάση Membership του μοντέλου μας:

```
#Using this method we can validate that a user can have only one
#entry in a community. By doing so, we manage to create a composite
#primary key behaviour CPK_of_Membership ={user_id, community_id}.
     def self.user_already_in(user, community)

     #returns nil if the user is not a member of the community
           member =
           self.find_by_user_id_and_community_id(user, community)

           member

     end
```

Η μέθοδος αυτή δέχεται ως παραμέτρους το id της εγγραφής του πίνακα users που αντιστοιχεί στον χρήστη μας και το id της εγγραφής του πίνακα communities που αντιστοιχεί στην κοινότητα. Εκτελεί ένα δυναμικό query χρησιμοποιώντας την μέθοδο find() του Active Record. Η μέθοδος find εφαρμόζεται πάνω στην κλάση Membership και το query το οποίο εκτελεί είναι το εξής :

SELECT *

FROM memberships m

WHERE m.user_id = user AND m.community_id = community

Η μέθοδος επιστρέφει nil, δηλαδή το κενό αντικείμενο της Ruby, αν ο χρήστης δεν είναι μέλος της κοινότητας. Διαφορετικά, στην περίπτωση που ο χρήστης είναι μέλος της κοινότητας, η μέθοδος επιστρέφει την κατάλληλη εγγραφή του πίνακα membership.

Έχοντας δημιουργήσει τους κατάλληλους πίνακες στην βάση δεδομένων μας που αφορούν τις οντότητες - δράστες του συστήματός μας και έχοντας διαμορφώσει το μοντέλο για την διαχείριση αυτών των πινάκων από την εφαρμογή μας, θα μελετήσουμε και τα θέματα που αφορούν τις λειτουργίες αυτών των οντοτήτων. Σε αυτό λοιπόν το σημείο, θα παρουσιάσουμε τους controllers που χρειάστηκαν για την διαχείριση του μοντέλου μας.

Σύμφωνα με τον σχεδιασμό μας, για να αλληλεπιδράσει πλήρως ένας χρήστης του διαδικτύου με το σύστημά μας πρέπει να δημιουργήσει έναν λογαριασμό με username και password. Σε περίπτωση που ο χρήστης δεν έχει λογαριασμό, η αλληλεπίδραση με το σύστημά μας είναι περιορισμένη. Συγκεκριμένα, ο χρήστης μπορεί να προσπελάσει μόνο μία σελίδα του συστήματος, η οποία παρουσιάζει δυνατότητες εισόδου του χρήστη στο σύστημα (sign in), δυνατότητα εγγραφής στο σύστημα (sign up) και δυνατότητα αναζήτησης αγαθών (search). Για να πλοηγηθεί στις υπόλοιπες σελίδες του site μας, χρειάζεται να κάνει sign in. Επομένως, προκύπτει η ανάγκη ελέγχου πρόσβασης χρηστών. Γι' αυτό το λόγο, διαμορφώνουμε έναν controller ο οποίος διαχειρίζεται εισόδους από χρήστες ο οποίοι δεν έχουν εισέλθει στο σύστημά μας, αλλά μπορούν να αλληλεπιδράσει μερικώς με αυτό. Ο controller αυτός διαχειρίζεται ουσιαστικά τις εισόδους που έχουμε από την πρώτη σελίδα του συστήματος. Η κλάση που υλοποιεί αυτόν τον controller είναι η κλάση :

```
class EnterController < ApplicationController
```

Η κλάση αυτή αποτελεί υποκλάση της κλάσης ApplicationContoller και φέρει τις μεθόδους με τις οποίες το σύστημά μας διαχειρίζεται αλληλεπιδράσεις με χρήστη που δεν έχει εισέλθει στην VCommunity, και μπορεί να αλληλεπιδράσει μερικώς με το σύστημα. Συγκεκριμένα, οι μέθοδοι αυτοί είναι :

```
def index
  …
end

def sign_up
  …
end

def logout
  …
end

def search
  …
end
```

Επειδή οι views είναι στενά συνδεδεμένες με τους controllers, η δημιουργία του controller `EnterController` οδηγεί στην δημιουργία ενός view - component με όνομα Enter, ο οποίος περιέχει όλα τα rhtml αρχεία που αντιστοιχούν στις views που θα αποτελέσουν τις σελίδες στις οποίες θα πλοηγείται ο χρήστης, όταν θα δίνει εισόδους στο σύστημά μας που ενεργοποιούν τις μεθόδους του `EnterController`.

Για να καλέσουμε λοιπόν την σελίδα μας πληκτρολογούμε στον browser μας :

http://<host_name>/enter

Και με αυτό τον τρόπο καλείται η μέθοδος index του controller EnterController. Η σελίδα που παρουσιάζεται αντιστοιχεί στο αρχείο index.rhtml του enter component των views και είναι μία login page, που καλεί τον χρήστη να εισάγει username και password για να εισέλθει στο σύστημα. Η σελίδα αυτή περιλαμβάνει και κατάλληλα GUI στοιχεία για να δώσει δυνατότητα στον χρήστη να κάνει sign up, δηλαδή να δημιουργήσει λογαριασμό στο σύστημά μας, ή να κάνει μία αναζήτηση στα αγαθά του συστήματός μας. Η επιλογή του χρήστη θα αποτελέσει είσοδο για το σύστημά μας την οποία θα επεξεργαστεί ο EnterController με μία από τις μεθόδους που περιέχει. Το αποτέλεσμα της επιλογής του χρήστη θα είναι μία νέα σελίδα στο σύστημά μας.

Αρχικά, ας αναλύσουμε την υλοποίηση της εγγραφής του χρήστη στο σύστημά μας. Σε αυτή την περίπτωση, ο χρήστης θα ενεργοποιήσει το κατάλληλο hyperlink που υπάρχει στην login page και με αυτό τον τρόπο θα ενεργοποιήσει την μέθοδο sign_up. Η μέθοδος sign_up εμπεριέχει μία συνθήκη ελέγχου με την οποία διαπιστώνει τον χαρακτήρα ενεργοποίησης της μεθόδου. Η συνθήκη ελέγχου έχει την παρακάτω μορφή :

```
if request.post?
    …
end
```

Όταν η μέθοδος ενεργοποιείται λόγω ενός HTTP GET Request πλοηγούμαστε σε μία νέα σελίδα η οποία περιέχει μία φόρμα. Η φόρμα περιέχει κατάλληλα πεδία, τα οποία θα συμπληρώσουμε με προσωπικά μας στοιχεία, για να δημιουργήσουμε τον λογαριασμό μας στο σύστημα, και περιλαμβάνει και ένα submit button. Η φόρμα ζητάει της συμπλήρωση των εξής πεδίων :

- Username
- Password
- Password Confirmation
- First Name
- Surname

- Email
- Address
- Phone Number
- Age
- Working Status

Τα παιδία Address, Phone Number, Age, Working Status είναι προαιρετικού χαρακτήρα. Αφού συμπληρωθεί η φόρμα, με πάτημα του Submit button, ενεργοποιούμε ξανά την μέθοδο sign_up, αλλά αυτή τη φορά με HTTP Post Request. Το σώμα της συνθήκης ελέγχου που προαναφέραμε, διαμορφώνει την κατάλληλη εγγραφή για τον πίνακα users της βάσης μας και την αποθηκεύει. Με αυτό τον τρόπο ο χρήστης έχει δημιουργήσει τον λογαριασμό του στο σύστημά μας. Μετά την δημιουργία του λογαριασμού του ο χρήστης μας μπορεί να εισέλθει στο σύστημά μας.

Ωστόσο, ένα από τα σημαντικότερα θέματα που προκύπτει από την δημιουργία λογαριασμού χρήστη είναι η ασφάλεια του λογαριασμού του χρήστη. Ο λογαριασμός ενός χρήστη θα πρέπει να είναι προσωπικός, καθώς το site μας θα πρέπει να διαχειρίζεται με ασφάλεια προσωπικές πληροφορίες. Εξαιτίας αυτού διαφαίνεται η ανάγκη να χρησιμοποιήσουμε κρυπτογράφηση για να προστατεύσουμε τον κωδικό ενός χρήστη. Αντί να αποθηκεύσουμε , λοιπόν, στην βάση δεδομένων τον κωδικό ως απλό κείμενο, θα χρησιμοποιήσουμε την SHAI κρυπτογραφική «χώνεψη» (SHAI digest) για να αποθηκεύσουμε τελικά έναν κρυπτογραφημένο «αντιπρόσωπο» του κωδικού που δίνει ο χρήστης. Αυτή η μέθοδος κρυπτογράφησης είναι «μέθοδος μίας κατεύθυνσης», δηλαδή παρέχει την δυνατότητα επαλήθευσης κωδικού αλλά δεν επιτρέπει να ανακτήσουμε τον κωδικό, ο οποίος αποθηκεύεται στην βάση δεδομένων ως ένας 160 – bit hash.

Το πρώτο βήμα κατά την δημιουργία της εγγραφής του πίνακα users, που αντιστοιχεί στον λογαριασμό χρήστη, είναι η δημιουργία ενός αντικείμένου της κλάσης User του μοντέλου μας. Οι μεταβλητές αυτού του αντικειμένου τίθενται στις τιμές που έχουν προκύψει μέσω από την φόρμα υποβολής στοιχείων λογαριασμού του χρήστη. Ωστόσο, ο κωδικός δεν αποθηκεύεται ως απλό κείμενο, δηλαδή στην μορφή που εισήγαγε τον κωδικό ο χρήστης. Η διαδικασία μετατροπής του κωδικού που εισάγει ο χρήστης ως απλό κείμενο, κατά την υποβολή της φόρμας στοιχείων δημιουργίας λογαριασμού, στην μορφή στην οποία θα αποθηκευτεί στην βάση δεδομένων περιλαμβάνει τον συνδυασμό του κωδικού ως απλό κείμενο με μία τιμή που αποτελεί το λεγόμενο “salt” του κωδικού. Η τιμή αυτή είναι μοναδική για κάθε χρήστη. Ο συνδυασμός της με τον κωδικό οδηγεί στην δημιουργία ενός string μεγέθους 40 χαρακτήρων από δεκαεξαδικά ψηφία. Όταν ένας χρήστης υποβάλλει την φόρμα δημιουργίας λογαριασμού, πριν αποθηκευτεί η κατάλληλη εγγραφή στον πίνακα users

στην βάση δεδομένων πρέπει να διαμορφώσουμε τον κωδικό. Αρχικά, δημιουργούμε την τιμή "salt":

```
def create_new_salt
     self.salt = self.object_id.to_s + rand.to_s
end
```

Η τιμή του "salt" δεν έχει ιδιαίτερη σημειολογία. Αρκεί η τιμή να μην είναι προβλέψιμη. Η τιμή αυτή, όπως φαίνεται από την παραπάνω μέθοδο δημιουργίας "salt", είναι ένα string, το οποίο διαμορφώνεται με την βοήθεια του id του αντικειμένου της κλάσης User που έχει δημιουργηθεί και μίας τυχαίας τιμής (`rand.to_s`).Με αυτό τον τρόπο επιτυγχάνουμε την παραγωγή μίας μη προβλέψιμης τιμής. Εν συνεχεία, δημιουργούμε την κρυπτογραφημένη τιμή του κωδικού με την βοήθεια της παρακάτω μεθόδου :

```
def self.encrypted_password(password, salt)
      string_to_hash =
            password + "niki&anestis" + salt
      Digest::SHA1.hexdigest(string_to_hash)
end
```

Παρατηρούμε ότι ο κωδικός ως απλό κείμενο, συνδυάζεται με την τιμή "salt" σε ένα string. Επιπλέον, χρησιμοποιούμε και ένα ακόμα string, για να δυσκολέψουμε ακόμα περισσότερο την πρόβλεψη της τιμής του κωδικού, και με αυτό τον τρόπο παράγουμε τελικά ένα string που περιλαμβάνει και τον κωδικό ως απλό κείμενο. Το string αυτό υποβάλλεται στην SHAI «χώνεψη». Ο κωδικός στην τελική του μορφή καθώς και η τιμή "salt" αποθηκεύονται στην εγγραφή που αντιστοιχεί στον λογαριασμό του χρήστη στον πίνακα users της βάση δεδομένων.

Προτού όμως αναλύσουμε την υλοποίηση της εισόδου του χρήστη στο σύστημα, ας παρουσιάσουμε και την επιπλέον λειτουργικότητα που παρουσιάζεται στην αρχική σελίδα του συστήματος. Όπως αναφέραμε, ο χρήστης που δεν έχει εισέλθει στο σύστημά μας έχει τη δυνατότητα να αναζητήσει αγαθά που είναι καταχωρημένα. Αυτό επιτυγχάνεται μέσω μιας φόρμας η οποία περιλαμβάνει ένα πεδίο αναζήτησης καθώς και δύο drop down lists. Τα τελευταία του δίνουν τη δυνατότητα να κάνει την αναζήτησή του με βάση κάποια κριτήρια όπως το αν επιθυμεί κάποιο αγαθό που ζητείται ή προσφέρεται και το είδος της συναλλαγής όπως βλέπουμε και από το κώδικα για την αναζήτηση παρακάτω.

```
def search
  if params[:choice] == "Demand"
    @result = Commodity.find(:all, :include=>[:demand], :conditions
=> ["name like ? escape '!' and type_of_transaction = ?" ,
"%"+params[:field].gsub(/%/, "!%")+"%", params[:type]])

  elsif params[:choice] == "Offer"
```

```
    @result = Commodity.find(:all, :include=>[:offer], :conditions =>
["name like ? escape '!' and type_of_transaction = ?" ,
"%"+params[:field].gsub(/%/, "!%")+"%", params[:type]])
  end
end
```

Ο χρήστης μπορεί να εισάγει ολόκληρη τη φράση που αναζητά ή μέρος αυτής και να του εμφανίσει όλα τα αποτελέσματα που την περιέχουν. Αφήνοντας κενό το πεδίο αναζήτησης, εμφανίζονται όλα τα αγαθά που ικανοποιούν τα υπόλοιπα κριτήρια.

Ας αναλύσουμε την υλοποίηση της εισόδου χρήστη στο σύστημά μας. Η login page περιλαμβάνει μία φόρμα με δύο πεδία για την εισαγωγή του username και του password και ένα submit button. Εδώ ο χρήστης εισάγει τα στοιχεία του λογαριασμού του και πατάει το submit button. Με αυτό τον τρόπο ενεργοποιείται η μέθοδος index η οποία περιλαμβάνει συνθήκη ελέγχου της μορφής:

```
if request.post?
    …
end
```

Το HTTP Request με το οποίο ενεργοποιείται η μέθοδος index είναι τύπου POST. Εκτελείται το σώμα της παραπάνω εντολής ελέγχου το οποίο

- Ελέγχει την αυθεντικότητα των στοιχείων που εισήγαγε ο χρήστης
- Αν τα στοιχεία αντιστοιχούν σε έγκυρο λογαριασμό χρήστη
  - Δημιουργείται ένα αντικείμενο session που φέρει την πληροφορία του id του χρήστη που αντιστοιχεί στην εγγραφή του πίνακα users.
  - Οδηγούμαστε κατά σύμβαση στην κεντρική σελίδα του συστήματος ή στην σελίδα την οποία προσπάθησε να προσπελάσει χρήστης χωρίς να έχει κάνει sign in
- Αν τα στοιχεία δεν αντιστοιχούν σε έγκυρο λογαριασμό χρήστη
  - Ο χρήστης δεν επιτρέπεται να έχει πρόσβαση στο σύστημά μας και παραμένει στην login page
  - Εμφανίζεται κατάλληλο μήνυμα στην login page που ενημερώνει τον χρήστη ότι τα στοιχεία λογαριασμού που εισήγαγε δεν ήταν έγκυρα.

Σε αυτό το σημείο, θα αναλύσουμε τον ρόλο των sessions στο σύστημά μας. Η Ruby on Rails υποστηρίζει sessions με πολύ απλό τρόπο. Για να εισάγουμε λοιπόν τα sessions στον κώδικά μας, δημιουργήσαμε το μοντέλο session. Η Ruby on Rails κατά σύμβαση αποθηκεύει τα αντικείμενα sessions που δημιουργούνται σε απλό αρχείο. Επειδή θέλαμε να αποθηκεύονται

τα sessions στην βάση δεδομένων μας, αλλάξαμε την παραμετροποίηση της εφαρμογής μας. Σημειώνουμε ότι ο μηχανισμός που χρησιμοποιεί η Ruby on Rails για την υποστήριξη των session βασίζεται στα cookies. Τα cookies είναι αντικείμενα τα οποία αποστέλλει ο server στον client και κρατούνται στην μνήμη του client. Κατά συνέπεια, όταν ο browser στέλνει μία αίτηση στην εφαρμογή, τα cookies που αφορούν αυτή την εφαρμογή συνοδεύουν την αίτηση.

Το βασικό στοιχείο που επιδιώκουμε να κρατάμε ανάμεσα στις HTTP αιτήσεις αφορά το αναγνωριστικό του χρήστη (δηλαδή το id που αντιστοιχεί στην εγγραφή του χρήστη στον πίνακα users). Το αντικείμενο session δημιουργείται με κάθε είσοδο χρήστη στο σύστημά μας. Η πληροφορία η οποία φέρει είναι το αναγνωριστικό του χρήστη (id) που έχει εισέλθει σστο σύστημα. Για να κρατήσουμε την πληροφορία του id του χρήστη εισάγουμε την παρακάτω εντολή:

```
session[:user_id] = userID
```

Όπου το userID είναι το αναγνωριστικό του χρήστη, δηλαδή η τιμή του πεδίου id της εγγραφής του χρήστη στον πίνακα users.

Πριν προχωρήσουμε ας σχολιάσουμε τον έλεγχο αυθεντικότητας των στοιχείων του χρήστη που πραγματοποιείται κατά την προσπάθεια εισόδου του χρήστη στο σύστημα. Ο έλεγχος αυθεντικότητας των στοιχείων του χρηστή είναι άρρηκτα συνδεδεμένος με το μοντέλο User καθώς έχει άμεση αναφορά στον πίνακα users της βάσης δεδομένων. Συγκεκριμένα στην κλάση User του μοντέλου μας, συμπεριλαμβάνουμε την παρακάτω μέθοδο:

```
def self.authenticate(name, password)
      user = self.find_by_name(name)
      if user
         expected_password = encrypted_password(password, user.salt)
         if user.hashed_password != expected_password
                  user = nil
         end
      end
      user
end
```

Η μέθοδος αυτή αναζητά στον πίνακα users της βάσης δεδομένων μας, χρήστη με όνομα – username που δίδεται ως παράμετρος. Υπενθυμίζουμε ότι σύμφωνα με το σχεδιασμό μας, το username κάθε χρήστη είναι μοναδικό. Η μέθοδος χρησιμοποιεί την μέθοδο find_by_name που προκαλεί την εκτέλεση του παρακάτω query στην βάση μας:

SELECT *

FROM users

WHERE users.name = name

Αν δεν υπάρχει εγγραφή στον πίνακα users, η μέθοδος find_by_name θα επιστρέψει nil και τελικά η μέθοδος authenticate θα επιστρέψει nil. Αν υπάρχει εγγραφή στον πίνακα users, η μέθοδος find_by_name θα επιστρέψει ένα αντικείμενο της κλάσης User το οποίο θα αντιπροσωπεύει την εγγραφή αυτή. Έπειτα, θα γίνει σύγκριση του κωδικού με τον κωδικό που κρατάει η βάση δεδομένων. Όπως είπαμε, η κρυπτογράφηση που έχουμε χρησιμοποιήσει, επιτρέπει την σύγκριση ενός κωδικού με τον κωδικό που έχει θέσει ο χρήστης κατά την δημιουργία του λογαριασμού του. Χρησιμοποιώντας την τιμή "salt" που έχουμε αποθηκεύσει στην βάση δεδομένων μας για τον συγκεκριμένο χρήστη, τρέχουμε ξανά την μέθοδο `encrypted_password` που παρουσιάσαμε παραπάνω, με παραμέτρους τον κωδικό με τον οποίο προσπαθεί ο χρήστης να εισέλθει στο σύστημα και την τιμή "salt" που έχουμε κρατήσει ως πληροφορία στον λογαριασμό του συγκεκριμένου χρήστη. Η μέθοδος αυτή περνάει ξανά μέσα από SHAI «χώνεψη» τον κωδικό με τον οποίο προσπαθεί ο χρήστης να εισέλθει στο σύστημα. Παράγει λοιπόν έναν κρυπτογραφημένο κωδικό με τον ίδιο ακριβώς τρόπο με τον οποίο δημιουργήθηκε ο κωδικός που έχουμε κρατήσει στην βάση δεδομένων για τον συγκεκριμένο λογαριασμό χρήστη. Αυτόν τον κρυπτογραφημένο κωδικό συγκρίνουμε με την τιμή του κωδικού που κρατείται για τον χρήστη στον πίνακα users της βάσης δεδομένων. Αν οι κωδικοί ταυτίζονται, που σημαίνει ότι θα επιτραπεί η είσοδος του χρήστη στο σύστημα, η μέθοδος `authenticate` επιστρέφει αντικείμενο της κλάσης User που αντιστοιχεί στην εγγραφή του πίνακα users και αντιστοιχεί στον συγκεκριμένο χρήστη. Με αυτό τον τρόπο διαπιστώσαμε αν το username και password, που έδωσε χρήστης προσπαθώντας να εισέλθει στο σύστημά μας, αντιστοιχούν σε έγκυρο λογαριασμό.

Αφού διαπιστωθεί η εγκυρότητα του συνδυασμού username και password, ο χρήστης εισέρχεται στο σύστημα. Ο χρήστης τώρα μπορεί να δημιουργήσει κοινότητες, να συμμετάσχει σε κοινότητες, να δει λίστα με τις κοινότητες στις οποίες ανήκει και να ενημερωθεί για τους χρήστες – μέλη των κοινοτήτων. Η κεντρική σελίδα του συστήματος παρέχει κατάλληλα GUI στοιχεία με τα οποία ο χρήστης μπορεί να εκτελέσει τις παραπάνω ενέργειες, οι οποίες υλοποιούνται από κατάλληλες μεθόδους που, λόγω της λογικής συνάφειάς τους, εμπεριέχονται σε ειδική κλάση του controller component μας. Η κλάση αυτή είναι:

```
class CommunityController < ApplicationController
```

Έστω ότι ο χρήστης ενεργοποιεί την διαδικασία δημιουργίας μίας κοινότητας. Η μέθοδος που αντιστοιχεί στην δημιουργία της κοινότητας ονομάζεται `community_create` και εμπεριέχεται στην κλάση CommunityController. Η μέθοδος αυτή διαχειρίζεται τις εισόδους που στέλνονται ως παράμετροι ενός HTTP POST Request που προκύπτει από κατάλληλη φόρμα που συμπληρώνει ο χρήστης. Η φόρμα αυτή περιλαμβάνει τα παρακάτω πεδία:

- Όνομα Κοινότητας
- Θέμα Ενδιαφέροντος Κοινότητας
- Περιγραφή Κοινότητας

Η μέθοδος community_create όταν δεχθεί το HTTP POST Request δημιουργεί ένα αντικείμενο της κλάσης Community και θέτει τις μεταβλητές αυτού του αντικειμένου στις τιμές που έρχονται ως παράμετροι του HTTP POST Request. Ένα ακόμα στοιχείο που θέλουμε να κρατήσουμε για την κοινότητα είναι η ημερομηνία δημιουργίας της. Γι' αυτό το σκοπό χρησιμοποιούμε την μεταβλητή της Ruby

```
Time.now.to_date
```

Που μας επιστρέφει την τρέχουσα ημερομηνία. Επιπλέον, θέλουμε να κρατήσουμε και την πληροφορία του χρήστη - δημιουργού της κοινότητας. Συγκεκριμένα, μας χρειάζεται το id της εγγραφής αυτού του χρήστη στον πίνακα users. Την τιμή αυτή την λαμβάνουμε από το αντικείμενο session:

```
session[:user_id]
```

Με αυτό τον τρόπο έχουμε όλες τις τιμές στις οποίες θέτουμε τις μεταβλητές του αντικειμένου Community που δημιουργήσαμε. Οι κλάσεις του μοντέλου στην Ruby on Rails διαθέτουν την μέθοδο save. Η μέθοδος αυτή εκτελεί το κατάλληλο sql insert statement για την εισαγωγή εγγραφής στον πίνακα της βάσης δεδομένων, που αντιστοιχεί στην κλάση του μοντέλου, για το αντικείμενο της οποίας εκτελείται η μέθοδος. Εκτελούμε την μέθοδο save! για να σώσουμε την κατάλληλη εγγραφή στον πίνακα communities της ΒΔ. Η παραβίαση οποιουδήποτε περιορισμού, που υφίσταται για τον πίνακα της βάσης δεδομένων στον οποίο θα αποθηκευτεί η εγγραφή, οδηγεί στην έγερση ενός Exception και συγκεκριμένα του Exception `ActiveRecord::RecordInvalid`. Η διαχείριση αυτής της Exception μας διασώζει από παραβίαση περιορισμών καθώς και από λάθη κατά την εκτέλεση του κώδικά μας. Ένας από τους βασικούς περιορισμούς που διέπουν το μοντέλο της κοινότητας αφορά την μοναδικότητα του ονόματος της κοινότητας. Για να εκφράσουμε αυτό τον περιορισμό εισάγουμε την παρακάτω δήλωση στην κλάση Community του μοντέλου μας:

```
validates_uniqueness_of :name
```

Η παραβίαση αυτού του περιορισμού προλαμβάνεται λόγω της διαχείρισης της `ActiveRecord::RecordInvalid` Exception. Συγκεκριμένα, αν κατά την αποθήκευση της εγγραφής στον πίνακα communities εγερθεί `ActiveRecord::RecordInvalid` Exception, η εγγραφή δεν καταχωρείται τελικά και το σύστημα επιστρέφει στην αρχική του σελίδα με κατάλληλο ενημερωτικό μήνυμα για την αποτυχία δημιουργίας της κοινότητας.

Αν ωστόσο δεν υπάρξει πρόβλημα κατά την αποθήκευση της εγγραφής της νέας κοινότητας στον πίνακα communities, φροντίζουμε να σώσουμε στην ΒΔ την κατάλληλη εγγραφή στον πίνακα entities που αντιστοιχεί στην κοινότητα που δημιουργήσαμε. Επιπλέον, στον πίνακα memberships εισάγουμε κατάλληλη εγγραφή που υποδεικνύει ότι ο χρήστης - δημιουργός της κοινότητας αποτελεί μέλος της με την ιδιότητα του administrator. Αν οποιαδήποτε από αυτές τις εισαγωγές εγγραφών στους πίνακες τις βάσης μας αποτύχει, θα εγερθεί `ActiveRecord::RecordInvalid` Exception. Σε αυτή την περίπτωση, διαγράφουμε όλες τις υπόλοιπες εγγραφές που συνέβησαν κατά την δημιουργία της κοινότητας, η δημιουργία κοινότητας αποτυγχάνει και πλοηγούμαστε στην αρχική σελίδα, με κατάλληλο μήνυμα που μας ενημερώνει για την αποτυχία δημιουργίας της κοινότητας. Διαφορετικά, η δημιουργία της κοινότητας ολοκληρώνεται επιτυχώς και ο χρήστης πλοηγείται στην αρχική σελίδα με όπου του παρουσιάζεται και κατάλληλο ενημερωτικό μήνυμα για την επιτυχή δημιουργία της κοινότητας.

Μία ακόμα μέθοδος που παρέχει το σύστημά μας αφορά την ενημέρωση του χρήστη για τις υπάρχουσες κοινότητες. Η μέθοδος αυτή ονομάζεται `show_communities` και περιλαμβάνει την εντολή

```
Community.find(:all, :order => "name")
```

Η μέθοδος find είναι μία μέθοδος της κλάσης `ActiveRecord::Base`. Η κλάση Community ως υποκλάση της `ActiveRecord::Base`, κληρονομεί την μέθοδο αυτή. Η μέθοδος find δημιουργεί και τρέχει ένα query στον πίνακα που αντιστοιχεί στην κλάση για την οποία καλείται η μέθοδος. Στην παραπάνω περίπτωση το query που πραγματοποιεί η find είναι:

```
SELECT *
FROM communities
ORDER BY communities.name
```

Η μέθοδος επιστρέφει έναν πίνακα από αντικείμενα της κλάσης Community, που αντιστοιχούν στις εγγραφές του πίνακα communities της ΒΔ. Το μέγεθος του πίνακα είναι ίσο με το πλήθος των εγγραφών του πίνακα communities. Τα στοιχεία του πίνακα είναι ταξινομημένα κατά αύξουσα σειρά με βάση το όνομα της κοινότητας.

Η view που αντιστοιχεί σε αυτή τη μέθοδο καλείται show_communities.rhtml και παρουσιάζει σε λίστα τις κοινότητες του συστήματός. Ο πίνακας των κοινοτήτων που αποτελεί τοπική μεταβλητή της μεθόδου show_communities του `CommunityController` αποτελεί παράμετρο που προσπελαύνει η αντίστοιχη view και παρουσιάζει σε λίστα τις κοινότητες του συστήματος.

Ο σχεδιασμός του συστήματός μας υποδεικνύει και την δυνατότητα ενός χρήστη να εντάσσεται σε μία κοινότητα. Η λίστα των κοινοτήτων παρέχει κατάλληλα buttons. Η

ενεργοποίηση ενός button οδηγεί στην εκτέλεση της μεθόδου join που εμπεριέχεται στην κλάση `CommunityController`. Η μέθοδος αυτή εκτελεί τις παρακάτω ενέργειες :

- Ελέγχει αν ο χρήστης είναι ήδη μέλος της κοινότητας
- Αν ο χρήστης δεν είναι μέλος της κοινότητας, εντάσσεται στην κοινότητα
- Διαφορετικά, ο χρήστης δεν εντάσσεται στην κοινότητα
- Ο χρήστης πλοηγείται στην σελίδα με την λίστα των κοινοτήτων όπου παρουσιάζεται κατάλληλο μήνυμα για την επιτυχή ή μη επιτυχή ένταξη του χρήστη στην κοινότητα.

Ο χρήστης έχει την δυνατότητα να δει τις κοινότητες στις οποίες ανήκει. Η μέθοδος που υλοποιεί αυτή την λειτουργικότητα καλείται check_my_membership, η οποία:

- Βρίσκει την εγγραφή που αντιστοιχεί στον συγκεκριμένο χρήστη, με την βοήθεια του αντικειμένου session
- Ανακτά σε πίνακα τις εγγραφές του πίνακα memberships που υποδεικνύουν σε ποιες κοινότητες ανήκει ο χρήστης

Η view που αντιστοιχεί σε αυτή τη μέθοδο είναι η check_my_membership.rhtml. Ο πίνακας των αντικειμένων της κλάσης Membership, τον οποίο διαμορφώσαμε από την μέθοδο check_my_membership, προσπελαύνεται κατάλληλα από το rhtml μας, το οποίο παράγει μία html σελίδα που περιλαμβάνει μία λίστα με τις κοινότητες στις οποίες ανήκει ο χρήστης.

Επιπλέον, στον χρήστη του συστήματός μας παρέχεται η δυνατότητα να ενημερωθεί για τα μέλη μίας κοινότητας. Συγκεκριμένα, από την σελίδα με την λίστα των κοινοτήτων του συστήματός μας, έχει τη δυνατότητα πατώντας κατάλληλο button να δει τα μέλη της κοινότητας. Η μέθοδος καλείται `members` και εκτελεί τα παρακάτω :

- Ανακτά το αντικείμενο που αντιστοιχεί στην κοινότητα, για της οποίας τα μέλη ο χρήστης επιθυμεί να ενημερωθεί
- Ανακτά τις εγγραφές του πίνακα users που αφορούν μέλη της κοινότητας αυτής σε αντικείμενα της κλάσης User και τα εισάγει σε πίνακα.

Το αρχείο members.rhtml που αποτελεί την view μας σε αυτή την περίπτωση, παρουσιάζει μία λίστα με τα μέλη της κοινότητας που έχει επιλέξει ο χρήστης. Διαχειρίζεται τον πίνακα των αντικειμένων της κλάσης User που προέκυψε από την εκτέλεση της μεθόδου members.

5.1.2 Εφαρμογή αλγορίθμου υπόληψης

Το υποσύστημα της υπόληψης είναι ένα από τα βασικά στοιχεία της δομής του συστήματός μας. Οι απαιτήσεις του χαρακτηριστικού της υπόληψης οδήγησαν στην σχεδίαση και ανάπτυξη ενός μοντέλου υπόληψης, του οποίου την υλοποίηση θα παρουσιάσουμε σε αυτό το

εδάφιο. Σημειώνουμε ότι ο σχεδιασμός του συστήματός της υπόληψης είχε μεριμνήσει για την ύπαρξη κατηγοριών αγαθών. Ωστόσο, η υλοποίηση του συστήματός μας, σε πρώτο επίπεδο, δεν θα αντιμετωπίσει το θέμα της κατηγοριοποίησης των αγαθών και γι' αυτό το λόγο το σύστημα της υπόληψης δεν θα λάβει υπόψη του την παράμετρο της κατηγοριοποίησης των αγαθών.

Σχετικά με το υποσύστημα της υπόληψης, η πληροφορία για την δραστηριότητα μίας οντότητας αφορά:

- Τους βαθμούς που έχει λάβει η οντότητα για συναλλαγές που έχει πραγματοποιήσει η ίδια η οντότητα
- Το πλήθος των συναλλαγών που έχει πραγματοποιήσει η ίδια η οντότητα
- Τον τελικό βαθμό υπόληψης της οντότητας, όπως διαμορφώνεται από τις αλληλεπιδράσεις των βαθμών των οντοτήτων του συστήματός μας

Γι' αυτό το λόγο, η βάση δεδομένων μας περιλαμβάνει τον παρακάτω πίνακα που κρατά αυτή την πληροφορία:

Reputations(INTEGER id, INTEGER entity_id, DECIMAL grade, INTEGER transNo, DECIMAL final_grade)

Η παρακάτω κλάση δημιουργεί τον πίνακα αυτό στην βάση δεδομένων μας:

```
class CreateReputations < ActiveRecord::Migration
    def self.up
       create_table :reputations do |t|
            t.column :entity_id, :integer
            t.column :grade, :decimal, :precision => 6, :scale => 4
            t.column :transNo, :integer
            t.column :final_grade, :decimal, :precision => 6,
                        :scale => 4
       end

       execute "alter table reputations add constraint
            fk_reputation_entities foreign key (entity_id) references
            entities(id)"
     end

     def self.down
          drop_table :reputations
     end
end
```

Ο σχεδιασμός του υποσυστήματος της υπόληψης μας οδήγησε στην ανάγκη να κρατάμε μία view που υποδεικνύει το μέσο πλήθος συναλλαγών ανά «ενεργό» χρήστη- μέλος κοινότητας. Όμως η Ruby on Rails, δεν υποστηρίζει views. Η αναγκαιότητα όμως να διατηρούμε την πληροφορία του μέσου πλήθους συναλλαγών ανά «ενεργό» χρήστη- μέλος κοινότητας, μας οδήγησε στην δημιουργία ενός πίνακα που θα φέρει αυτή την πληροφορία. Ο πίνακας αυτός, φέρει πληροφορίες για την δραστηριότητα των μελών μίας κοινότητας και συγκεκριμένα,

καταγράφει το πλήθος των ενεργών μελών της κοινότητας, το συνολικό πλήθος συναλλαγών που έχουν εκτελέσει τα μέλη της και το μέσο πλήθος συναλλαγών ανά ενεργό μέλος της. Η μορφή του είναι:

Vactivities(INTEGER id, INTEGER community_id, INTEGER activeUsers, INTEGER transNO, DECIMAL avg)

Η κλάση που δημιουργεί αυτό τον πίνακα είναι:

```
class CreateVactivities < ActiveRecord::Migration
   def self.up
     create_table :vactivities do |t|
       t.column :community_id, :integer
       t.column :activeUsers, :integer
       t.column :transNo, :integer
       t.column :avg, :decimal, :precision => 6, :scale => 4

     end

     execute "alter table vactivities add constraint
            fk_vactivity_communities foreign key (community_id)
            references communities(id)"

   end

   def self.down
     drop_table :vactivities
   end

end
```

Αυτοί οι δύο πίνακες αποτελούν το μοντέλο του υποσυστήματος της υπόληψης. Οι κλάσεις που δομούν το μοντέλο του υποσυστήματος της υπόληψης αντιστοιχούν στους δύο αυτούς πίνακες και είναι οι :

```
class Reputation < ActiveRecord::Base

      belongs_to :entity

end
```

και

```
class Vactivity < ActiveRecord::Base

    belongs_to :community

end
```

Σημειώνουμε ότι ο πίνακας reputations έχει το πεδίο entity_id που είναι ξένο κλειδί και αναφέρεται στον πίνακα entities. Αντίστοιχα, ο πίνακας vactivity έχει το πεδίο community_id που είναι ξένο κλειδί και αναφέρεται στον πίνακα vactivity.

Ας, αναπτύξουμε τώρα την υλοποίηση των controllers που διαχειρίζονται την ανανέωση των βαθμών υπόληψης των οντοτήτων του συστήματός μας. Η μεθοδολογία υπολογισμού της υπόληψη ενός χρήστη και μίας κοινότητας διαφέρουν. Το γεγονός αυτό οδήγησε στην ανάγκη διαμόρφωσης δύο μεθόδων, εκ των οποίων η πρώτη αφορά την υπόληψη ενός χρήστη και η δεύτερη την υπόληψη μίας κοινότητας.

Η υπόληψη ενός χρήστη είναι η συνισταμένη δύο παραμέτρων, της μέσης βαθμολογίας που έχει λάβει ο χρήστης για συναλλαγές που έχει εκτελέσει ο ίδιος και του μέσου βαθμού υπόληψης που έχουν οι κοινότητες στις οποίες συμμετέχει ο χρήστης. Φυσικά, αναφερόμαστε στις «ενεργές» κοινότητες στις οποίες ανήκει ο χρήστης, δηλαδή στις κοινότητες οι οποίες έχουν τελικό βαθμό υπόληψης. Στην ειδική περίπτωση του «ανεξάρτητου» χρήστη, η υπόληψη δομείται αποκλειστικά από τους βαθμούς που έχει λάβει ο χρήστης για συναλλαγές που έχει πραγματοποιήσει ο ίδιος. Ενώ, λοιπόν, στην γενική περίπτωση υπάρχει διαφορά ανάμεσα στον μέσο βαθμό υπόληψης του χρήστη εξαιτίας των βαθμών που έχει λάβει για συναλλαγές που έχει διεκπεραιώσει ο ίδιος και τον τελικό βαθμό υπόληψής του, στην περίπτωση του «ανεξάρτητου» χρήστη, οι δύο αυτοί βαθμοί ταυτίζονται.

Η ανανέωση λοιπόν του βαθμού υπόληψης ενός χρήστη συντελείται εξαιτίας:

- της ανανέωσης της μέσης βαθμολογίας συναλλαγών που έχει πραγματοποιήσει ο χρήστης
- της ανανέωσης του μέσου βαθμού υπόληψης των κοινοτήτων στις οποίες αυτός ανήκει

Συγκεκριμένα, το γεγονός που επιφέρει αλλαγή της μέσης βαθμολογίας συναλλαγών ενός χρήστη είναι η ολοκλήρωση μίας συναλλαγής του χρήστη, η οποία σηματοδοτείται από την βαθμολόγηση που προσδίδει στον χρήστη το άλλο μέλος της συναλλαγής. Από την άλλη μεριά, ο μέσος βαθμός υπόληψης των κοινοτήτων στις οποίες ανήκει ο χρήστης μεταβάλλεται εξαιτίας διαφόρων συμβάντων, που συνοψίζονται στα εξής:

- ολοκλήρωση συναλλαγής που έχει εκτελέσει μία από τις κοινότητες στις οποίες συμμετέχει ο χρήστης
- δήλωση συμμετοχής του χρήστη σε κοινότητα που έχει τελικό βαθμό υπόληψης
- διαγραφή χρήστη από κοινότητα στην οποία συμμετέχει και η οποία έχει τελικό βαθμό υπόληψης
- εισαγωγή/ διαγραφή «ενεργού» μέλους από κοινότητα στην οποία ανήκει ο χρήστης
- ολοκλήρωση συναλλαγής από μέλος κοινότητας στην οποία ανήκει ο χρήστης

Ας ξεκινήσουμε από την περίπτωση κατά την οποία την οποία ο βαθμός υπόληψης του χρήστη ανανεώνεται εξαιτίας βαθμού που λαμβάνει μετά την ολοκλήρωση συναλλαγής που εκτελεί ο ίδιος. Το πρώτο βήμα, αφορά την εύρεση της εγγραφής που κρατά την

πληροφορία της υπόληψης του χρήστη. Αυτή η εγγραφή είναι εγγραφή του πίνακα reputations. Αν αυτή η εγγραφή δεν υπάρχει, συνάγουμε ότι ο χρήστης δεν έχει λάβει βαθμολογία για συναλλαγή που έχει εκτελέσει και δεν έχει επηρεαστεί από την υπόληψη κοινοτήτων στις οποίες ενδεχομένως ανήκει. Σε αυτή την περίπτωση, δημιουργούμε μία εγγραφή στον πίνακα reputations για τον χρήστη. Το επόμενο βήμα αφορά την ανανέωση του μέσου βαθμού υπόληψης του χρήστη λόγω συναλλαγών που έχει διεκπεραιώσει ο ίδιος. Με αυτό τον τρόπο ανανεώνουμε την τιμή της στήλης grade του πίνακα reputations. Συγχρόνως, αυξάνουμε κατά ένα την ένδειξη του πλήθους των συναλλαγών που έχει εκτελέσει ο χρήστης, δηλαδή την τιμή της στήλης transNo του πίνακα reputations. Στην περίπτωση που αναφερόμαστε σε «ανεξάρτητο» χρήστη, το τελευταίο βήμα είναι να θέσουμε τον τελικό βαθμό υπόληψης του χρήστη ίσο με τον μέσο βαθμό υπόληψης του χρήστη λόγω των συναλλαγών που έχει εκτελέσει ο ίδιος και να σώσουμε την ανανεωμένη εγγραφή του χρήστη στον πίνακα reputations. Διαφορετικά, η ανανέωση της μέσης βαθμολογίας του χρήστη επηρεάζει τους τελικούς βαθμούς υπόληψης των κοινοτήτων στις οποίες συμμετέχει. Επομένως, το επόμενο βήμα είναι να ανανεώσουμε τους τελικούς βαθμούς υπόληψης των κοινοτήτων τις οποίες μετέχει ο χρήστης. Ο τύπος υπολογισμού του τελικού βαθμού υπόληψης της κοινότητα είναι:

$$\text{C\_FinalGrade(i,j)} = w \cdot C\_grade(i,j) + (1-w) \cdot \Big(\sum_{\substack{1 \le l \le d \\ user\_l \in community\_i}} U\_grade(l,j) \Big) / m$$

Όπου $$w = \frac{C\_Trans(i,j)}{MeanU\_Trans(i,j) + C\_Trans(i,j)}$$

Και $$\text{MeanU\_Trans(i, j)} = \sum_{\substack{1 \le l \le d \\ user\_l \in community\_i}} U\_Trans(l,j) / m$$

Παρατηρούμε ότι η λήψη βαθμού ενός χρήστη για συναλλαγή που έχει εκτελέσει ο ίδιος, συνεπάγεται την ανανέωση:

- της τιμής του μέσου πλήθους συναλλαγών χρήστη-μέλους κοινότητας MeanU_Trans(i, j) και συνεπώς του παράγοντα w
- της δεύτερης συνιστώσας του τύπου υπολογισμού τελικού βαθμού υπόληψης χρήστη

Συνεπώς, για κάθε κοινότητα στην οποία ανήκει ο χρήστης, πρέπει να επαναϋπολογίζουμε τον παράγοντα $\sum_{\substack{1 \le l \le d \\ user\_l \in community\_i}} U\_grade(l,j) / m$ καθώς και τον παράγοντα $\sum_{\substack{1 \le l \le d \\ user\_l \in community\_i}} U\_Trans(l,j) / m$ για τους m το πλήθος «ενεργούς» χρήστες της κάθε κοινότητας

Για κάθε κοινότητα στην οποία ανήκει ο χρήστης μας, κατά τον υπολογισμό των παραπάνω παραγόντων, υπολογίζουμε τους «ενεργούς» χρήστες-μέλη της κοινότητας και τις συνολικές συναλλαγές που έχουν διεκπεραιώσει όλα τα μέλη της. Αυτές οι παράμετροι είναι σημαντικές για τους υπολογισμούς που εκτελούμε στο υποσύστημα της υπόληψης για λόγους που θα αναπτύξουμε παρακάτω. Αυτές λοιπόν τις παραμέτρους φροντίζουμε να αποθηκεύσουμε στον πίνακα vactivities. Συγκεκριμένα, στην στήλη activeUsers, αποθηκεύουμε το πλήθος των «ενεργών» χρηστών-μελών της κοινότητας, στην στήλη transNO, το συνολικό πλήθος των συναλλαγών που έχουν εκτελέσει τα «ενεργά» μέλη της κοινότητας και τέλος στην στήλη avg αποθηκεύουμε το μέσο όρο συναλλαγών ανά χρήστη της κοινότητας.

Αφού υπολογίσουμε τον παράγοντα $\sum_{\substack{1\le l\le d \\ user\_l\in community\_i}} U\_Trans(l,j)/m$, υπολογίζουμε τον παράγοντα w. Έπειτα, έχοντας ήδη υπολογίσει τον παράγοντα $\sum_{\substack{1\le l\le d \\ user\_l\in community\_i}} U\_grade(l,j)/m$ υπολογίζουμε τελικά τον τελικό βαθμό υπόληψης της κοινότητας.

Η ανανέωση των τελικών βαθμών των κοινοτήτων στις οποίες ανήκει ο χρήστης, όπως παρουσιάσαμε παραπάνω, συνεπάγεται την ανανέωση των τελικών βαθμών των χρηστών αυτών των κοινοτήτων σύμφωνα με τον τύπο:

$$\text{U\_FinalGrade(i, j)} = w \cdot U\_grade(i,j) + (1-w)\cdot\Big(\sum_{\substack{1\le l\le v \\ user\_i\in community\_l}} C\_FinalGrade(l,j)\Big)/m$$

Ο πρώτος υπολογισμός που πρέπει να γίνει αφορά τον παράγοντα w:

$$w = \frac{U\_Trans(i,j)}{MeanC\_Trans(i,j) + U\_Trans(i,j)}$$

όπου $\text{MeanC\_Trans(i, j)} = \Big(\sum_{\substack{1\le c\le v \\ user\_i\in community\_c}} \text{CommunityActivity(c, j)}\Big) / \text{n}$

και

$$\text{CommunityActivity(i, j)} = \frac{TotalTransOfActiveUsers(i,j) + C\_Trans(i,j)}{NoOfActiveUsers(i,j)+1}$$

Άρα, για κάθε κοινότητα στην οποία ανήκει ο χρήστης και η οποία έχει τελικό βαθμό υπόληψης, υπολογίζουμε τον παράγοντα CommunityActivity(i, j). Η παράμετρος C_Trans(i, j) είναι το πλήθος των συναλλαγών που έχει διεκπεραιώσει η ίδια η κοινότητα και προκύπτει άμεσα από την εγγραφή της κοινότητας, την οποία μελετάμε, στον πίνακα reputations. Η παράμετρος TotalTransOfActiveUsers(i, j) είναι το συνολικό πλήθος των συναλλαγών των «ενεργών» χρηστών-μελών της και η παράμετρος NoOfActiveUsers(i, j) είναι ο συνολικός

αριθμός των «ενεργών» χρηστών-μελών της κοινότητας. Εξάγουμε τις πληροφορίες αυτές άμεσα από τον πίνακα vactivities. Έτσι, μπορούμε να υπολογίσουμε την τιμή της CommunityActivity(i, j) για όλες τις κοινότητες στις οποίες ανήκει ο χρήστης και η οποίες έχουν τελικό βαθμό υπόληψης.

Ο τύπος MeanC_Trans(i, j) = ($\sum_{\substack{1\le c\le v \\ user\_i\in community\_c}} \text{CommunityActivity(c, j)}$) / n υποδεικνύει την

μέση δραστηριότητα των κοινοτήτων στις οποίες ανήκει ο χρήστης. Ο διαιρέτης n είναι το πλήθος των κοινοτήτων στις οποίες ανήκει ο χρήστης και οι οποίες έχουν τελικό βαθμό υπόληψης.

Αφού υπολογίσουμε τις παραπάνω τιμές μπορούμε να υπολογίσουμε τον παράγοντα w, λαμβάνοντας υπόψη ότι η παράμετρος U_Trans(i, j) είναι το πλήθος συναλλαγών που έχει διεκπεραιώσει ο χρήστης μας.

Το επόμενο βήμα, είναι ο υπολογισμός του παράγοντα $\sum_{\substack{1\le l\le v \\ user\_i\in community\_l}} C\_FinalGrade(l, j)/m$

δηλαδή του μέσου τελικού βαθμού υπόληψης των κοινοτήτων στις οποίες ανήκει ο χρήστης. Σημειώνουμε ότι ο διαιρέτης m είναι το πλήθος των κοινοτήτων στις οποίες ανήκει ο χρήστης και έχουν τελικό βαθμό υπόληψης.

Μετά από τον υπολογισμό των παραπάνω τιμών, εφαρμόζουμε τον τύπο:

$$\text{U\_FinalGrade(i, j)} = w \cdot U\_grade(i, j) + (1 - w) \cdot \left(\sum_{\substack{1\le l\le v \\ user\_i\in community\_l}} C\_FinalGrade(l, j) \right) / m$$

Με αυτό τον τρόπο, προκύπτει ο τελικός βαθμός υπόληψης χρήστη καθώς και οι επιπλέον ανανεώσεις των υπολήψεων των οντοτήτων που επηρεάζονται, μετά την λήψη βαθμού για συναλλαγή που διεκπεραίωσε ο χρήστης.

Αντίστοιχα βήματα ακολουθεί το υποσύστημα της υπόληψης όταν μία κοινότητα λάβει βαθμό για συναλλαγή την οποία διεκπεραίωσε η ίδια. Το πρώτο βήμα λοιπόν κατά την λήψη βαθμού μίας κοινότητας είναι η ανάκτηση της εγγραφής του πίνακα reputations που αφορά την κοινότητα αυτή. Αν ωστόσο δεν υπάρχει τέτοια εγγραφή, το υποσύστημα της υπόληψης δημιουργεί αυτή την εγγραφή. Μία ακόμα εγγραφή που πρέπει να ανακτηθεί είναι η εγγραφή του πίνακα vactivities για την συγκεκριμένη κοινότητα. Η εγγραφή αυτή αφορά την δραστηριότητα της κοινότητας και εκτός από τα στοιχεία τα οποία θα λάβουμε για τους υπολογισμούς μας είναι προφανές ότι οι τιμές που κρατά πρέπει να ανανεωθούν. Το επόμενο βήμα αφορά την ανανέωση της μέσης βαθμολογίας που έχει λάβει η κοινότητα για συναλλαγές που έχει εκτελέσει η ίδια. Ανανεώνεται λοιπόν η τιμή του πεδίου grade της εγγραφής που αφορά την κοινότητα στον πίνακα reputations καθώς και αυξάνεται κατά ένα η

τιμή του πεδίου transNo της εγγραφής που αφορά την κοινότητα στον πίνακα reputations και δείχνει το πλήθος των συναλλαγών που έχει διεκπεραιώσει η κοινότητα. Όμως η ανανέωση της μέσης βαθμολογίας της κοινότητας, συνεπάγεται την ανανέωση του τελικού βαθμού υπόληψης της κοινότητας όπως μπορούμε να καταλάβουμε από τον παρακάτω τύπο:

$$\text{C\_FinalGrade(i,j)} = w \cdot C\_grade(i,j) + (1-w) \cdot \Big(\sum_{\substack{1 \le l \le d \\ user\_l \in community\_i}} U\_grade(l,j) \Big) / m$$

Όπου $$w = \frac{C\_Trans(i,j)}{MeanU\_Trans(i,j) + C\_Trans(i,j)}$$

Και $$\text{MeanU\_Trans(i, j)} = \sum_{\substack{1 \le l \le d \\ user\_l \in community\_i}} U\_Trans(l,j) / m$$

Παρατηρούμε ότι στην περίπτωση κατά την οποία η κοινότητα λαμβάνει βαθμό για μία συναλλαγή που διεκπεραίωσε, ανανεώνεται η τιμή της πρώτης συνιστώσας του παραπάνω τύπου, όπως έχουμε ήδη αναφέρει, καθώς και η τιμή του παράγοντα w. Η τιμή του παράγοντα w μεταβάλλεται λόγω της ανανέωσης της τιμής του πλήθους των συναλλαγών που έχει διεκπεραιώσει η κοινότητα, C_Trans(i, j).

Η ανανέωση της τιμής του τελικού βαθμού υπόληψης της κοινότητας συνεπάγεται την ανανέωση των τελικών βαθμών υπόληψης των μελών της κοινότητας, σύμφωνα με τον παρακάτω τύπο:

$$\text{U\_FinalGrade(i, j)} = w \cdot U\_grade(i,j) + (1-w) \cdot \Big(\sum_{\substack{1 \le l \le v \\ user\_i \in community\_l}} C\_FinalGrade(l,j) \Big) / m$$

Όπου $$w = \frac{U\_Trans(i,j)}{MeanC\_Trans(i,j) + U\_Trans(i,j)}$$

$$\text{MeanC\_Trans(i, j)} = \Big(\sum_{\substack{1 \le c \le v \\ user\_i \in community\_c}} \text{CommunityActivity(c, j)} \Big) / \text{n}$$

και

$$\text{CommunityActivity(i, j)} = \frac{TotalTransOfActiveUsers(i,j) + C\_Trans(i,j)}{NoOfActiveUsers(i,j) + 1}$$

Η μεταβολή των τιμών C_Trans(i, j) και C_Final_grade(i, j) ευθύνονται για την ανανέωση των τελικών βαθμών των μελών της κοινότητας.

Εκτός όμως από τα γεγονότα ολοκλήρωσης συναλλαγής, οι τελικοί βαθμοί υπόληψης των οντοτήτων του συστήματος μας μεταβάλλονται όταν ένας χρήστης εισέλθει ή διαγραφεί από κοινότητα. Σε αυτή την περίπτωση, το πρώτο βήμα είναι η ανανέωση του τελικού βαθμού

υπόληψης της κοινότητας στην οποία εισήλθε ή από την οποία διαγράφηκε ο χρήστης, σύμφωνα με τον τύπο:

$$\text{C\_FinalGrade(i,j)} = w \cdot C\_grade(i,j) + (1-w) \cdot \Big(\sum_{\substack{1 \le l \le d \\ user\_l \in community\_i}} U\_grade(l,j) \Big) / m$$

Όπου $$w = \frac{C\_Trans(i,j)}{MeanU\_Trans(i,j) + C\_Trans(i,j)}$$

Και $$\text{MeanU\_Trans(i, j)} = \sum_{\substack{1 \le l \le d \\ user\_l \in community\_i}} U\_Trans(l,j) / m$$

Είναι προφανές ότι αν ο χρήστης που εισήλθε ή διαγράφηκε από κοινότητα δεν ήταν «ενεργός», τότε δεν θα υπάρξει καμία μεταβολή στους τελικούς βαθμούς υπόληψης των κοινοτήτων στις οποίες ανήκει ο χρήστης. Αντίθετα, αν ο χρήστης που εισήλθε ή διαγράφηκε από κοινότητα ήταν «ενεργός» χρήστης, τότε θα έχουμε μεταβολή του πλήθους των ενεργών χρηστών m καθώς και της δεύτερης συνιστώσας του τελικού βαθμού υπόληψης της κοινότητας. Η είσοδος λοιπόν ενός «ενεργού» χρήστη σε μία κοινότητα οδηγεί στον επαναϋπολογισμό του παράγοντα w , του παράγοντα $\sum_{\substack{1 \le l \le d \\ user\_l \in community\_i}} U\_grade(l,j) / m$ και επομένως του τελικού βαθμού υπόληψης κοινότητας για την κοινότητα στην οποία εισήλθε. Η διαγραφή ενός «ενεργού» χρήστη από κοινότητα συνεπάγεται τον επαναϋπολογισμό του παράγοντα w , του παράγοντα $\sum_{\substack{1 \le l \le d \\ user\_l \in community\_i}} U\_grade(l,j) / m$ και επομένως του τελικού βαθμού υπόληψης κοινότητας για την κοινότητα από την οποία διαγράφηκε. Όμως, εκτός από τον τελικό βαθμό υπόληψης της κοινότητας που επηρεάζεται, ανανέωση έχουν και οι τιμές που αφορούν την δραστηριότητα της κοινότητας και κρατούνται στον πίνακα vactivity. Συγκεκριμένα, ανακτούμε την εγγραφή του πίνακα vactivity που αναφέρεται στην κοινότητα που μελετάμε. Αν ο χρήστης εισήλθε σε αυτή την κοινότητα, το πλήθος συναλλαγών του επηρεάζει τον μέσο πλήθος συναλλαγών ανά χρήστη της κοινότητας και προστίθεται και στις συνολικές συναλλαγές των μελών της κοινότητας. Αν ο χρήστης διαγράφηκε από την κοινότητα, θα επηρεαστεί το μέσο πλήθος συναλλαγών ανά χρήστη της κοινότητας και από το συνολικό πλήθος συναλλαγών των μελών της κοινότητας θα αφαιρεθούν οι συνολικές συναλλαγές του χρήστη.

Η ανανέωση του τελικού βαθμού μίας κοινότητας προκαλεί μεταβολή των τελικών βαθμών των χρηστών - μελών της κοινότητας. Αφού λοιπόν εφαρμόζουμε τις παραπάνω ανανεώσεις,

θα επαναϋπολογίσουμε τον τελικό βαθμό υπόληψης των μελών της κοινότητας που επηρεάστηκε σύμφωνα με τον παρακάτω τύπο.

$$\text{U\_FinalGrade(i, j)} = w \cdot U\_grade(i, j) + (1 - w) \cdot \Big(\sum_{\substack{1 \le l \le v \\ user\_i \in community\_l}} C\_FinalGrade(l, j) \Big) / m$$

Σημειώνουμε ότι αν ο χρήστης εισήλθε στην κοινότητα, τότε η εφαρμογή του παραπάνω υπολογισμού συμπεριλαμβάνει και τον χρήστη αυτό. Αν ωστόσο, ο χρήστης διαγράφηκε από την κοινότητα, τότε και πάλι εφαρμόζουμε τον παραπάνω τύπο για τον υπολογισμό του τελικού βαθμού υπόληψης του χρήστη με δεδομένο ότι τώρα το σύνολο των κοινοτήτων στις οποίες ανήκει ο χρήστης έχει μεταβληθεί.

5.1.3 Διαχείριση της συναλλαγής

Αναπόσπαστο κομμάτι του συστήματος μας είναι ο κύκλος που επιτελεί μια συναλλαγή από τη στιγμή που κάποιος χρήστης βρίσκει κάτι που θέλει να ανταλλάξει, δωρίσει ή μοιράσει μέχρι την φυσική πράξη μεταξύ των συναλλασσομένων και την αξιολόγηση της. Θεωρώντας ότι υπάρχει κάποιο αγαθό στο σύστημα που ζητείται ή προσφέρεται, η συναλλαγή αρχίζει με την εκδήλωση της από μέρους ενός χρήστη Α και ολοκληρώνεται με την αξιολόγηση της και από τις δύο οντότητες.

Αρχικά για να καταχωρίσει ένας χρήστης ή μια κοινότητα κάποιο αγαθό πρέπει κατά την εισαγωγή του να ορίσει αν το προσφέρει ή το ζητά και με ποιον τρόπο επιθυμεί να γίνει. Έτσι στη βάση του συστήματος μας υπάρχει μια λίστα από αντικείμενα που ζητούνται και άλλη μια με εκείνα που προσφέρονται. Για το σκοπό αυτό υλοποιήσαμε δύο πίνακες, τον *demands* και τον *offers*, των οποίων οι εγγραφές περιλαμβάνουν αντίστοιχα τα αγαθά που ζητούνται και τα αγαθά που προσφέρονται. Με αυτόν τον τρόπο κάθε αγαθό μπορεί είτε να είναι ενεργό, δηλαδή να βρίσκεται σε κάποιον από τους δύο πίνακες, είτε να είναι απλά στην κατοχή μίας οντότητας αλλά να μην διαπραγματεύεται. Για το λόγο αυτό ήταν αναγκαίες οι παρακάτω δηλώσεις ώστε να ολοκληρωθούν οι συσχετίσεις μεταξύ των αγαθών και των λιστών :

- Για την λίστα με τα αγαθά που προσφέρονται

```
class Offer < ActiveRecord::Base
  belongs_to :commodity
  TRANSACTION_TYPES = [
      ["Exchange", "exchange"],
      ["Gift", "gift"],
      ["Share", "share"]
      ]
```

```
  validates_inclusion_of :type_of_transaction, :in =>
TRANSACTION_TYPES.map {|disp, value| value}

end
```

- Για την λίστα με τα αγαθά που ζητούνται

```
class Demand < ActiveRecord::Base
  belongs_to :commodity
  TRANSACTION_TYPES = [
      ["Exchange", "exchange"],
      ["Gift", "gift"],
      ["Share", "share"]
      ]

  validates_inclusion_of :type_of_transaction, :in =>
TRANSACTION_TYPES.map {|disp, value| value}
end
```

Με τον τρόπο αυτό μπορούμε να δούμε αν κάποιο αγαθό ζητείται , προσφέρεται ή τίποτα από τα δύο. Βλέπουμε ακόμα τον φυσικό περιορισμό που επιβάλουμε στον τύπο της συναλλαγής η οποία πρέπει να είναι μια εκ των δωρεά, κοινοχρησία ή ανταλλαγή.

Η διαδικασία κάθε συναλλαγής συνίσταται σε τρία βασικά βήματα.

- Στην αρχή πρέπει κάποια οντότητα να ξεκινήσει την συναλλαγή εκφράζοντας την επιθυμία της να αποκτήσει ή να προσφέρει ένα αγαθό που υπάρχει αντίστοιχα σε μία από τις λίστες που αναφέραμε νωρίτερα.
- Στη συνέχεια το άλλο μέλος πρέπει να αποδεχτεί την καλύτερη πρόταση, πιθανών ανάμεσα σε πολλές για το συγκεκριμένο αγαθό.
- Τέλος αφού πραγματοποιηθεί η συναλλαγή, κάθε μέλος πρέπει να την βαθμολογήσει ανάλογα με το πώς εξελίχθηκε.

Θα περιγράψουμε αναλυτικά την εξέλιξη μίας συναλλαγής, με βάση αν το αγαθό για το οποίο ενδιαφέρεται ένας χρήστης προσφέρεται ή ζητείται και τι τύπου συναλλαγή είναι.

1) Ο χρήστης Α ενδιαφέρεται για κάποιο αγαθό που προσφέρει ο χρήστης Β.
   a) Το αγαθό προσφέρεται για δώρο.
      i) Ο χρήστης Α δηλώνει μέσα από μια φόρμα αν επιθυμεί να το αποκτήσει ως απλό μέλος ή ως διαχειριστής μίας κοινότητας.
      ii) Ο χρήστης Β ειδοποιείται με email για την πρόταση του χρήστη Α
      iii) Ο χρήστης Β επιλέγει μία από τις διαθέσιμες προτάσεις που υπάρχουν για το αγαθό
      iv) Ο χρήστης Α ενημερώνεται με email για την επιλογή του χρήστη Β

v) Αφού γίνει η φυσική συναλλαγή, ο χρήστης Α δίνει ένα βαθμό αξιοπιστίας στον χρήστη Β για τη συγκεκριμένη συναλλαγή

b) Το αγαθό προσφέρεται για κοινοχρησία.

i) Ο χρήστης Α δηλώνει μέσα από μια φόρμα ότι επιθυμεί να το μοιραστεί.

ii) Ο χρήστης Β ειδοποιείται με email για την πρόταση του χρήστη Α

iii) Ο χρήστης Β επιλέγει μία από τις διαθέσιμες προτάσεις που υπάρχουν για το αγαθό

iv) Ο χρήστης Α ενημερώνεται με email για την επιλογή του χρήστη Β

v) Αφού γίνει η φυσική συναλλαγή, ο χρήστης Α δίνει ένα βαθμό αξιοπιστίας στον χρήστη Β για τη συγκεκριμένη συναλλαγή

c) Το αγαθό 1 προσφέρεται για ανταλλαγή

i) Ο χρήστης Α δηλώνει μέσα από μια φόρμα αν επιθυμεί να το αποκτήσει ως απλό μέλος ή ως διαχειριστής μίας κοινότητας καθώς και το αγαθό 2 που έχει στην κατοχή του και θέλει να ανταλλάξει.

ii) Ο χρήστης Β ειδοποιείται με email για την πρόταση του Α.

iii) Ο χρήστης Β επιλέγει μία από τις διαθέσιμες προτάσεις που υπάρχουν για το αγαθό 1.

iv) Ο χρήστης Α ενημερώνεται με email για την επιλογή του χρήστη Β

v) Αφού γίνει η φυσική συναλλαγή, ο χρήστης Α δίνει ένα βαθμό αξιοπιστίας στον χρήστη Β για τη συγκεκριμένη συναλλαγή καθώς και ο χρήστης Β στον χρήστη Α.

2) Ο χρήστης Α ενδιαφέρεται για κάποιο αγαθό 1 που ζητάει ο χρήστης Β.

a) Το αγαθό 1 ζητείται για δώρο.

i) Ο χρήστης Α δηλώνει μέσα από μια φόρμα αν επιθυμεί να το χαρίσει ως απλό μέλος ή ως διαχειριστής μίας κοινότητας όπως επίσης επιλέγει το αγαθό 2 που έχει στην κατοχή του και θέλει να δωρίσει.

ii) Ο χρήστης Β ειδοποιείται με email για την πρόταση του χρήστη Α

iii) Ο χρήστης Β επιλέγει μία από τις διαθέσιμες προτάσεις που υπάρχουν για το αγαθό

iv) Ο χρήστης Α ενημερώνεται με email για την επιλογή του χρήστη Β

v) Αφού γίνει η φυσική συναλλαγή, ο χρήστης Β δίνει ένα βαθμό αξιοπιστίας στον χρήστη Α για τη συγκεκριμένη συναλλαγή.

b) Το αγαθό 1 ζητείται για κοινοχρησία.

i) Ο χρήστης Α επιλέγει το αγαθό 2 που έχει στην κατοχή του και θέλει να μοιραστεί.

ii) Ο χρήστης Β ειδοποιείται με email για την πρόταση του χρήστη Α

iii) Ο χρήστης Β επιλέγει μία από τις διαθέσιμες προτάσεις που υπάρχουν για το αγαθό 1.

iv) Ο χρήστης Α ενημερώνεται με email για την επιλογή του χρήστη Β.

v) Αφού γίνει η φυσική συναλλαγή, ο χρήστης Β δίνει ένα βαθμό αξιοπιστίας στον χρήστη Α για τη συγκεκριμένη συναλλαγή

c) Το αγαθό 1 ζητείται για ανταλλαγή

i) Ο χρήστης Α δηλώνει μέσα από μια φόρμα αν επιθυμεί να συμμετέχει ως απλό μέλος ή ως διαχειριστής μίας κοινότητας καθώς και το αγαθό 2 που έχει στην κατοχή του και θέλει να ανταλλάξει.

ii) Ο χρήστης Β ειδοποιείται με email για την πρόταση του Α.

iii) Ο χρήστης Β επιλέγει μία από τις διαθέσιμες προτάσεις που υπάρχουν για το αγαθό 1 και επιλέγει το αγαθό 3 που έχει στην κατοχή του και θέλει να ανταλλάξει.

iv) Ο χρήστης Α ενημερώνεται με email για την επιλογή του χρήστη Β.

v) Ο χρήστης Α εγκρίνει ή απορρίπτει την πρόταση του χρήστη Β.

vi) Αφού γίνει η φυσική συναλλαγή, ο χρήστης Α δίνει ένα βαθμό αξιοπιστίας στον χρήστη Β για τη συγκεκριμένη συναλλαγή καθώς και ο χρήστης Β στον χρήστη Α.

Για να μπορέσουμε να διαχειριστούμε τη συναλλαγή, δημιουργήσαμε την κλάση/αντικείμενο *Xaction* η οποία χρησιμοποιείται σε όλη τη διάρκεια που είναι ενεργή μια συναλλαγή. Με τον όρο "ενεργή", εννοούμε κάθε συναλλαγή που βρίσκεται ένα στάδιο πριν την φυσική επαφή των δυο μελών. Για να μπορέσουμε να κρατήσουμε την πληροφορία που μας χρειάζεται, αποθηκεύουμε σε κάθε συναλλαγή τις δύο οντότητες που παίρνουν μέρος καθώς και τα αγαθά με τα οποία συμμετέχουν σε αυτή όπως βλέπουμε και από το παρακάτω σχήμα :

```
class CreateXactions < ActiveRecord::Migration
  def self.up
    create_table :xactions do |t|
      t.column :commodity1_id, :integer, :references=>:commodities
      t.column :commodity2_id, :integer, :references=>:commodities
      t.column :entity1_id, :integer, :references=>:entities
      t.column :entity2_id, :integer, :references=>:entities
      t.column :type_of_transaction, :string
      t.column :date, :datetime
      t.column :state, :char
    end
```

```
  end

  def self.down
    drop_table :xactions
  end
end
```

Επιβάλλουμε κάποιες συσχετίσεις ώστε κάθε αγαθό ή οντότητα της συναλλαγής, να αντιστοιχίζεται άμεσα με το κατάλληλο αντικείμενο του συστήματός μας.

```
class Xaction < ActiveRecord::Base
  belongs_to :commodity1, :class_name=>"Commodity",
:foreign_key=>"commodity1_id"
  belongs_to :commodity2, :class_name=>"Commodity",
:foreign_key=>"commodity2_id"
  belongs_to :entity1, :class_name=>"Entity",
:foreign_key=>"entity1_id"
  belongs_to :entity2, :class_name=>"Entity",
:foreign_key=>"entity2_id"
end
```

Από το σχήμα της κλάσης που χρησιμοποιούμε για την συναλλαγή, προκύπτει ένα ακόμα πεδίο, εκείνο της κατάστασης στην οποία μπορεί να βρίσκεται κάθε στιγμή. Ανάλογα με το εάν έχει αποδεχθεί το δεύτερο μέλος την πρόταση και αν έχει πραγματοποιηθεί η βαθμολόγηση, μια συναλλαγή μπορεί να είναι σε κατάσταση

- "**pending**" ("p") όταν εκκρεμεί η αποδοχή της αρχικής πρότασης.
- "**executed**" ("e") όταν ο χρήστης έχει αποδεχθεί κάποια πρόταση.
- "**semi - executed"** ("s") στην περίπτωση της ανταλλαγής με αγαθό που ζητείται, όταν εκκρεμεί η αποδοχή της δεύτερης πρότασης.
- "**half - rated**" ("h") όταν πρόκειται για ανταλλαγή που έχει βαθμολογήσει μόνο το ένα από τα δύο μέλη.
- "**rated**" ("r") όταν πλέον έχει ολοκληρωθεί η συναλλαγή με την πλήρη βαθμολόγηση.

Να σημειώσουμε εδώ πως όλες οι συναλλαγές που γίνονται στο σύστημα μας, δεν δεσμεύουν τα αγαθά τα οποίο συμμετέχουν κάθε φορά, μέχρι να αποδεχθούν και τα δύο μέρη την πρόταση. Δηλαδή ένα αγαθό είναι ανοιχτό σε προσφορές μέχρι ο χρήστης να αποδεχθεί μια εξ αυτών. Μόνο τότε παύει να διαπραγματεύεται οπότε και βγαίνει από τις λίστες προσφοράς ή ζήτησης, ανάλογα με το πού είχε αρχικά καταχωρηθεί.

Μετά τη συναλλαγή και ανάλογα με το είδος της, το αγαθό ή τα αγαθά που συμμετείχαν αλλάζουν ιδιοκτησία. Πλέον, στο σύστημα είναι καταχωρημένα στον καινούργιο ιδιοκτήτη τους ο οποίος μπορεί να τα επαναφέρει για συναλλαγή όποτε θέλει. Εξαίρεση αποτελεί η

περίπτωση της κοινοχρησίας, στην οποία ο ιδιοκτήτης παραμένει η κοινότητα η οποία μοιράζεται και το αγαθό.

Η διαδικασία αποστολής των email γίνεται σχεδόν αυτοματοποιημένα στην Ruby με τη βοήθεια των Rails. Χρησιμοποιούμε την κλάση *TransactionMailer* στην οποία δηλώνουμε τα απαραίτητα πεδία για την αποστολή του email όπως τον αποστολέα, τον παραλήπτη, το θέμα καθώς και το περιεχόμενο. Για κάθε διαφορετικό είδος ειδοποίησης μέσω email έχουμε και ένα διαφορετικό πρότυπο στα views της συγκεκριμένης κλάσης, για παράδειγμα το *notice.rhtml* που χρησιμοποιούμε για την ειδοποίηση του χρήστη για μια πρόταση. Έτσι με μία κλήση του τύπου

```
TransactionMailer.deliver_notice(@transaction)
```

στέλνουμε ένα email τύπου *notice* με παραμέτρους που ορίζονται στην μεταβλητή *@transaction*.

Τέλος, αφού ολοκληρωθεί η συναλλαγή και η βαθμολόγηση, το αντικείμενο *Xaction* που κρατούσε τα στοιχεία της διαγράφεται και η συναλλαγή περνάει σε ένα ιστορικό. Εκεί κρατάμε όλες τις πληροφορίες που σχετίζονται με τη συναλλαγή όπως φαίνεται και από το σχήμα της βάσης στους πίνακες *gifts, sharings* και *swaps*. Στα αντίστοιχα μοντέλα προσθέσαμε τις παρακάτω συσχετίσεις

```
class Gift < ActiveRecord::Base
  belongs_to :commodity
  belongs_to :entity
end
```

```
class Sharing < ActiveRecord::Base
  belongs_to :commodity
  belongs_to :entity
end
```

```
class Swap < ActiveRecord::Base
  belongs_to :commodity1, :class_name=>"Commodity",
:foreign_key=>"commodity1_id"
  belongs_to :commodity2, :class_name=>"Commodity",
:foreign_key=>"commodity2_id"
  belongs_to :entity1, :class_name=>"Entity",
:foreign_key=>"entity1_id"
  belongs_to :entity2, :class_name=>"Entity",
:foreign_key=>"entity2_id"
end
```

Αυτές οι συσχετίσεις μας επιτρέπουν να αντιστοιχίζουμε το κάθε αγαθό και οντότητα που υπάρχει στο σύστημά μας με αυτά που κρατάμε στο ιστορικό.

5.2 Πλατφόρμες και προγραμματιστικά εργαλεία

Στη συνέχεια θα δώσουμε τα χαρακτηριστικά των προγραμματιστικών εργαλείων που χρησιμοποιήσαμε για να υλοποιήσουμε το έργο μας. Η πρώτη σοβαρή επιλογή που έπρεπε να κάνουμε ήταν η γλώσσα στην οποία θα γράφαμε τον κώδικά μας. Αφού ψάξαμε ποιες προγραμματιστικές γλώσσες είναι κατάλληλες για ανάπτυξη δικτυακών εφαρμογών καθώς και τι πλατφόρμα ανάπτυξης προσφέρουν, καταλήξαμε σε τρείς υποψήφιες, καθεμιά από τις οποίες είχε και ένα διαφορετικό χαρακτηριστικό. Η τεχνολογία Active Server Pages (ASP) της Microsoft, η γλώσσα PHP που είναι η πιο ευρέως διαδεδομένη για την ανάπτυξη τέτοιων εφαρμογών, η Java με τις Java Servlet Pages (JSP) και η Ruby ήταν αυτές που προκρίναμε ως κατάλληλες. Η τελική επιλογή της Ruby έγινε σε συνδυασμό με την πλατφόρμα ανάπτυξης Rails, ο οποίος προσφέρει ταχεία συγγραφή κώδικα και αυτοματοποιημένες διαδικασίες καθώς και πολύ καλή ενσωμάτωση της βάσης δεδομένων. Τα χαρακτηριστικά αυτά καθώς και το ότι αποτελεί μια σχετικά καινούργια και αναπτυσσόμενη ιδέα στον προγραμματισμό δικτυακών εφαρμογών μας έκαναν να απορρίψουμε τις υπόλοιπες λύσεις.

Για τη βάση δεδομένων μας, οι λύσεις που είχαμε ως προς το εργαλείο ανάπτυξής της ήταν δύο, ο SQL Server της Microsoft και ο MySQL. Επειδή οι απαιτήσεις της εφαρμογής μας δεν ήταν ακραίες προτιμήσαμε τον MySQL αφού αποτελεί και προϊόν ελεύθερου κώδικα σε αντίθεση με τον SQL Server που θα χρειαζόταν οικονομική υποστήριξη για εξειδικευμένα χαρακτηριστικά αλλά και για τη συντήρησή του.

Τέλος οι επιλογές που είχαμε για τον εξυπηρετητή δικτύου ήταν αρκετές μεταξύ αυτών ο πιο ευρέως διαδεδομένος, ο Apache και ο WEBrick. Ήταν φυσικό να επιλέξουμε τον δεύτερο αφού αυτός είναι γραμμένος για την γλώσσα προγραμματισμού που επιλέξαμε, την Ruby.

- Ruby on Rails

Η Ruby είναι μία δυναμική, ανοιχτού κώδικα, γλώσσα προγραμματισμού, με έμφαση στην απλότητα και την αποδοτικότητα. Έχει ένα "κομψό" συντακτικό το οποίο διαβάζεται με φυσικότητα και γράφεται με ευκολία. Η Ruby είναι αντικειμενοστραφής και εμπνευσμένη από την Perl και από τα αντικειμενοστραφή χαρακτηριστικά της Smalltalk, αλλά και με επιρροές από άλλες, όπως η Python, η Lisp, η Dylan και η CLU. Όπως χαρακτηριστικά αναφέρει ο δημιουργός της, Yukihiro Matsumoto που την δημοσίευσε το 1995, η Ruby είναι απλή στην εμφάνιση, αλλά πολύπλοκη στο εσωτερικό της, όπως το ανθρώπινο σώμα.

Ο ακριβής χαρακτηρισμός της γλώσσας δεν είναι "απλή", αλλά "φυσική". Αυτό, όχι μόνο δεν μειώνει την αποτελεσματικότητα και τις δυνατότητες της, αλλά αντίθετα είναι ένα από τα πολλά στοιχεία, που την κάνουν μια πραγματικά ευχάριστη, αποτελεσματική και διαφορετική γλώσσα προγραμματισμού. Τα κυριότερα χαρακτηριστικά της είναι

- Interpreted
- Αντικειμενοστραφής
- Χωρίς τύπους
- Συλλογή απορριμμάτων
- Εύκολο και καθαρό συντακτικό
- Φορητότητα

Η Ruby on Rails είναι μία πλατφόρμα (framework) ανάπτυξης web εφαρμογών βασισμένο στην Ruby. Υποστηρίζει πλήρως και εγγενώς όλες τις νέες τεχνολογίες και τάσεις:

- MVC αρχιτεκτονική
- ORM (αντικειμενοσχεσιακή αντιστοίχηση)
- AJAX λειτουργικότητα
- Web Services
- Templating για το layout των σελίδων χρησιμοποιώντας εμφωλευμένο Ruby κώδικα σε html, CSS, Javascript κ.λ.π.

Τα Rails βασίζονται σε δυο αξιώματα :

- **Μην επαναλαμβάνεις τον εαυτό σου! (DRY – Don't Repeat Yourself)**
- **Προτυποποίηση αντί διαμόρφωσης! (convention over configuration)**

Η πληροφορία δεν θα πρέπει να υπάρχει δύο (ή και περισσότερες) φορές γιατί διπλότυπη πληροφορία επιφέρει δυσκολία στις αλλαγές, μειωμένη διαφάνεια και αυξημένη πιθανότητα για «ασυνέπειες» στον κώδικα.

Χρησιμοποιώντας την τεχνική αυτή επιτυχώς, σημαίνει ότι η αλλαγή ενός συγκεκριμένου στοιχείου δεν επηρεάζει τα υπόλοιπα, λογικά ασύνδετα στοιχεία του συστήματος. Επιπλέον, στοιχεία που συνδέονται λογικά, αλλάζουν ομοιογενώς και απόλυτα προβλέψιμα και έτσι είναι συγχρονισμένα

Βάση του δεύτερου αξιώματος ο προγραμματιστής χρειάζεται να ορίσει μόνο τις παραμέτρους που δεν είναι δυνατόν να «προτυποποιηθούν», ας πούμε μία κλάση Post μπορεί αυτομάτως να συσχετιστεί με έναν πίνακα posts. Έτσι, οι «συμβάσεις» που χρησιμοποιούνται από το Rails μπορούν να μειώσουν αισθητά τον κώδικα που χρειάζεται να γραφεί. Φυσικά, οι συμβάσεις αυτές μπορούν να αλλάξουν όταν αυτό είναι απαραίτητο ή επιθυμητό.

Η φιλοσοφία της πλατφόρμας πλαισιώνεται ακόμα από την ιδέα του agile που σημαίνει ευκίνητος, σβέλτος. Η μεθοδολογία αυτή είναι μία οικογένεια μεθόδων για την διαδικασία της ανάπτυξης προϊόντων (γενικά) που έχει προταθεί, έχοντας όμως κατά νου, κυρίως το λογισμικό. Ουσιαστικά μιλάμε για την δημιουργία λογισμικού με έναν πιο «ελαφρύ», «γρήγορο» και ανθρωποκεντρικό τρόπο.

Το Rails χρησιμοποιεί την MVC αρχιτεκτονική (Model-View-Controller):

Model: Τα δεδομένα μας ή αλλιώς η επιχειρησιακή λογική της εφαρμογής μας

View: Το επίπεδο παρουσίασης, δηλαδή το πώς εμφανίζονται τα αποτελέσματα - δεδομένα

Controller: Η διεπαφή με τον χρήστη και ουσιαστικά η λειτουργικότητα της εφαρμογής

Άλλα στοιχεία της πλατφόρμας είναι το Active Record που αποτελεί μια αντικειμενοσχεσιακή αντιστοίχηση (object-relation mapping – ORM). Είναι ουσιαστικά η σύνδεση ανάμεσα στην βάση δεδομένων και την επιχειρησιακή λογική του προγράμματος, δηλαδή τα μοντέλα/κλάσεις. Ακόμα υπάρχει το Action Pack που ενσωματώνει πρακτικά όλη τη λειτουργικότητα της εφαρμογής από την πλευρά του χρήστη. Εδώ υλοποιείται τόσο το στρώμα παρουσίασης (View) όσο και ο controller της MVC αρχιτεκτονικής. Το κομμάτι του controller χειρίζεται τις εισερχόμενες αιτήσεις από τον browser του χρήστη και τις δρομολογεί στην κατάλληλη μέθοδο μίας κλάσης controller. Το κομμάτι της παρουσίασης «συνθέτει» την απάντηση που θα σταλθεί πίσω στον browser του χρήστη (π.χ. σε html, xml, κ.λ.π.).

Επίσης δομικά συστατικά της πλατφόρμας αποτελούν το Prototype που υλοποιεί την AJAX λειτουργικότητα του site, όπως drag-and-drop, οπτικά εφέ κλπ., ο Action Mailer που χειρίζεται την λήψη και την αποστολή emails και το Action Web Service που επιτρέπει με ευκολία να προσθέσουμε ένα web service στην εφαρμογή και υποστηρίζει όλες τις γνωστές και διαδεδομένες τεχνολογίες, όπως SOAP, REST, XML-RPC, WSDL.

Η ROR παρέχει επίσης κάποια εργαλεία για την παραγωγή έτοιμου κώδικα, τα Scripts και τους Generators. Έτσι γλιτώνουμε από κάποιες τυπικές και βαρετές διαδικασίες με αποτέλεσμα να συγκεντρωνόμαστε στο πραγματικό ζητούμενο, δηλαδή την δημιουργία

της εφαρμογής μας. Ακόμα μας δίνει τη δυνατότητα να έχουμε ταυτόχρονα τρία περιβάλλοντα εργασίας, Development, Production και Testing. Το καθένα χρησιμοποιεί την δικιά του έκδοση κώδικα και την δικιά του βάση. Έτσι, στο development, βλέπουμε δυναμικά τις αλλαγές που κάνουμε και στο testing βάζουμε κάποιον δοκιμαστικό κώδικα που προσπαθεί να «προβλέψει» την συμπεριφορά της εφαρμογής μας. Όταν αποφασίσουμε ότι θέλουμε να δημοσιεύσουμε την καινούρια έκδοση αυτό γίνεται με μία εντολή. Φυσικά, καθ' όλη την διάρκεια αυτή, ο χρήστης εξακολουθεί να «βλέπει» την έκδοση που δουλεύει σωστά.

Από τα πιο σημαντικά εργαλεία που παρέχει η πλατφόρμα είναι τα Migrations με τα οποία μπορούμε με ασφάλεια να επιστρέψουμε στην προηγούμενη κατάσταση που βρισκόταν η βάση εάν κάτι δεν πάει καλά. Αυτό συμβαίνει διότι μία εφαρμογή σε πραγματικές συνθήκες δεν παραμένει σταθερή και πρέπει συνεχώς να εξελίσσεται η βάση δεδομένων (αφού αντικατοπτρίζει την επιχειρησιακή λογική).

Τέλος το Rails υποστηρίζει την χρήση plugins το οποίο μπορεί να τροποποιεί ή να επεκτείνει μία λειτουργία του framework. Με τα plugins οι προγραμματιστές μπορούν να μοιράζονται ιδέες «αιχμής» χωρίς να «πειράζουν» τον κυρίως κώδικα του rails. Ακόμα παρέχουν μία κατανεμημένη αρχιτεκτονική που επιτρέπει σε πακέτα κώδικα να ανανεώνονται ανεξάρτητα καθώς και ένα τρόπο ώστε οι προγραμματιστές να παρέχουν νέες δυνατότητες και λειτουργίες γρήγορα και χωρίς να επηρεάζουν τον υπάρχοντα κώδικα

- MySQL

Η MySQL είναι ένα σχεσιακό σύστημα διαχείρισης δεδομένων ανοιχτού κώδικα που δουλεύει πάνω στο πρότυπο εξυπηρετητή-πελάτη. Τα βασικά χαρακτηριστικά της είναι η ταχύτητα, η αξιοπιστία και ο εύκολος χειρισμός της. Τα βασικά εργαλεία που χρησιμοποιήσαμε για τον χειρισμό της βάσης μας ήταν ο MySQL Administrator και ο MysSQL Query Browser. Ο MySQL Query Browser είναι ένα γραφικό εργαλείο, το οποίο παρέχεται από την MySQL AB, για την δημιουργία, εκτέλεση και βελτιστοποίηση αιτημάτων σε γραφικό περιβάλλον. Ενώ ο MySQL Administrator έχει σχεδιαστεί για την διαχείριση ενός MySQL εξυπηρετητή, ο MySQL Query Browser έχει σχεδιαστεί για να βοηθήσει στην δημιουργία αιτημάτων και ανάλυσης δεδομένων, τα οποία είναι αποθηκευμένα στην MySQL βάση δεδομένων. Στην παρούσα εργασία χρησιμοποιήθηκε η έκδοση MySQL 5.1

- WEBrick server

Για την ανάπτυξη της εφαρμογής μας, χρησιμοποιήσαμε τον WEBrick server, έναν pure Ruby web server. Αυτό συμβαίνει διότι παρέχει μερικές σημαντικές ευκολίες στο στάδιο της παραγωγής κώδικα. Βέβαια όταν περάσουμε στο στάδιο της ανάπτυξης της εφαρμογής θα εμπιστευθούμε εμπορικούς και ευρέως χρησιμοποιημένους εξυπηρετητές δικτύου όπως ο Apache ο οποίος επιτρέπει την ταυτόχρονη εξυπηρέτηση πολλών αιτήσεων.

Με τον WEBrick δεν χρειάστηκε να επανεκκινήσουμε την εφαρμογή μας καθώς προσθέταμε κώδικα σε αυτήν. Και όμως κάθε αλλαγή που κάναμε ήταν διαθέσιμη κάθε φορά που αιτούμασταν πρόσβαση στην εφαρμογή μέσω ενός φυλλομετρητή. Αυτό συμβαίνει γιατί στο στάδιο της παραγωγής (σε αντίθεση με τα στάδια των δοκιμών ή της ανάπτυξης, επαναφορτώνει αυτόματα τα αρχεία κώδικα για κάθε νέα αίτηση που θα δεχτεί. Με αυτόν τον τρόπο, όταν επεξεργαζόμαστε την εφαρμογή μας, ο αποστολέας μας εξασφαλίζει ότι χρησιμοποιεί τις πιο πρόσφατες αλλαγές. Ωστόσο, η ευελιξία αυτή έρχεται με ένα κόστος αφού προκαλεί μια σύντομη διακοπή από τη στιγμή που εισαχθεί το URL μέχρι να απαντήσει.

6

Έλεγχος

6.1 Μεθοδολογία ελέγχου

Η υλοποίηση του συστήματός μας συνοδεύτηκε από μία σειρά ελέγχων για την διαπίστωση της απόκρισής της στις απαιτήσεις που έχουν εξαρχής προδιαγραφεί καθώς και της άρτιας, σταθερής και πλήρους λειτουργικότητας, που παρέχει η εφαρμογή μας. Στην ενότητα αυτή θα παρουσιάσουμε την μεθοδολογία ελέγχου που εφαρμόσαμε και θα καλύψουμε τα use cases που έχουν αναφερθεί στο έγγραφο προδιαγραφής απαιτήσεων παρουσιάζοντας τα σενάρια ελέγχου για καθένα από αυτά. Θα ακολουθήσουν υποενότητες οι οποίες θα έχουν τίτλο σεναρίων καθώς και σύντομη περιγραφή του κάθε σεναρίου.

Η Ruby on Rails υποστηρίζει τον έλεγχο της υλοποίησης της εφαρμογής μας, παρακολουθώντας την από την αρχή, και δομώντας την υποδομή για τον έλεγχο συγχρόνως με την δημιουργία της εφαρμογής. Συγκεκριμένα, ο σκελετός της εφαρμογής όπως παρέχεται από το framework, περιλαμβάνει ειδικό πεδίο για τον έλεγχο. Όπως για την εφαρμογή μας υπάρχει ειδικός φάκελος (app) εντός του φακέλου του workspace μας όπου εντάσσονται τα τρία component της εφαρμογής (Model – View – Controller), έτσι και για τον έλεγχο υπάρχει ειδικός φάκελος (test). Κατά σύμβαση, η Ruby on Rails ονομάζει τους ελέγχους που αφορούν το μοντέλο της εφαρμογής unit tests και τους ελέγχους που αφορούν τους controllers της εφαρμογής functional tests. Υπάρχουν λοιπόν οι φάκελοι unit και functional εντός του φακέλου test, οι οποίοι διατηρούν τα αρχεία για τον έλεγχο των κλάσεων του μοντέλου και

των controllers αντίστοιχα. Κάθε κλάση του μοντέλου έχει αντίστοιχη κλάση ελέγχου στον φάκελο unit ενώ κάθε κλάση του controller έχει αντίστοιχη κλάση ελέγχου στον φάκελο functional. Οι μέθοδοι αυτών των κλάσεων περιέχουν τον κώδικα που υλοποιεί ελέγχουν για το στοιχείο του συστήματός μας στο οποίο αντιστοιχεί η κλάση. Επιπλέον, η Ruby on Rails μεριμνεί για την δημιουργία δεδομένων ελέγχου ανεξάρτητων από των δεδομένων που χρησιμοποιήθηκαν από την εφαρμογή μας κατά την ανάπτυξη του κώδικα καθώς και ανεξάρτητα από τα δεδομένα που θα χρησιμοποιηθούν κατά την φάση της εγκατάστασης της εφαρμογής σε πραγματικό περιβάλλον. Συγκεκριμένα, το framework υποστηρίζει την ύπαρξη τριών βάσεων δεδομένων για το σύστημά μας. Οι βάσεις αυτές έχουν κατά σύμβαση τα εξής ονόματα :

- <όνομα εφαρμογής>_ development
- <όνομα εφαρμογής>_ test
- <όνομα εφαρμογής>_ production

Οι τρεις αυτές βάσεις δεδομένων έχουν ακριβώς το ίδιο σχήμα και, όπως υποδεικνύουν και τα ονόματά τους, η πρώτη βάση δεδομένων αφορά τα δεδομένα που χρησιμοποιούνται κατά την ανάπτυξη του κώδικα, η δεύτερη τα δεδομένα ελέγχου και η τρίτη τα πραγματικά δεδομένα , μετά την εγκατάσταση του συστήματος σε πραγματικό περιβάλλον.

Η Ruby on Rails, λοιπόν, παρέχει στον σκελετό για την διαμόρφωση των ελέγχων και αφήνει στον προγραμματιστή μόνο την επιμέλεια της δημιουργίας του σωστού ελέγχου.

6.1.1 Έλεγχοι Μοντέλου Συστήματος – Unit Testing

Οι έλεγχοι που αφορούν τις κλάσεις του μοντέλου του συστήματός μας έχουν συγκεκριμένους στόχους. Υπενθυμίζουμε ότι μία κλάση του μοντέλου μας, αντιπροσωπεύει στο αντικειμενοστραφές μοντέλο, έναν πίνακα της βάσης δεδομένων του συστήματός μας. Ο πρώτος στόχος των ελέγχων που θα εφαρμοστούν για το μοντέλο του συστήματός μας αφορά τη διαπίστωση της σωστής λειτουργίας των κανόνων εγκυρότητας των δεδομένων. Κάθε πίνακας του σχήματος μας περιλαμβάνει περιορισμούς, οι οποίοι στοιχειοθετούνται για να εκφράσουν την ορθότητα των δεδομένων, σύμφωνα με τη λογική του συστήματος. Συγκεκριμένα:

- Μοντέλο Κοινοτήτων (Community) :
  - Διαπίστωση μοναδικότητας ονόματος κοινότητας στο σύστημά μας
  - Διαπίστωση ύπαρξης ονόματος και θέματος ενδιαφέροντος
- Μοντέλο Χρήστη (User) :

- Διαπίστωση ύπαρξης ονόματος (username), ονοματεπώνυμου, e-mail, ημερομηνίας εγγραφής στο σύστημα
- Διαπίστωση ύπαρξης σωστής επαλήθευσης κωδικού χρήστη και εγκυρότητας κωδικού
- Διαπίστωση ορθότητας τιμής της ηλικίας χρήστη (1-100)

- Μοντέλο Συμμετοχής σε κοινότητα (Membership) :
  - Διαπίστωση εγκυρότητας τιμής ρόλου χρήστη σε κοινότητα, εντός των τιμών «απλός χρήστης», «administrator»
- Μοντέλο Αγαθού (Commodity) :
  - Διαπίστωση ύπαρξης ονόματος
- Μοντέλο Εικόνας Αγαθού (Picture)
  - Διαπίστωση εγκυρότητας τύπου αρχείου εικόνας
- Μοντέλο Ζήτησης (Demand) :
  - Διαπίστωση εγκυρότητας τιμής τύπου συναλλαγής εντός των τιμών «ανταλλαγή», «δωρεά», «κοινοχρησία», «όλοι οι τύποι»
- Μοντέλο Προσφοράς (Offer)
  - Διαπίστωση εγκυρότητας τιμής τύπου συναλλαγής εντός των τιμών «ανταλλαγή», «δωρεά», «κοινοχρησία», «όλοι οι τύποι»

Η Ruby on Rails έχει φροντίσει για την εισαγωγή δεδομένων ελέγχου στην βάση δεδομένων μας. Συγκεκριμένα, παρέχει ειδικό φάκελο, τον φάκελο fixtures, κάτω από τον φάκελο test, ο οποίος περιέχει αρχεία για την είσοδο εγγραφών σε πίνακες της βάσης δεδομένων του συστήματος. Αρχικά, σημειώνουμε ότι ένα fixture είναι το περιβάλλον ελέγχου επί του οποίου θα λειτουργήσει η εφαρμογή μας κατά την διαδικασία ελέγχου. Στην Ruby on Rails, τα fixtures αποτελούν την αρχικοποίηση των μοντέλων τα οποία θα υποβληθούν σε έλεγχο. Στόχος μας στο περιβάλλον ελέγχου είναι να ξεκινήσουμε με γνωστά δεδομένα εντός των πινάκων της βάσης δεδομένων μας. Γι' αυτό το λόγο η Ruby on Rails, επιτρέπει την διαμόρφωση αρχείων με ειδική μορφή, εντός των οποίων εισάγονται τα δεδομένα αυτά. Σε κάθε κλάση, που δημιουργείται στο μοντέλο μας και συνοδεύει την δημιουργία ενός πίνακα της βάσης δεδομένων μας, αντιστοιχεί ένα αρχείο εντός του φακέλου fixtures της εφαρμογής μας. Κατά αυστηρή σύμβαση, το όνομα αυτού του αρχείου συμφωνεί με το όνομα του πίνακα της βάσης δεδομένων, τον οποίο πρόκειται να αρχικοποιήσει. Ένα τέτοιου είδους αρχείο περιέχει μία εγγραφές οι οποίες αντιστοιχούν σε εγγραφές που πρόκειται να εισέλθουν στον πίνακα της βάσης δεδομένων, που αντιστοιχεί σε αυτό το αρχείο. Κάθε τέτοια εγγραφή έχει αυθαίρετο όνομα αλλά τα πεδία της εγγραφής έχουν το όνομα της στήλης του πίνακα για τις

οποίες φέρουν τιμές. Για παράδειγμα, μία εγγραφή για τον πίνακα communities της βάσης δεδομένων θα αρχικοποιηθεί λόγω αρχείου fixture με εγγραφή σε μορφή:

first_community:

id: <value>

name: <value>

theme_of_interest: <value>

description: <value>

user_id: <value>

date: <value>

Όπως είναι φυσικό το όνομα της εγγραφής, το οποίο είναι αυθαίρετο, δεν έχει καμία σημασία για την βάση δεδομένων μας, αλλά χρησιμοποιείται προκειμένου να αναφερθούμε σε αυτές τις γραμμές του πίνακα στον κώδικα που γράφουμε για έλεγχο.

Για να γίνει η εισαγωγή αυτών των δεδομένων στους πίνακες της βάσης δεδομένων του συστήματός μας θα πρέπει να εισάγουμε κατάλληλη εντολή στα αρχεία ελέγχου, πριν από τις υπόλοιπες εντολές ελέγχου. Η εντολή αυτή είναι:

fixture :<όνομα_πίνακα>

Εξαιτίας αυτής της εντολής, όταν θα δοθεί η εντολή να τρέξει ο κώδικας ελέγχου, πριν εκτελεστούν οι εντολές που του κώδικα, ο πίνακας της βάσης δεδομένων θα αδειάσει και έπειτα θα γεμίσει με τις γραμμές τις οποίες υποδεικνύει το αρχείο fixture. Με αυτό τον τρόπο, κάθε μέθοδος ελέγχου, ενεργεί επί νέων, γνωστών δεδομένων.

Η Ruby on Rails για την προσπέλαση των δεδομένων που προσφέρονται από τα fixtures, παρέχει μέθοδο ίδιο με το όνομα του αρχείου fixture και επομένως ίδιο με το όνομα του πίνακα της βάσης δεδομένων. Η μέθοδος αυτή παίρνει ως παράμετρο το όνομα που έχει δοθεί στην εγγραφή εντός του αρχείου fixture και επιστρέφει το αντικείμενο που εκπροσωπεί την αντίστοιχη εγγραφή στον πίνακα της βάσης δεδομένων μας. Το αντικείμενο αυτό είναι αντικείμενο της κλάσης του μοντέλου που αντιστοιχεί σε αυτό τον πίνακα της βάσης δεδομένων.

6.1.2 Έλεγχοι Λειτουργικότητας Συστήματος – Functional Testing

Οι controllers καθοδηγούν την επίδειξη των δεδομένων. Λαμβάνουν εισερχόμενες αιτήσεις , που συνήθως προκύπτουν από την είσοδο που δίνει ο χρήστης στο σύστημά μας, και αλληλεπιδρούν με το μοντέλο μας για να διαπιστώσουν την κατάσταση του συστήματος και να συλλέξουν τα δεδομένα. Η απόκριση των controllers συνίσταται στην έγερση της κατάλληλης view και στην διαμόρφωσή της ώστε να παρουσιάσει κατάλληλα δεδομένα στον

χρήστη. Ο έλεγχος λοιπόν των controllers συνίσταται στην διαπίστωση ότι το σύστημά μας αποκρίνεται κατάλληλα σε μία δεδομένη αίτηση. Κατά την πραγματική λειτουργία της εφαρμογής μας στο Διαδίκτυο, η εφαρμογή μας διαχειρίζεται μία αίτηση δημιουργώντας ένα κατάλληλο αντικείμενο, το αντικείμενο request, το οποίο περιέχει τις επικεφαλίδες HTTP και τα δεδομένα Post ή Get. Ωστόσο, στο περιβάλλον ελέγχου που μας παρέχει το framework μας, υπάρχει έκδοση του αντικειμένου request η οποία μπορεί να αρχικοποιηθεί χωρίς να χρειάζεται μία πραγματική, εισερχόμενη HTTP request. Επίσης, η απόκριση που παράγει ο controller σε μία HTTP request, είναι μία HTTP response η οποία αποστέλλεται παρασκηνιακά στον browser. Τα αντικείμενα request και response που αλληλεπιδρούν με τον controller είναι πολύ σημαντικά για την εφαρμογή και για τον λειτουργικό έλεγχο της εφαρμογής. Γι' αυτό το λόγο, το περιβάλλον ελέγχου που παρέχει η Ruby on Rails, δεν επιβάλλει την ανάγκη ύπαρξης ενός πραγματικού web server, ενός δικτύου ή ενός client, για να διαχειριστεί τα requests και responses. Υπάρχει ειδική μέθοδος η οποία προσομοιώνει μία HTTP request, την κατευθύνει στον controller και μετά αιχμαλωτίζει και διαχειρίζεται την απόκριση του controller.

Το σύστημά μας περιλαμβάνει N controllers. Στον φάκελο functional που βρίσκεται κάτω από τον φάκελο test του workspace μας διαπιστώνουμε την ύπαρξη N αρχείων. Κάθε ένα από αυτά τα αρχεία αντιστοιχεί σε έναν controller και περιλαμβάνει τους ελέγχους που πρέπει να εφαρμοστούν για την επαλήθευση της σωστής λειτουργίας του.

Ένας από τους σημαντικότερους ελέγχους που πρέπει να γίνουν στο σύστημά μας, αφορά την είσοδο ενός χρήστη στο σύστημά μας. Επειδή έχουμε περιορίσει την αλληλεπίδραση ενός χρήστη που δεν αποτελεί μέλος του συστήματός μας , πρέπει να διασφαλίσουμε την εφαρμογή των δικαιωμάτων προσπέλασης χρήστη στο σύστημά μας. Οποιαδήποτε προσπάθεια ενός χρήστη, που δεν έχει εισέλθει στο σύστημά μας, να προσπελάσει σελίδες στις οποίες επιτρέπεται πρόσβαση μόνο σε χρήστη του συστήματός, προκαλεί την έγερση της ιστοσελίδας που ζητάει στον χρήστη να εισέλθει στο σύστημα κάνοντας log in. Η δομή του συστήματός μας υποδεικνύει την απαγόρευση προσπέλασης σε όλες τις ιστοσελίδες του συστήματος και τις υπηρεσίες που αυτές παρέχουν, εκτός από την login ιστοσελίδα την οποία διαχειρίζεται ο EnterController.

Η διαδικασία λοιπόν που εφαρμόσαμε για να εξασφαλίσουμε ότι οι ιστοσελίδες μας θα είναι απροσπέλαστες από χρήστες που δεν έχουν κάνει login, περιέλαβε όλους τους controllers συστήματος που είναι αρμόδιοι για την εμφάνιση ιστοσελίδων. Συγκεκριμένα, προσομοιώσαμε HTTP αιτήσεις τύπου GET, στο περιβάλλον ελέγχου που παρέχει το framework, τις οποίες κατευθύναμε προς τις index μεθόδους όλων των controllers, εκτός του EnterContoller, και μεριμνήσαμε για την σύλληψη των αποκρίσεων. Επιπλέον, κάναμε χρήση

της μεθόδου assert_response της Ruby on Rails, η οποία κρίνει αν η απόκριση που λάβαμε ήταν επιτυχής.

Ο παραπάνω έλεγχος έδωσε επιτυχή αποτελέσματα μόνο στην περίπτωση του EnterController. Στην περίπτωση όλων των άλλων controllers, παρατηρήσαμε ότι η HTTP απόκριση που λάβαμε είχε κωδικό 302. Αυτό σημαίνει ότι η αίτηση που θέσαμε, ανακατευθύνθηκε και έτσι δεν θεωρήθηκε ανεπιτυχής. Το αποτέλεσμα αυτό συνάδει με την λειτουργικότητα την οποία έχουμε θέσει, καθώς κάθε controller έχει ένα φίλτρο εισόδου το οποίο διαχειρίζεται κάθε κλήση προς μέθοδο την οποία ο χρήστης δεν έχει δικαίωμα να προσπελάσει. Το φίλτρο εισόδου προωθεί την αίτηση προς μέθοδο η οποία ελέγχει αν το αντικείμενο session φέρει έναν έγκυρο αριθμό user_id. Στην περίπτωση που υπάρχει έγκυρο user_id, η κλήση διαχειρίζεται από την μέθοδο, για την οποία αρχικά προοριζόταν. Διαφορετικά, η κλήση ανακατευθύνεται προς την login μέθοδο του EnterController.

Εν συνεχεία, διαπιστώσαμε ότι ένας χρήστης ο οποίος έχει κάνει login μπορεί να προσπελάσει όλες τις ιστοσελίδες του συστήματος. Για αυτό το λόγο, εφαρμόσαμε ξανά την παραπάνω διαδικασία, αλλά αυτή τη φορά φροντίσαμε ο χρήστης που στέλνει την HTTP αίτηση να φαίνεται πως έχει κάνει login. Το αποτέλεσμα ήταν επιτυχές για όλους τους controllers.

Στην παραπάνω διαδικασία, παρουσιάστηκε το θέμα της προσομοίωσης ενός χρήστη που έχει κάνει login στο σύστημά μας. Για να προσομοιώσουμε τον παραπάνω χρήστη, λοιπόν, αρκεί το αντικείμενο session να φέρει έγκυρο user_id. Όμως, το user_id το οποίο φέρει το αντικείμενο session δεν θα μπορούσε να είναι ένας τυχαίος αριθμός, καθώς η μέθοδος που διαπιστώνει την εγκυρότητά του φέρνει από τον πίνακα των χρηστών την εγγραφή που αντιστοιχεί στον συγκεκριμένο χρήστη, σύμφωνα με το user_id του αντικειμένου session. Γι' αυτό το λόγο θα πρέπει να εισάγουμε εγγραφές στον πίνακα χρηστών της βάση δεδομένων. Η εγκυρότητα των δεδομένων μας είναι πολύ σημαντική καθώς επιδιώκουμε να δημιουργήσουμε ένα περιβάλλον ελέγχου το οποίο να ανταποκρίνεται στην εικόνα που θα έχει το σύστημά μας όταν θα είναι στο επίπεδο παραγωγής. Προσπαθώντας, λοιπόν, να έχουμε δεδομένα ελέγχου που δίνουν εικόνα σύμφωνη με πραγματικά δεδομένα, παρουσιάζεται η ιδιαιτερότητα που εμφανίζει η αποθήκευση του κωδικού του χρήστη. Τα fixtures που παρουσιάσαμε στο προηγούμενο εδάφιο, χρησιμοποιούνται και από τις μεθόδους ελέγχου των controllers για την αρχικοποίηση των πινάκων της βάσης δεδομένων με τα οποία θα αλληλεπιδράσουν οι controllers. Ωστόσο, στην περίπτωση που μελετάμε , η αποθήκευση του κωδικού στην αντίστοιχη στήλη του πίνακα της βάσης δεδομένων περιλαμβάνει την διαδικασία κρυπτογράφησης του κωδικού. Η διαδικασία αυτή ενορχηστρώνεται από μέθοδο του μοντέλου της εφαρμογής μας. Σε αυτό το σημείο λοιπόν εισάγουμε την έννοια του δυναμικού fixture, το οποίο μπορεί να χρησιμοποιήσει κώδικα της εφαρμογής για την

διαμόρφωση τιμής ενός πεδίου της εγγραφής που περιέχει. Με αυτή την τεχνική διαμορφώνουμε κατάλληλα τις εγγραφές που θα αρχικοποιήσουν τον πίνακα users για να εφαρμόσουμε τον έλεγχο πρόσβασης των χρηστών.

Ένας ακόμα έλεγχος που πρέπει να εφαρμοστεί αφορά την προσπέλαση εγγραφών στην βάση δεδομένων μας, όταν τις αναζητάμε σύμφωνα με το πρωτεύον κλειδί του πίνακα, δηλαδή σύμφωνα με το πεδίο id . Η προσπάθεια προσπέλασης εγγραφής με id, το οποίο δεν υπάρχει στον πίνακα της βάσης δεδομένων μας προκαλεί την έγερση exception η οποία αν δεν διαχειριστεί από τον κώδικά μας οδηγεί σε παρουσίαση σφάλματος στον χρήστη. Γι' αυτό το λόγο, φροντίσαμε να διαχειριστούμε κάθε προσπάθεια προσπέλασης εγγραφών των πινάκων της βάσης δεδομένων μας, εξασφαλίζοντας την σύλληψη πιθανών exceptions. Με αυτό τον τρόπο ακόμα και αν οι παράμετροι μίας HTTP αίτησης φέρουν λανθασμένο id, το οποίο θα χρησιμοποιηθεί για τον εντοπισμό εγγραφής στην βάση δεδομένων, το πρόγραμμά μας δεν οδηγείται σε σφάλμα κατά την εκτέλεση λόγω της σε exception που εγείρεται. Αντίθετα, ο κώδικάς μας διαχειρίζεται κατάλληλα αυτή την exception. Για να διαπιστώσουμε αυτή τη λειτουργικότητα, προσομοιώσαμε HTTP αιτήσεις με λανθασμένες παραμέτρους που θα οδηγούσαν σε exceptions και ελέγξαμε ότι το σύστημά μας δεν οδηγείται σε σφάλμα ή σε λανθασμένη κατάσταση.

Εκτός όμως από την ορθότητα του κώδικά μας, ο έλεγχος που εφαρμόσαμε επιδίωξε την επιθεώρηση του συστήματός μας και από την σκοπιά του τελικού χρήστη του συστήματος, ώστε να διαπιστωθεί η απόκριση της λειτουργικότητας στις απαιτήσεις που έχουν προδιαγραφεί, από την σκοπιά του τελικού χρήστη του συστήματος. Οι περιπτώσεις χρήσεις και τα σενάρια ελέγχου που διαμορφώθηκαν επιδίωξαν να καλύψουν την πλήρη λειτουργικότητα του συστήματος όπως αυτή έχει καταγραφεί στο δεύτερο κεφάλαιο που παρουσιάζει τις προδιαγραφές απαιτήσεων του συστήματος. Στο επόμενο εδάφιο παρουσιάζουμε αναλυτικά τα σενάρια ελέγχου που εφαρμόστηκαν και την απόκρισή τους.

6.2 Αναλυτική παρουσίαση ελέγχου

Στόχος του συστήματος που υλοποιήσαμε είναι να εξυπηρετεί τον τελικό χρήστη μας με τρόπο απλό και κατανοητό, παρέχοντας του τη λειτουργικότητα που έχει προδιαγραφεί, διατηρώντας την διαφάνεια των λειτουργιών σε επίπεδο business logic και αποκρύπτοντας τον τρόπος υλοποίησης των λειτουργιών. Για να εξασφαλίσουμε την πιστότητα του συστήματός μας, καλύψαμε τις περιπτώσεις χρήσης και διαμορφώσαμε σενάρια ελέγχου σύμφωνα με τις προδιαγραφές απαιτήσεων του συστήματός μας. Παρακάτω παρουσιάζεται αναλυτικά το σχέδιο ελέγχου το οποίο εφαρμόσαμε και τα αποτελέσματα τα οποία

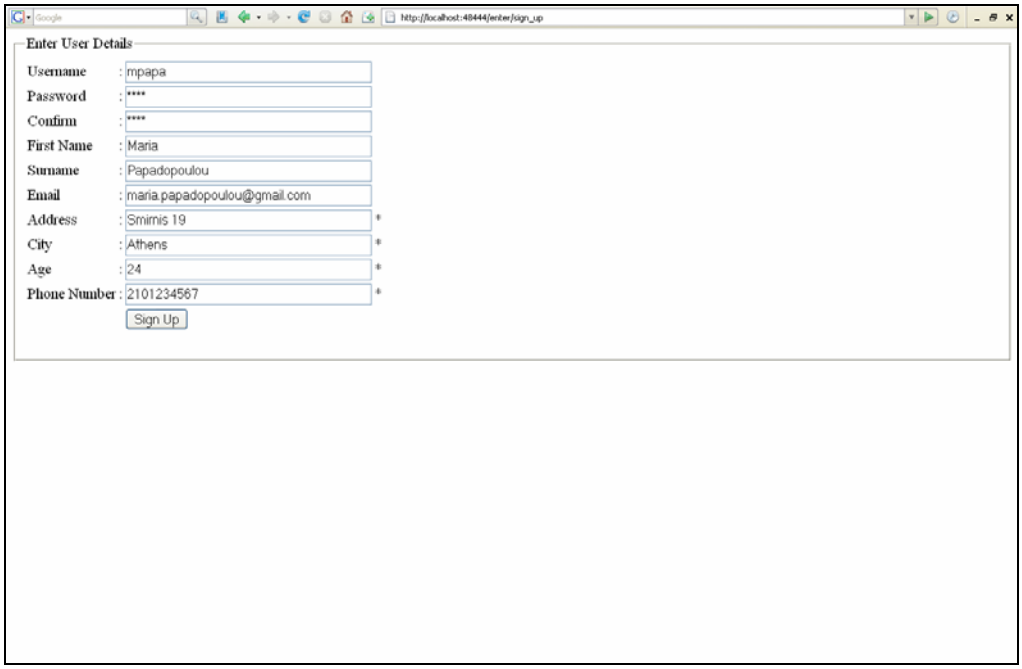
προέκυψαν. Για να γίνουν άμεσα κατανοητά τα σενάρια, χρησιμοποιούμε ένα απλό παράδειγμα σε κάθε ένα από αυτά, παραθέτοντας ταυτόχρονα ορισμένες αποτυπώσεις του συστήματος μας κατά την εκτέλεση τους.

6.2.1 Πρώτη Εγγραφή Χρήστη στο Συστήματα

Η χρήστης Maria Papadopoulou ενεργοποιεί το πλήκτρο signup της αρχικής σελίδας του συστήματος και οδηγείται σε σελίδα που περιλαμβάνει φόρμα και καλείται να συμπληρώσει ορισμένα πεδία της υποχρεωτικά και ορισμένα προαιρετικά.

6.2.1.1 <u>Σενάριο1:</u> Έγκυρα και Πλήρη Δεδομένα Εγγραφής

Η χρήστης Maria συμπληρώνει πλήρως την φόρμα με έγκυρα δεδομένα και πατάει το κουμπί κατάθεσης της φόρμας.



Διαπιστώνουμε ότι:

- Δημιουργείται εγγραφή στον πίνακα users της βάσης δεδομένων μας με πεδία Maria, Papadopoulou, mpapa, maria.papadopoulou@gmail.com 1234, Smirnis 19, Athens, 24 και 2101234567 σύμφωνα με τις παραμέτρους που έθεσε ο χρήστης Maria.
- Δημιουργία εγγραφής στον πίνακα entities της βάσης δεδομένων μας με τιμή του πεδίου actor_id ίση με την τιμή του πρωτεύοντος κλειδιού του πίνακα user και τιμή ‘User’ στο πεδίο actor_type.
- Η χρήστης Maria πλοηγείται στην login ιστοσελίδα του συστήματος.

- Η login ιστοσελίδα του συστήματος παρουσιάζει κατάλληλο μήνυμα που ενημερώνει τον χρήστη για την επιτυχή δημιουργία λογαριασμού στο σύστημά μας και τον προτρέπει να εισέλθει στο σύστημα.

6.2.1.2 <u>Σενάριο2:</u> Έγκυρα Δεδομένα Εγγραφής – Μη συμπλήρωση προαιρετικών πεδίων

Η χρήστης Maria συμπληρώνει την φόρμα με έγκυρα δεδομένα αλλά δεν συμπληρώνει τα προαιρετικά πεδία και πατάει το κουμπί κατάθεσης της φόρμας.

Διαπιστώνουμε ότι:

- Δημιουργείται εγγραφή στον πίνακα users της βάσης δεδομένων μας με κατάλληλα συμπληρωμένα τα πεδία σύμφωνα με τις παραμέτρους που έθεσε η χρήστης Maria και με NULL τιμές στα προαιρετικά πεδία.
- Δημιουργία εγγραφής στον πίνακα entities της βάσης δεδομένων μας με τιμή του πεδίου actor_id ίση με την τιμή του πρωτεύοντος κλειδιού του πίνακα user και τιμή 'User' στο πεδίο actor_type.
- Η χρήστης Maria πλοηγείται στην login ιστοσελίδα του συστήματος
- Η login ιστοσελίδα του συστήματος παρουσιάζει κατάλληλο μήνυμα που ενημερώνει τον χρήστη για την επιτυχή δημιουργία λογαριασμού στο σύστημά μας και τον προτρέπει να εισέλθει στο σύστημα.

6.2.1.3 <u>Σενάριο3:</u> Μη Πλήρης Συμπλήρωση Δεδομένων Φόρμας Εγγραφής

Η χρήστης Maria συμπληρώνει την φόρμα αλλά αφήνει κενό ένα από τα μη προαιρετικά πεδία της φόρμας, όπως π.χ. πεδίο του ονόματος του χρήστη και πατάει το κουμπί κατάθεσης της φόρμας.

Διαπιστώνουμε ότι:

- Δεν δημιουργείται εγγραφή στον πίνακα users της βάσης δεδομένων.
- Δεν δημιουργείται εγγραφή στον πίνακα entities της βάσης δεδομένων.
- Η χρήστης Maria παραμένει στην σελίδα που περιλαμβάνει την φόρμα συμπλήρωσης στοιχείων.
- Κατάλληλο μήνυμα παρουσιάζεται στη χρήστη Maria, το οποίο υποδεικνύει την συμπλήρωση του/των πεδίου/ων που παρέλειψε ο χρήστης.

6.2.1.4 <u>Σενάριο4:</u> Μη Έγκυρα Δεδομένα Εγγραφής

Η χρήστης Maria συμπληρώνει την φόρμα με μη έγκυρα δεδομένα εγγραφής και πατάει το κουμπί κατάθεσης της φόρμας. Η περιπτώσεις μη έγκυρων δεδομένων είναι οι εξής:

- Username που κατέχει ήδη άλλος χρήστης του συστήματος
- Διαφορετικός κωδικός στο πεδίο password και στο πεδίο password confirmation
- Λανθασμένη τιμή του πεδίου της ηλικίας της φόρμας εγγραφής, όπως π.χ. αλφαριθμητική τιμή ή αριθμητική τιμή εκτός του διαστήματος $[1-100]$.

Διαπιστώνουμε ότι:

- Δεν δημιουργείται εγγραφή στον πίνακα users της βάσης δεδομένων.
- Δεν δημιουργείται εγγραφή στον πίνακα entities της βάσης δεδομένων.
- Η χρήστης Maria παραμένει στην σελίδα που περιλαμβάνει την φόρμα συμπλήρωσης στοιχείων.
- Κατάλληλο μήνυμα παρουσιάζεται στη χρήστη Maria, το οποίο υποδεικνύει την λανθασμένη συμπλήρωση πεδίου/ων.

6.2.2 Είσοδος Χρήστη στο Σύστημα

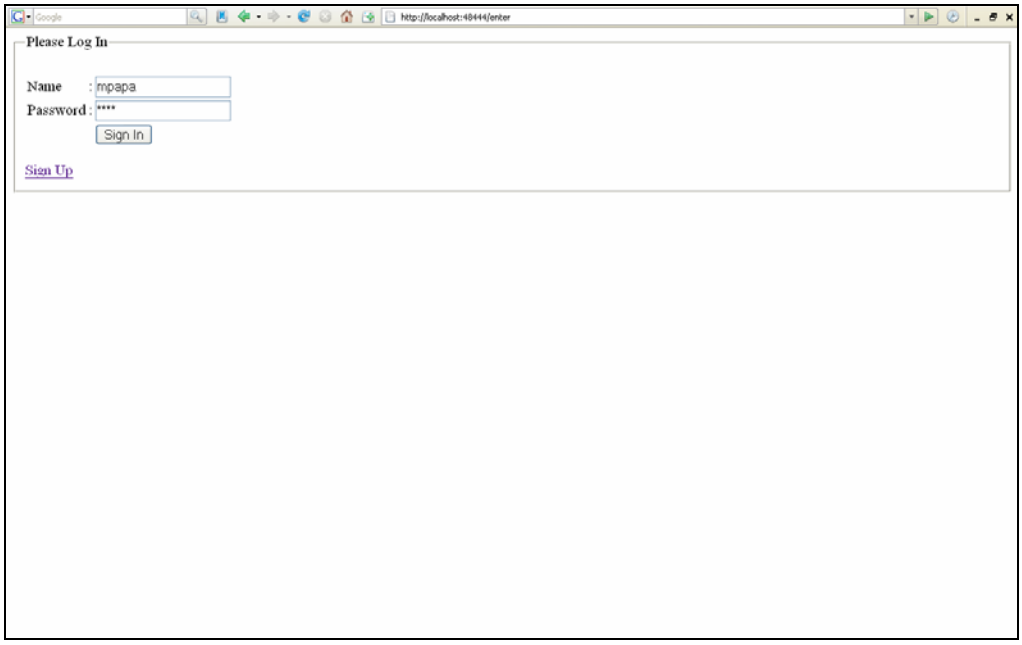
Η χρήστης Maria πλοηγείται στην αρχική σελίδα (login ιστοσελίδα) του συστήματος η οποία περιλαμβάνει δύο πεδία για την συμπλήρωση username και password του χρήστη. Σημειώνουμε ότι η χρήστης Maria μπορεί να πλοηγηθεί στην login ιστοσελίδα του συστήματός μας με δύο τρόπους:

- Θέτει HTTP αίτηση προς την login ιστοσελίδα
- Θέτει HTTP αίτηση προς άλλη ιστοσελίδα του συστήματος την οποία δεν έχει δικαίωμα να προσπελάσει γιατί δεν έχει κάνει login

Στην πρώτη περίπτωση, επιτυχές login στο σύστημα θα οδηγήσει στον χρήστη στην κεντρική σελίδα του συστήματος . Αντίθετα, στην δεύτερη περίπτωση η χρήστης Maria θα πλοηγηθεί στην σελίδα την οποία είχε εξαρχής ζητήσει να προσπελάσει. Παρακάτω αναφέρουμε την σελίδα στην οποία πλοηγείται η χρήστης με επιτυχές login ως «ιστοσελίδα στόχος».

6.2.2.1 ***Σενάριο1:*** *Έγκυρα και Πλήρη Δεδομένα Εισόδου*

Η χρήστης Maria συμπληρώνει τα πεδία του username και password με τα αντίστοιχα στοιχεία λογαριασμού που έχει ήδη δημιουργήσει και πατάει το κουμπί για την είσοδο στο σύστημα.



Διαπιστώνουμε ότι:

- Δημιουργείται κατάλληλο αντικείμενο session το οποίο φέρει στο πεδίο του user_id την τιμή της στήλης id που αντιστοιχεί στην εγγραφή του χρήστη Maria στον πίνακα users της βάσης δεδομένων.
- Δημιουργείται κατάλληλη εγγραφή στον πίνακα sessions η οποία έχει κατάλληλα συμπληρωμένα τα πεδία της.
- Η χρήστης Maria πλοηγείται στην «ιστοσελίδα στόχο».

6.2.2.2 ***Σενάριο2:*** *Ελλιπή Δεδομένα Εισόδου*

Η χρήστης Maria συμπληρώνει ελλιπώς τα πεδία για την είσοδο στο σύστημα και πατάει το κουμπί εισόδου στο σύστημα. Οι περιπτώσεις ελλιπούς συμπλήρωσης των πεδίων είναι οι εξής:

- Μη συμπλήρωση των πεδίων.
- Συμπλήρωση μόνο του πεδίου username.
- Συμπλήρωση μόνο του πεδίου password.

Διαπιστώνουμε ότι:

- Η χρήστης Maria παραμένει στην login ιστοσελίδα.

- Η χρήστης Maria ενημερώνεται για την ελλιπή συμπλήρωση των στοιχείων του λογαριασμού του με κατάλληλο μήνυμα που εμφανίζεται στην login ιστοσελίδα.
- Δεν λαμβάνει χώρα καμία ανανέωση σε πίνακες της βάσης δεδομένων μας.

6.2.2.3 <u>Σενάριο3:</u> Εσφαλμένη Συμπλήρωση Στοιχείων Λογαριασμού Χρήστη

Η χρήστης Maria συμπληρώνει εσφαλμένα τα πεδία για την είσοδο στο σύστημα και πατάει το κουμπί εισόδου στο σύστημα. Οι περιπτώσεις εσφαλμένης συμπλήρωσης των πεδίων είναι οι εξής:

- Μη έγκυρο username – Μη έγκυρο password.
- Μη έγκυρο username – Έγκυρο password.
- Έγκυρο username – Μη έγκυρο password.

Διαπιστώνουμε ότι:

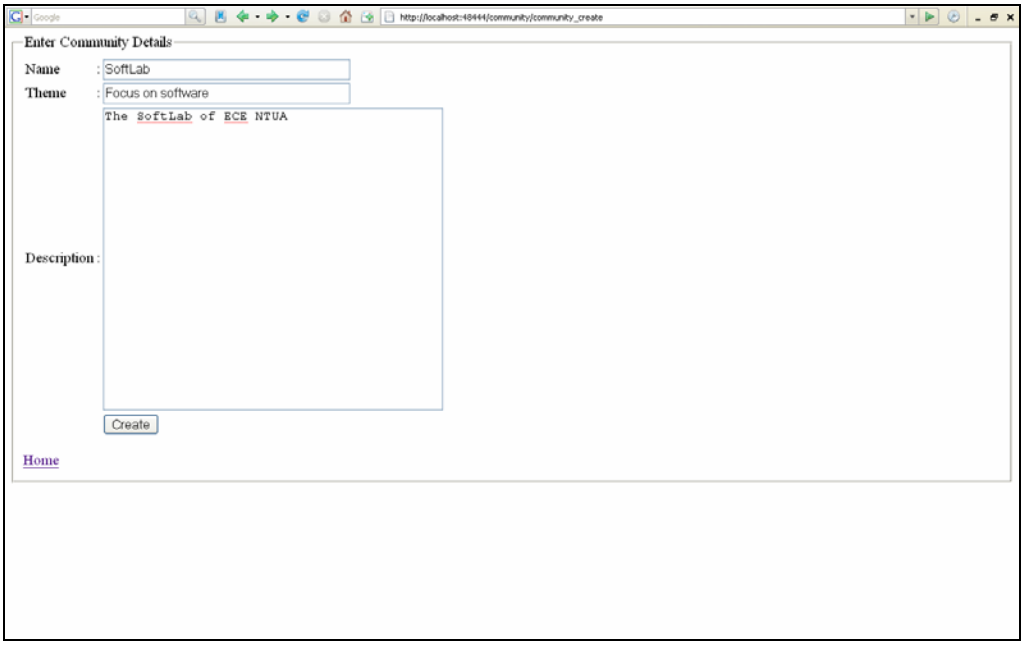
- Η χρήστης Maria παραμένει στην login ιστοσελίδα.
- Η χρήστης Maria ενημερώνεται για την εσφαλμένη συμπλήρωση των στοιχείων του λογαριασμού του με κατάλληλο μήνυμα που εμφανίζεται στην login ιστοσελίδα.
- Δεν λαμβάνει χώρα καμία ανανέωση σε πίνακες της βάσης δεδομένων μας.

6.2.3 Ορισμός Κοινοτήτων

Ένας χρήστης που έχει εισέλθει στο σύστημά μας, μπορεί να δημιουργήσει μία κοινότητα. Στην περίπτωση αυτή, ο χρήστης πλοηγείται σε κατάλληλη φόρμα την οποία καλείται να υποβάλει, αφού συμπληρώσει με τα κατάλληλα στοιχεία.

6.2.3.1 <u>Σενάριο1:</u> Έγκυρα και Πλήρη Στοιχεία Κοινότητας

Η χρήστης Maria συμπληρώνει πλήρως, με έγκυρα στοιχεία τα πεδία της φόρμας για να δημιουργήσει την κοινότητα SoftLab και υποβάλει την φόρμα. Η εγκυρότητα των στοιχείων θίγεται μονάχα στην περίπτωση κατά την οποία η χρήστης Maria προσπαθήσει να δημιουργήσει κοινότητα με όνομα το οποίο κατέχει ήδη άλλη κοινότητα του συστήματος.



Διαπιστώνουμε ότι:

- Δημιουργείται εγγραφή με πεδία κατάλληλα συμπληρωμένα στον πίνακα communities της βάσης δεδομένων. Σημειώνουμε ότι ένα από τα πεδία της εγγραφής αυτής δηλώνει τον δημιουργό της κοινότητας. Συγκεκριμένα το πεδίο αυτό λαμβάνει την τιμή του πεδίου id της εγγραφής του πίνακα users, που αντιστοιχεί στον χρήστη που δημιουργεί την κοινότητα. Την τιμή αυτή την λαμβάνουμε από το πεδίο user_id του αντικειμένου session που αντιστοιχεί στον χρήστη που δημιουργεί την κοινότητα.
- Δημιουργείται εγγραφή στον πίνακα entities που έχει τιμή στην στήλη actor_id την τιμή του πεδίου id της εγγραφής που δημιουργήθηκε στον πίνακα communities για την κοινότητα αυτή. Επιπλέον, το πεδίο actor_type της εγγραφής του πίνακα entities, έχει την τιμή ‘Community’.
- Δημιουργείται εγγραφή στον πίνακα membership που δηλώνει ότι ο δημιουργός της κοινότητας αποτελεί μέλος της κοινότητας με ρόλο administrator.
- Η χρήστης Maria πλοηγείται σε ιστοσελίδα που παρουσιάζει όλες τις κοινότητες του συστήματος.

6.2.3.2 <u>Σενάριο2:</u> Έγκυρα Στοιχεία Κοινότητας – Μη Συμπλήρωση προαιρετικού πεδίου

Η φόρμα δημιουργίας κοινότητας περιλαμβάνει ένα προαιρετικό πεδίο. Το πεδίο αυτό αφορά την περιγραφή της κοινότητας. Η χρήστης Maria, λοιπόν, συμπληρώνει

με έγκυρα στοιχεία τα πεδία της φόρμας δημιουργίας κοινότητας αλλά παραλείπει την απόδοση τιμής στο πεδίο περιγραφής της κοινότητας. Έπειτα υποβάλει την φόρμα. Διαπιστώνουμε ότι:

- Δημιουργείται εγγραφή με πεδία κατάλληλα συμπληρωμένα στον πίνακα communities της βάσης δεδομένων. Το πεδίο περιγραφής της κοινότητας λαμβάνει την τιμή NULL στην εγγραφή που αποθηκεύεται στον πίνακα communities.
- Δημιουργείται εγγραφή στον πίνακα entities που έχει τιμή στην στήλη actor_id την τιμή του πεδίου id της εγγραφής που δημιουργήθηκε στον πίνακα communities για την κοινότητα αυτή. Επιπλέον, το πεδίο actor_type της εγγραφής του πίνακα entities, έχει την τιμή 'Community'.
- Δημιουργείται εγγραφή στον πίνακα membership που δηλώνει ότι ο δημιουργός της κοινότητας αποτελεί μέλος της κοινότητας με ρόλο administrator.
- Η χρήστης Maria πλοηγείται σε ιστοσελίδα που παρουσιάζει όλες τις κοινότητες του συστήματος.

6.2.3.3 <u>Σενάριο3:</u> Ελλιπής Συμπλήρωση Στοιχείων Κοινότητας

Η χρήστης Maria συμπληρώνει ελλιπώς τα πεδία της φόρμας δημιουργίας της κοινότητας. Σημειώνουμε ότι το μοναδικά πεδίο το οποίο είναι προαιρετικό για την δημιουργία της κοινότητας είναι η περιγραφή της. Σε περίπτωση που κάποιο από τα άλλα πεδία δεν έχει συμπληρωθεί, η φόρμα περιλαμβάνει ελλιπή στοιχεία.

Διαπιστώνουμε ότι:

- Δεν δημιουργείται εγγραφή στον πίνακα communities.
- Δεν δημιουργείται εγγραφή στον πίνακα entities.
- Δεν δημιουργείται εγγραφή στον πίνακα membership.
- Η χρήστης Maria παραμένει στην σελίδα που περιέχει την φόρμα δημιουργίας κοινότητας.
- Στην σελίδα που περιέχει την φόρμα δημιουργίας κοινότητας εμφανίζεται κατάλληλο μήνυμα που υποδεικνύει την συμπλήρωση των πεδίων που έλειπαν.

6.2.3.4 <u>Σενάριο4:</u> Εσφαλμένα Στοιχεία Κοινότητας

Η χρήστης Maria συμπληρώνει εσφαλμένα τα πεδία της φόρμας δημιουργίας της κοινότητας. Σημειώνουμε ότι το μοναδικά πεδίο το οποίο μπορεί να λάβει λανθασμένη τιμή είναι το πεδίο του ονόματος την κοινότητας. Συγκεκριμένα, η χρήστης Maria ως όνομα της κοινότητας θέτει όνομα που κατέχει ήδη κοινότητα του συστήματος.

Διαπιστώνουμε ότι:

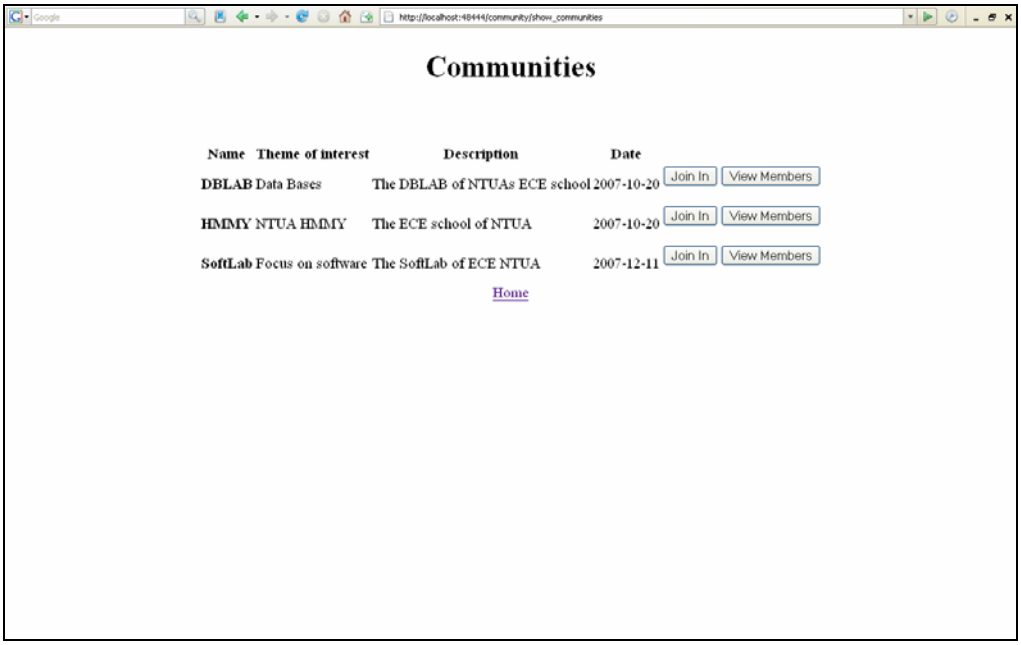
- Δεν δημιουργείται εγγραφή στον πίνακα communities.
- Δεν δημιουργείται εγγραφή στον πίνακα entities.
- Δεν δημιουργείται εγγραφή στον πίνακα membership.
- Η χρήστης Maria παραμένει στην σελίδα που περιέχει την φόρμα δημιουργίας κοινότητας.
- Στην σελίδα που περιέχει την φόρμα δημιουργίας κοινότητας εμφανίζεται κατάλληλο μήνυμα που υποδεικνύει ότι το όνομα της κοινότητας πρέπει να αλλάξει γιατί χρησιμοποιείται από άλλη κοινότητα.

6.2.4 Συμμετοχή σε Κοινότητα

Βασική προϋπόθεση για αυτή την περίπτωση χρήσης του συστήματός μας είναι η είσοδος του χρήστη στο σύστημα. Η χρήστης Maria θέτει HTTP αίτηση τύπου GET με παράμετρο το id της κοινότητας στην οποία θέλει να ενταχθεί.

6.2.4.1 <u>Σενάριο1:</u> Συμμετοχή σε υπάρχουσα - έγκυρη κοινότητα

Σε αυτό το σενάριο, το id της κοινότητας που τίθεται ως παράμετρος της HTTP αίτησης είναι έγκυρο. Αυτό σημαίνει ότι υπάρχει κοινότητα στο σύστημά μας, η οποία έχει εγγραφή στον πίνακα communities με τιμή πρωτεύοντος κλειδιού ίση με την τιμή που φέρει η HTTP αίτηση. Επιπλέον, η χρήστης Maria που θέτει την αίτηση, δεν αποτελεί μέλος της κοινότητας αυτής.



Διαπιστώνουμε ότι:

Δημιουργία εγγραφής στον πίνακα memberships που δηλώνει την συμμετοχή αυτού του χρήστη στην κοινότητα που έχει id σύμφωνο με την παράμετρο της HTTP αίτησης. Ο ρόλος του χρήστη όπως δηλώνεται σε κατάλληλο πεδίο της εγγραφής που δημιουργείται τίθεται στην τιμή member.

- Η χρήστης Maria πλοηγείται στην ιστοσελίδα που παρουσιάζει το σύνολο των κοινοτήτων του συστήματος.
- Η χρήστης Maria ενημερώνεται για την συμμετοχή του στην κοινότητα που επέλεξε μέσω κατάλληλου μηνύματος που εμφανίζεται στην ιστοσελίδα που παρουσιάζει το σύνολο των κοινοτήτων του συστήματος.

6.2.4.2 <u>Σενάριο2:</u> Συμμετοχή σε μη - έγκυρη κοινότητα

Σε αυτό το σενάριο, το id της κοινότητας που τίθεται ως παράμετρος της HTTP αίτησης δεν είναι έγκυρο. Οι περιπτώσεις μη έγκυρου id είναι οι εξής:

- Κενή τιμή.
- Αλφαριθμητική τιμή.
- Αριθμητική τιμή που δεν περιλαμβάνεται στο σύνολο τιμών που έχει ήδη λάβει το πρωτεύον κλειδί (πεδίο id) στον πίνακα communities της βάσης δεδομένων μας.

Διαπιστώνουμε ότι:

- Δεν δημιουργείται εγγραφή στον πίνακα membership.

- Η χρήστης Maria πλοηγείται στην ιστοσελίδα που παρουσιάζει το σύνολο των κοινοτήτων του συστήματος.
- Η χρήστης Maria ενημερώνεται για την αποτυχημένη προσπάθεια συμμετοχής σε κοινότητα μέσω κατάλληλου μηνύματος που εμφανίζεται στην ιστοσελίδα που παρουσιάζει το σύνολο των κοινοτήτων του συστήματος.

6.2.4.3 <u>Σενάριο3:</u> Προσπάθεια Συμμετοχής σε Κοινότητα στην οποία ο χρήστης είναι ήδη μέλος

Σε αυτό το σενάριο, το id της κοινότητας που τίθεται ως παράμετρος της HTTP αίτησης ταυτίζεται με id κοινότητας, στην οποία η χρήστης Maria ήδη ανήκει.

Διαπιστώνουμε ότι:

- Δεν δημιουργείται νέα εγγραφή στον πίνακα membership.
- Δεν ανανεώνεται κανένα πεδίο της παλιάς εγγραφής του πίνακα membership, η οποία αντιστοιχεί στην συμμετοχή του χρήστη σε αυτή την κοινότητα.
- Η χρήστης Maria πλοηγείται στην ιστοσελίδα που παρουσιάζει το σύνολο των κοινοτήτων του συστήματος.
- Η χρήστης Maria ενημερώνεται για την αποτυχημένη προσπάθεια συμμετοχής στην κοινότητα μέσω κατάλληλου μηνύματος που εμφανίζεται στην ιστοσελίδα που παρουσιάζει το σύνολο των κοινοτήτων του συστήματος. Το μήνυμα αυτό δηλώνει ότι η χρήστης Maria ανήκει ήδη σε αυτή την κοινότητα.

6.2.5 Παρουσίαση Κοινοτήτων

Η χρήστης Maria για να δει τις υπάρχουσες κοινότητες του συστήματός μας, πρέπει να έχει εισέλθει στο σύστημά μας εισάγοντας το username και το password του στην login ιστοσελίδα του συστήματός μας.

6.2.5.1 <u>Σενάριο1:</u> Παρουσίαση του συνόλου των κοινοτήτων του συστήματος

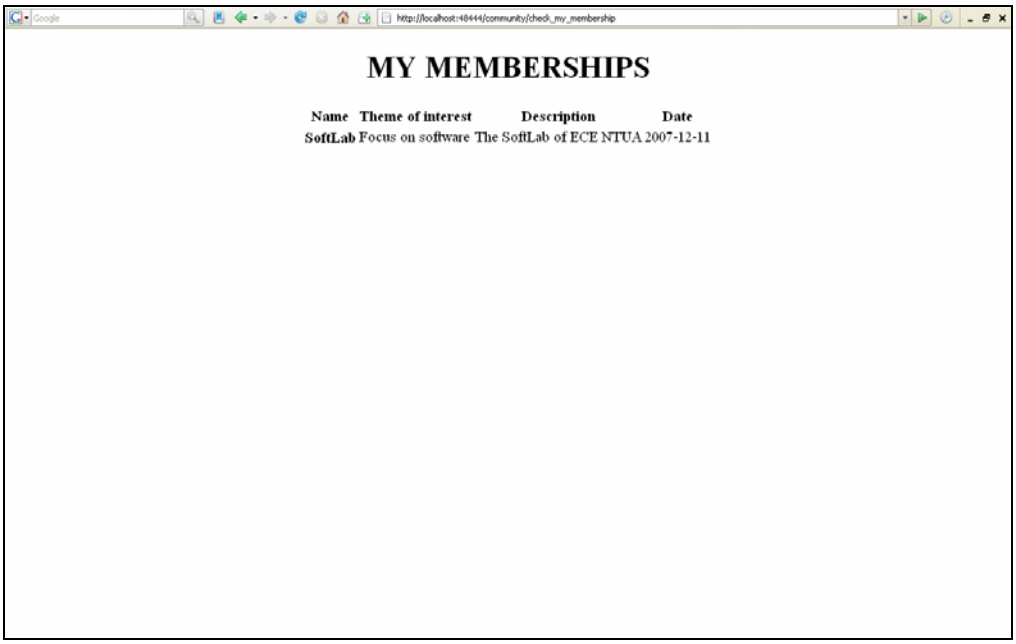
Η χρήστης Maria στέλνει HTTP αίτηση τύπου GET με κατάλληλο url στην εφαρμογή μας. Η αίτηση αυτή κατευθύνεται προς συγκεκριμένη μέθοδο του συστήματός μας.

Διαπιστώνουμε ότι:

- Η χρήστης Maria πλοηγείται σε κατάλληλη ιστοσελίδα στην οποία παρουσιάζονται όλες οι κοινότητες του συστήματός μας και τα ιδιαίτερα στοιχεία της καθεμίας τα οποία κρατούνται στον πίνακα communities.
- Τα στοιχεία της κάθε κοινότητας, όπως αυτά αναφέρονται στην ιστοσελίδα, έχουν τις αναμενόμενες τιμές σύμφωνες με αυτές που κρατάει η βάση δεδομένων μας.

6.2.5.2 <u>Σενάριο2:</u> Παρουσίαση κοινοτήτων στις οποίες ανήκει ο χρήστης

Η χρήστης Maria στέλνει HTTP αίτηση τύπου GET με κατάλληλο url στην εφαρμογή μας. Μέσω του αντικειμένου session, αναγνωρίζουμε την εγγραφή του χρήστη μας στον πίνακα users. Η μέθοδος επιστρέφει έναν πίνακα με τις κοινότητες στις οποίες ανήκει η χρήστης Maria.



Διαπιστώνουμε ότι:

- Η χρήστης Maria πλοηγείται σε κατάλληλη ιστοσελίδα στην οποία παρουσιάζονται όλες οι κοινότητες στις οποίες ανήκει. Καμία άλλη κοινότητα δεν παρουσιάζεται.
- Τα στοιχεία της κάθε κοινότητας, όπως αυτά αναφέρονται στην ιστοσελίδα, έχουν τις αναμενόμενες τιμές σύμφωνες με αυτές που κρατάει η βάση δεδομένων μας.

6.2.5.3 <u>Σενάριο3:</u> Παρουσίαση κοινοτήτων στις οποίες ανήκει έγκυρος χρήστης

Η χρήστης Maria στέλνει HTTP αίτηση τύπου GET με κατάλληλο url στην εφαρμογή μας. Η αίτηση αυτή φέρει ως παράμετρο το id του χρήστη, για τον οποίο ενδιαφερόμαστε να δούμε τις κοινότητες στις οποίες ανήκει. Η μέθοδος που ενεργοποιείται επιστρέφει έναν πίνακα με τις κοινότητες στις οποίες ανήκει η χρήστης Maria που υπονοείται από την παράμετρο της HTTP αίτησης.

Διαπιστώνουμε ότι:

- Η χρήστης Maria πλοηγείται σε κατάλληλη ιστοσελίδα στην οποία παρουσιάζονται όλες οι κοινότητες στις οποίες ανήκει ο χρήστης της παραμέτρου της αίτησης. Καμία άλλη κοινότητα δεν παρουσιάζεται.
- Τα στοιχεία των κοινοτήτων που παρουσιάζονται έχουν τις αναμενόμενες τιμές, δηλαδή τις τιμές που κρατάει η βάση δεδομένων μας.

6.2.5.4 <u>Σενάριο4:</u> Παρουσίαση κοινοτήτων στις οποίες ανήκει μη-έγκυρος χρήστης

Η χρήστης Maria στέλνει HTTP αίτηση τύπου GET με κατάλληλο url στην εφαρμογή μας. Η αίτηση αυτή φέρει ως παράμετρο το id του χρήστη, για τον οποίο ενδιαφερόμαστε να δούμε τις κοινότητες στις οποίες ανήκει. Όμως, η παράμετρος αυτή δεν είναι έγκυρη, δηλαδή έχει μία από τις παρακάτω τιμές:

- Κενή τιμή.
- Αλφαριθμητική τιμή.
- Αριθμητική τιμή που δεν περιλαμβάνεται στο σύνολο τιμών που έχει ήδη λάβει το πρωτεύον κλειδί (πεδίο id) στον πίνακα users της βάσης δεδομένων μας.

Διαπιστώνουμε ότι:

- Η χρήστης Maria πλοηγείται στην κεντρική ιστοσελίδα του συστήματος.
- Δεν υπάρχει καμία αναφορά αποτυχίας.
- Η κατάσταση των πινάκων της βάσης δεδομένων μας δεν μεταβάλλεται.

6.2.6 Παρουσίαση Στοιχείων Κοινότητας

Για να προχωρήσουμε στον έλεγχο αυτής της περίπτωσης, σημειώνουμε ότι η χρήστης Maria που επιδιώκει να δει τα στοιχεία της κοινότητας έχει ήδη εισέλθει στο σύστημα, κάνοντας login.

6.2.6.1 <u>Σενάριο1:</u> Παρουσίαση στοιχείων έγκυρης κοινότητας

Η χρήστης Maria στέλνει HTTP αίτηση τύπου GET με κατάλληλο url στην εφαρμογή μας. Η αίτηση αυτή φέρει ως παράμετρο το id της κοινότητας για της οποίας τα στοιχεία ο χρήστης δηλώνει ενδιαφέρον.

Διαπιστώνουμε ότι:

- Η χρήστης Maria πλοηγείται σε κατάλληλη ιστοσελίδα στην οποία παρουσιάζονται τα στοιχεία της κοινότητας για την οποία ο χρήστης εκδήλωσε ενδιαφέρον.
- Καμία ανανέωση δεν γίνεται στην κατάσταση των πινάκων της βάσης δεδομένων μας.

6.2.6.2 <u>Σενάριο2:</u> Παρουσίαση στοιχείων μη-έγκυρης κοινότητας

Η χρήστης Maria στέλνει HTTP αίτηση τύπου GET με κατάλληλο url στην εφαρμογή μας. Η αίτηση αυτή φέρει ως παράμετρο το id της κοινότητας για της οποίας τα στοιχεία η χρήστης Maria δηλώνει ενδιαφέρον. Η παράμετρος αυτή έχει εσφαλμένη τιμή, δηλαδή:

- Κενή τιμή.
- Αλφαριθμητική τιμή.
- Αριθμητική τιμή που δεν περιλαμβάνεται στο σύνολο τιμών που έχει ήδη λάβει το πρωτεύον κλειδί (πεδίο id) στον πίνακα communities της βάσης δεδομένων μας.

Διαπιστώνουμε ότι:

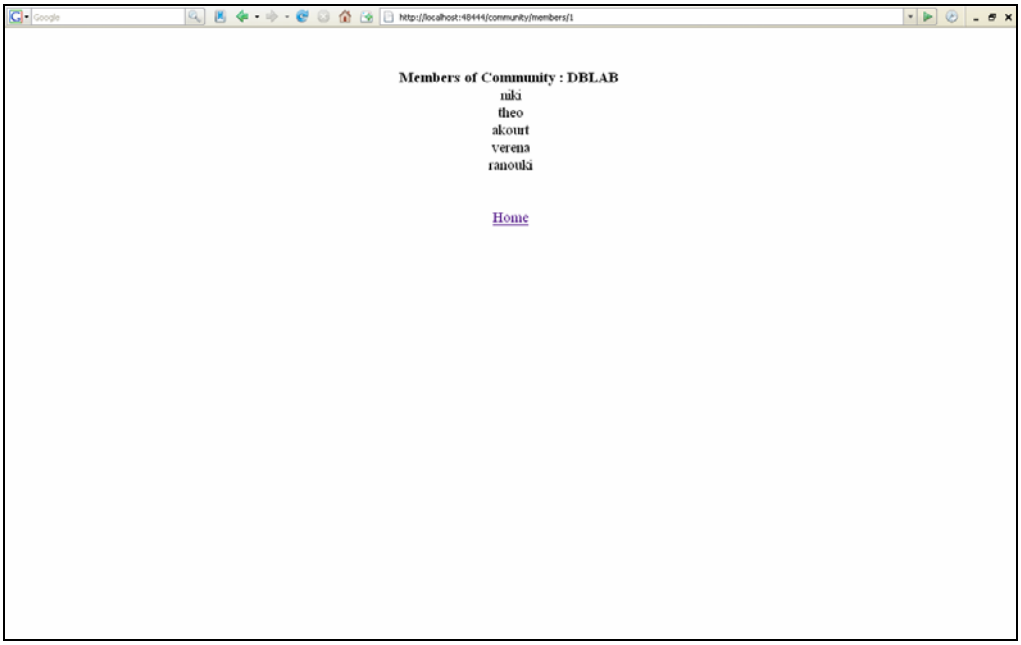
- Η χρήστης Maria πλοηγείται σε κατάλληλη ιστοσελίδα στην οποία παρουσιάζονται τα στοιχεία της κοινότητας για την οποία η χρήστης Maria εκδήλωσε ενδιαφέρον.
- Καμία ανανέωση δεν γίνεται στην κατάσταση των πινάκων της βάσης δεδομένων μας.

6.2.7 Παρουσίαση Μελών Κοινότητας

Η προϋπόθεση για την εκτέλεση της συγκεκριμένης περίπτωσης χρήσης, είναι η χρήστης Maria μας να έχει εισέλθει στο σύστημά μας.

6.2.6.1 <u>Σενάριο1:</u> Παρουσίαση Μελών έγκυρης κοινότητας

Η χρήστης Maria στέλνει HTTP αίτηση τύπου GET με κατάλληλο url στην εφαρμογή μας. Η αίτηση αυτή φέρει ως παράμετρο το id της κοινότητας της οποίας τα μέλη ενδιαφέρεται να δει η χρήστης Maria.



Διαπιστώνουμε ότι:

- Η χρήστης Maria πλοηγείται σε κατάλληλη ιστοσελίδα στην οποία παρουσιάζεται λίστα με τα μέλη της κοινότητας.
- Δεν υπάρχει μέλος της κοινότητας το οποίο να μην παρουσιάζεται στην λίστα αυτή.
- Καμία ανανέωση δεν γίνεται στην κατάσταση των πινάκων της βάσης δεδομένων μας.

6.2.6.2 <u>Σενάριο2:</u> Παρουσίαση Μελών μη-έγκυρης κοινότητας

Η χρήστης Maria στέλνει HTTP αίτηση τύπου GET με κατάλληλο url στην εφαρμογή μας. Η αίτηση αυτή φέρει ως παράμετρο το id της κοινότητας, της οποίας τα μέλη θέλει να δει η χρήστης Maria. Η παράμετρος αυτή έχει εσφαλμένη τιμή, δηλαδή:

- Κενή τιμή.
- Αλφαριθμητική τιμή.
- Αριθμητική τιμή που δεν περιλαμβάνεται στο σύνολο τιμών που έχει ήδη λάβει το πρωτεύον κλειδί (πεδίο id) στον πίνακα communities της βάσης δεδομένων μας.

Διαπιστώνουμε ότι:

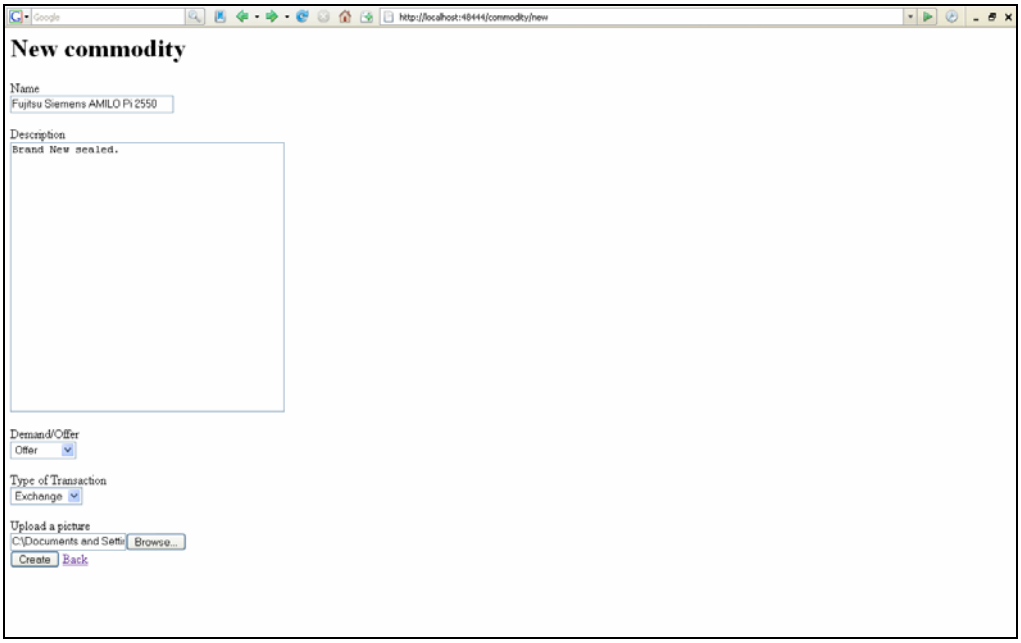
- Η χρήστης Maria πλοηγείται στην σελίδα που παρουσιάζει το σύνολο των κοινοτήτων του συστήματος.
- Στην ιστοσελίδα αυτή εμφανίζεται μήνυμα που προτρέπει τον χρήστη να ανακτήσει τα μέλη μίας από τις κοινότητες που περιλαμβάνει το σύστημα.
- Καμία ανανέωση δεν γίνεται στην κατάσταση των πινάκων της βάσης δεδομένων μας.

6.2.8 Διαχείριση Αγαθών

Η προϋπόθεση για να έχει ένας χρήστης δικαιώματα διαχείρισης αγαθών, είναι να έχει εισέλθει στο σύστημά μας.

6.2.6.1 <u>Σενάριο1:</u> Επιτυχής Εισαγωγή Αγαθού – Πλήρης Συμπλήρωση Στοιχείων Αγαθού

Η χρήστης Maria εισέρχεται στην σελίδα που περιέχει την φόρμα δημιουργίας αγαθού. Η χρήστης Maria συμπληρώνει πλήρως και με έγκυρα στοιχεία την φόρμα αυτή και την υποβάλλει.



Διαπιστώνουμε ότι:

- Δημιουργείται εγγραφή για το αγαθό αυτό στον πίνακα commodities της βάσης δεδομένων μας.
- Τα πεδία αυτής της εγγραφής συμπληρώνονται με βάση τις τιμές που έχει θέσει η χρήστης Maria. Το πεδίο της εγγραφής που υποδεικνύει τον κάτοχο

του αγαθού έχει την τιμή του id της εγγραφής του πίνακα entities που αντιστοιχεί στον χρήστη που εισήγαγε το αγαθό.

- Η χρήστης Maria πλοηγείται σε ιστοσελίδα που παρουσιάζει λίστα με τα αγαθά που κατέχει ο χρήστης και τα στοιχεία τους.

6.2.6.2 <u>Σενάριο2:</u> Επιτυχής Εισαγωγή Αγαθού – Μη Πλήρης Συμπλήρωση Στοιχείων Αγαθού

Το μοναδικό μη προαιρετικό πεδίο της φόρμας δημιουργίας αγαθού είναι το όνομα του αγαθού. Όλα τα υπόλοιπα πεδία της φόρμας μπορούν να έχουν κενές τιμές κατά την υποβολή της αίτησης. Θέτουμε λοιπόν μόνο το όνομα του αγαθού, σε έγκυρη τιμή, και υποβάλουμε την αίτηση.

Διαπιστώνουμε ότι:

- Δημιουργείται εγγραφή για το αγαθό αυτό στον πίνακα commodities της βάσης δεδομένων μας.
- Το πεδίο του ονόματος της εγγραφής τίθεται στο όνομα που έθεσε η χρήστης Maria. Επίσης η ημερομηνία εισόδου του αγαθού στην βάση του συστήματος καθώς και ο ιδιοκτήτης του αγαθού τίθενται στις σωστές τιμές. Συγκεκριμένα, το πεδίο της ημερομηνίας τίθεται στην τιμή της τρέχουσας ημερομηνίας υποβολής της φόρμας και το πεδίο της εγγραφής που υποδεικνύει τον κάτοχο του αγαθού έχει την τιμή του id της εγγραφής του πίνακα entities που αντιστοιχεί στον χρήστη που εισήγαγε το αγαθό.
- Η χρήστης Maria πλοηγείται σε ιστοσελίδα που παρουσιάζει λίστα με τα αγαθά που κατέχει.

6.2.6.3 <u>Σενάριο3:</u> Αποτυχημένη Εισαγωγή Αγαθού

Για την εισαγωγή ενός αγαθού στο σύστημά μας, η εφαρμογή μας επιβάλλει την ύπαρξη του ονόματος του αγαθού. Μελετήσαμε, λοιπόν, την περίπτωση κατά την οποία η χρήστης Maria υποβάλλει την φόρμα δημιουργίας του αγαθού χωρίς να έχει δώσει τιμή στο πεδίο του ονόματος.

Διαπιστώνουμε ότι:

- Δεν δημιουργείται εγγραφή για το αγαθό αυτό στον πίνακα commodities της βάσης δεδομένων μας.
- Η χρήστης Maria παραμένει στην ιστοσελίδα που περιλαμβάνει την φόρμα δημιουργίας αγαθού.

- Η χρήστης Maria ενημερώνεται από κατάλληλο μήνυμα, που εμφανίζεται στην ιστοσελίδα, ότι πρέπει να εισάγει το όνομα του αγαθού.

6.2.9 Αναζήτηση Αγαθών

Η φόρμα αναζήτησης αγαθών ζητά από τον χρήστη να καθορίσει τρία στοιχεία που προσδιορίζουν τα ιδιαίτερα χαρακτηριστικά των αγαθών που επιθυμεί να παρατηρήσει:

- Όνομα Αγαθού
- Τύπος διάθεσης αγαθού (δηλαδή προσφορά ή ζήτηση αγαθού)
- Είδος συναλλαγής

Τα σενάρια ελέγχου τα οποία εκτελέσαμε για να διαπιστώσουμε την ορθότητα της λειτουργίας της μεθόδου αναζήτησης αγαθού, κάλυψαν όλες τις περιπτώσεις επιτρεπτών αναζητήσεων.

6.2.9.1 Σενάριο1 : Αναζήτηση Αγαθών που προσφέρονται για ανταλλαγή

Στην φόρμα αναζήτησης αγαθού, θέτουμε το πεδίο του ονόματος του αγαθού στην τιμή που μας προσδιορίζει τα αγαθά για τα οποία ενδιαφερόμαστε. Επιπλέον, καθορίζουμε ότι θέλουμε να βρούμε αγαθά τα οποία προσφέρονται για ανταλλαγή. Πατάμε το κουμπί αναζήτησης και διαπιστώνουμε ότι:

- Η μέθοδος αναζήτησης επιστρέφει μία λίστα αγαθών που έχουν όνομα που ξεκινάει από την τιμή που έθεσε η χρήστης Maria ως όνομα αγαθού στην φόρμα αναζήτησης και τα οποία προσφέρονται για ανταλλαγή.
- Η χρήστης Maria πλοηγείται σε σελίδα που περιέχει την λίστα αγαθών με όλα τα ιδιαίτερα χαρακτηριστικά τους

6.2.9.2 Σενάριο2 : Αναζήτηση Αγαθών που προσφέρονται για δωρεά

Στην φόρμα αναζήτησης αγαθού, θέτουμε το πεδίο του ονόματος του αγαθού στην τιμή που μας προσδιορίζει τα αγαθά για τα οποία ενδιαφερόμαστε. Καθορίζουμε ότι θέλουμε να βρούμε αγαθά τα οποία προσφέρονται για δωρεά. Πατάμε το κουμπί αναζήτησης και διαπιστώνουμε ότι:

- Η μέθοδος αναζήτησης επιστρέφει μία λίστα αγαθών που έχουν όνομα που ξεκινάει από την τιμή που έθεσε η χρήστης Maria ως όνομα αγαθού στην φόρμα αναζήτησης και τα οποία προσφέρονται για δωρεά.

- Η χρήστης Maria πλοηγείται σε σελίδα που περιέχει την λίστα αγαθών με όλα τα ιδιαίτερα χαρακτηριστικά τους

6.2.9.3 <u>Σενάριο3 :</u> Αναζήτηση Αγαθών που προσφέρονται για κοινοχρησία

Στην φόρμα αναζήτησης αγαθού, θέτουμε το πεδίο του ονόματος του αγαθού στην τιμή που μας προσδιορίζει τα αγαθά για τα οποία ενδιαφερόμαστε. Καθορίζουμε ότι θέλουμε να αναζητήσουμε αγαθά τα οποία προσφέρονται για κοινοχρησία. Πατάμε το κουμπί αναζήτησης και διαπιστώνουμε ότι:

- Η μέθοδος αναζήτησης επιστρέφει μία λίστα αγαθών που έχουν όνομα που ξεκινάει από την τιμή που έθεσε η χρήστης Maria ως όνομα αγαθού στην φόρμα αναζήτησης και τα οποία προσφέρονται για κοινοχρησία.
- Η χρήστης Maria πλοηγείται σε σελίδα που περιέχει την λίστα αγαθών με όλα τα ιδιαίτερα χαρακτηριστικά τους

6.2.9.4 <u>Σενάριο4 :</u> Αναζήτηση Αγαθών που ζητούνται για ανταλλαγή

Στην φόρμα αναζήτησης αγαθού, θέτουμε το πεδίο του ονόματος του αγαθού στην τιμή που μας προσδιορίζει τα αγαθά για τα οποία ενδιαφερόμαστε. Επιπλέον, καθορίζουμε ότι θέλουμε να βρούμε αγαθά τα οποία ζητούνται για ανταλλαγή. Πατάμε το κουμπί αναζήτησης και διαπιστώνουμε ότι:

- Η μέθοδος αναζήτησης επιστρέφει μία λίστα αγαθών που έχουν όνομα που ξεκινάει από την τιμή που έθεσε η χρήστης Maria ως όνομα αγαθού στην φόρμα αναζήτησης και τα οποία ζητούνται για ανταλλαγή.
- Η χρήστης Maria πλοηγείται σε σελίδα που περιέχει την λίστα αγαθών με όλα τα ιδιαίτερα χαρακτηριστικά τους

6.2.9.5 <u>Σενάριο5 :</u> Αναζήτηση Αγαθών που ζητούνται για δωρεά

Στην φόρμα αναζήτησης αγαθού, θέτουμε το πεδίο του ονόματος του αγαθού στην τιμή που μας προσδιορίζει τα αγαθά για τα οποία ενδιαφερόμαστε. Καθορίζουμε ότι θέλουμε να βρούμε αγαθά τα οποία ζητούνται για δωρεά. Πατάμε το κουμπί αναζήτησης και διαπιστώνουμε ότι:

- Η μέθοδος αναζήτησης επιστρέφει μία λίστα αγαθών που έχουν όνομα που ξεκινάει από την τιμή που έθεσε η χρήστης Maria ως όνομα αγαθού στην φόρμα αναζήτησης και τα οποία ζητούνται για δωρεά.

- Η χρήστης Maria πλοηγείται σε σελίδα που περιέχει την λίστα αγαθών με όλα τα ιδιαίτερα χαρακτηριστικά τους

6.2.9.6 <u>Σενάριο6 :</u> Αναζήτηση Αγαθών που ζητούνται για κοινοχρησία

Στην φόρμα αναζήτησης αγαθού, θέτουμε το πεδίο του ονόματος του αγαθού στην τιμή που μας προσδιορίζει τα αγαθά για τα οποία ενδιαφερόμαστε. Καθορίζουμε ότι θέλουμε να αναζητήσουμε αγαθά τα οποία ζητούνται για κοινοχρησία. Πατάμε το κουμπί αναζήτησης και διαπιστώνουμε ότι:

- Η μέθοδος αναζήτησης επιστρέφει μία λίστα αγαθών που έχουν όνομα που ξεκινάει από την τιμή που έθεσε η χρήστης Maria ως όνομα αγαθού στην φόρμα αναζήτησης και τα οποία ζητούνται για κοινοχρησία.
- Η χρήστης Maria πλοηγείται σε σελίδα που περιέχει την λίστα αγαθών με όλα τα ιδιαίτερα χαρακτηριστικά τους

Έχοντας διαπιστώσει την ορθή λειτουργία της αναζήτησης για όλες τις δυνατές τιμές των πεδίων της φόρμας αναζήτησης, επιδιώκουμε να διαπιστώσουμε και την ορθή συμπεριφορά της μεθόδου αναζήτησης όταν εναλλάσσεται η τιμή του ονόματος που θέτουμε προς αναζήτηση.

6.2.9.7 <u>Σενάριο7 :</u> Αναζήτηση Αγαθών με όνομα σε μικτούς (κεφαλαίους και μικρούς) χαρακτήρες

Θέτουμε το πεδίο της τιμής του ονόματος αγαθού σε αλφαριθμητική τιμή με μικρούς και κεφαλαίους χαρακτήρες επιδιώκοντας να διαπιστώσουμε ότι η αναζήτησή μας γίνεται επί του ονόματος του αγαθού με case-insensitive τρόπο. Πράγματι, λοιπόν, διαπιστώνουμε ότι:

- Η μέθοδος αναζήτησης επιστρέφει μία λίστα αγαθών που έχουν όνομα που ξεκινάει από την τιμή που έθεσε η χρήστης Maria ως όνομα αγαθού στην φόρμα αναζήτησης, χωρίς να ενδιαφέρει αν το όνομα που τέθηκε στην φόρμα αναζήτησης ήταν σε κεφαλαίους ή μικρούς χαρακτήρες. Η λίστα αγαθών είναι ίδια ακριβώς με την λίστα που θα επέστρεφε το σύστημα αν θέταμε ίδιο όνομα προς αναζήτηση αλλά με όλο κεφαλαίους ή όλο μικρούς χαρακτήρες.

- Η χρήστης Maria πλοηγείται σε σελίδα που περιέχει την λίστα αγαθών με όλα τα ιδιαίτερα χαρακτηριστικά τους

6.2.9.8 <u>Σενάριο8 :</u> Αναζήτηση Αγαθών με όνομα σε ελληνικούς χαρακτήρες

Θέτουμε το πεδίο της τιμής του ονόματος αγαθού σε αλφαριθμητική τιμή με ελληνικούς χαρακτήρες.

Διαπιστώνουμε ότι:

- Η μέθοδος αναζήτησης επιστρέφει μία λίστα αγαθών που έχουν όνομα που ξεκινάει από την τιμή που έθεσε η χρήστης Maria ως όνομα αγαθού στην φόρμα αναζήτησης. Τα αγαθά αυτά έχουν όνομα με πρόθεμα σε ελληνικούς χαρακτήρες ίσο με την τιμή που έθεσε η χρήστης Maria. Η λίστα αυτή δεν περιέχει αγαθά που έχουν όνομα με το ίδιο ακριβώς πρόθεμα αλλά σε λατινικούς χαρακτήρες.
- Η χρήστης Maria πλοηγείται σε σελίδα που περιέχει την λίστα αγαθών με όλα τα ιδιαίτερα χαρακτηριστικά τους

6.2.9.9 <u>Σενάριο9 :</u> Αναζήτηση Αγαθών με όνομα σε λατινικούς χαρακτήρες

Θέτουμε το πεδίο της τιμής του ονόματος αγαθού σε αλφαριθμητική τιμή με λατινικούς χαρακτήρες.

Διαπιστώνουμε ότι:

- Η μέθοδος αναζήτησης επιστρέφει μία λίστα αγαθών που έχουν όνομα που ξεκινάει από την τιμή που έθεσε η χρήστης Maria ως όνομα αγαθού στην φόρμα αναζήτησης. Τα αγαθά αυτά έχουν όνομα με πρόθεμα σε λατινικούς χαρακτήρες ίσο με την τιμή που έθεσε η χρήστης Maria. Η λίστα αυτή δεν περιέχει αγαθά που έχουν όνομα με το ίδιο ακριβώς πρόθεμα αλλά σε ελληνικούς χαρακτήρες.
- Η χρήστης Maria πλοηγείται σε σελίδα που περιέχει την λίστα αγαθών με όλα τα ιδιαίτερα χαρακτηριστικά τους

6.2.9.9 <u>Σενάριο9 :</u> Αναζήτηση Αγαθών με κενό όνομα

Σε αυτό το σενάριο, αφήνουμε κενή την τιμή του πεδίου ονόματος αγαθού στην φόρμα αναζήτησης.

Διαπιστώνουμε ότι:

- Η μέθοδος αναζήτησης επιστρέφει μία λίστα αγαθών με όλα τα αγαθά που έχουν χαρακτηριστικά που προσδιορίζονται από τα άλλα πεδία της φόρμας αναζήτησης. Συγκεκριμένα, εκτελέσαμε αυτό το σενάριο για αναζήτηση αγαθών που προσφέρονται για ανταλλαγή. Η λίστα που επιστρέφει η μέθοδος είναι λίστα με όλα τα αγαθά της βάσης δεδομένων του συστήματος που προσφέρονται για ανταλλαγή.
- Ο χρήστης πλοηγείται σε σελίδα που περιέχει την λίστα αγαθών με όλα τα ιδιαίτερα χαρακτηριστικά τους

6.2.10 Εκτέλεση Συναλλαγών

Οι επιτρεπτές συναλλαγές στο σύστημά μας είναι τρεις:

- Δωρεά
- Κοινοχρησία
- Ανταλλαγή

Η εναρκτήριος μίας συναλλαγής σηματοδοτείται από την έκφραση της πρόθεσης μίας οντότητας να συναλλαχθεί. Σε αυτό το εδάφιο, η οντότητα αναφέρεται είτε σε χρήστη είτε σε κοινότητα η οποία εκπροσωπείται από τον administrator της.

Η διαδικασία διεκπεραίωσης μίας συναλλαγής θα μπορούσε να χωριστεί σε 3 στάδια:

- Εκδήλωση Επιθυμίας Συναλλαγής
- Αποδοχή Συναλλαγής
- Εκτέλεση Συναλλαγής

Θα πρέπει να σημειώσουμε ότι η εκδήλωση επιθυμίας για συναλλαγή εκφράζεται επί αγαθού το οποίο είτε προσφέρεται είτε ζητείται. Για να καλύψουμε λοιπόν όλες τις περιπτώσεις χρήσης που αφορούν τις συναλλαγές στο σύστημά μας, προσεγγίσαμε τα σενάρια εκτέλεσης συναλλαγών με βάση τον τρόπο διάθεσης του αγαθού.

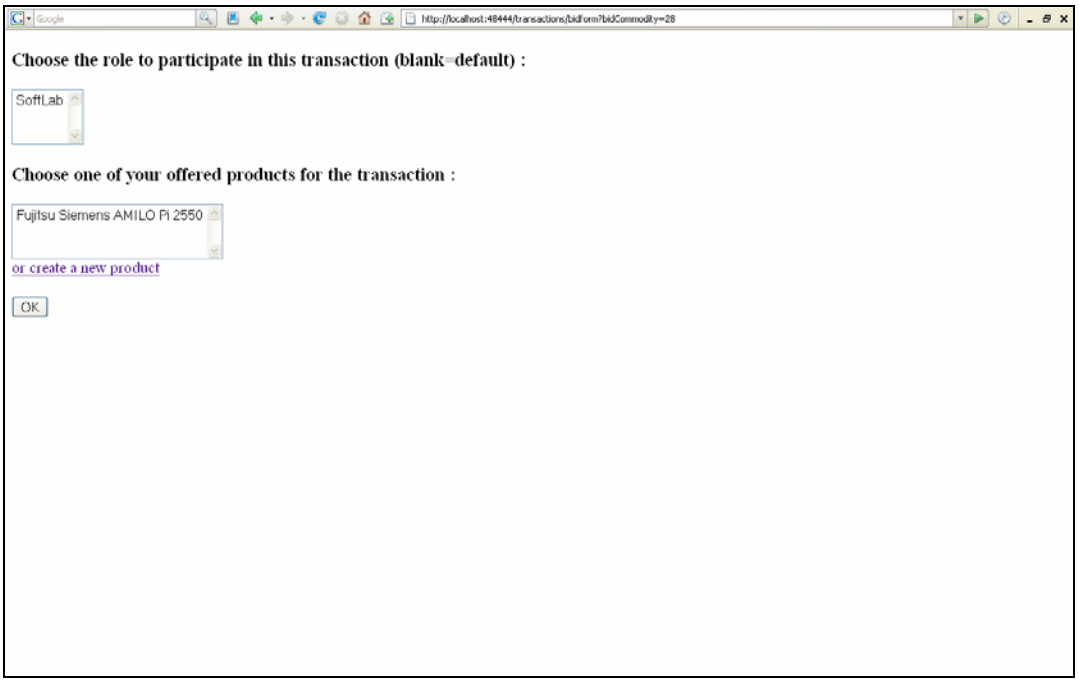
6.2.10.1 Η οντότητα Α ζητάει αγαθό C το οποίο προσφέρεται από την οντότητα Β

Η οντότητα Α έχει στην διάθεσή του μία λίστα με αγαθά. Εκφράζει την επιθυμία του να συναλλαχθεί, δηλώνοντας ότι επιθυμεί να αποκτήσει το αγαθό που προσφέρει η οντότητα Β.

6.2.10.1.1 <u>Σενάριο1:</u> Αγαθό που προσφέρεται για Ανταλλαγή

- *Εκδήλωση Επιθυμίας Συναλλαγής:*

Η οντότητα Α εκφράζει την επιθυμία της να συναλλαχθεί με την οντότητα Β, ζητώντας το αγαθό C και προσφέροντας ως αντάλλαγμα το αγαθό Κ, το οποίο της ανήκει.



Διαπιστώνουμε ότι:

- Δημιουργείται εγγραφή για αυτή τη συναλλαγή στον πίνακα xactions της βάσης δεδομένων μας. Το πεδίο state αυτής της εγγραφής τίθεται στην τιμή pending.
- Η οντότητα Β ενημερώνεται με mail για την επιθυμία της οντότητας Α για συναλλαγή

- *Αποδοχή Συναλλαγής:*

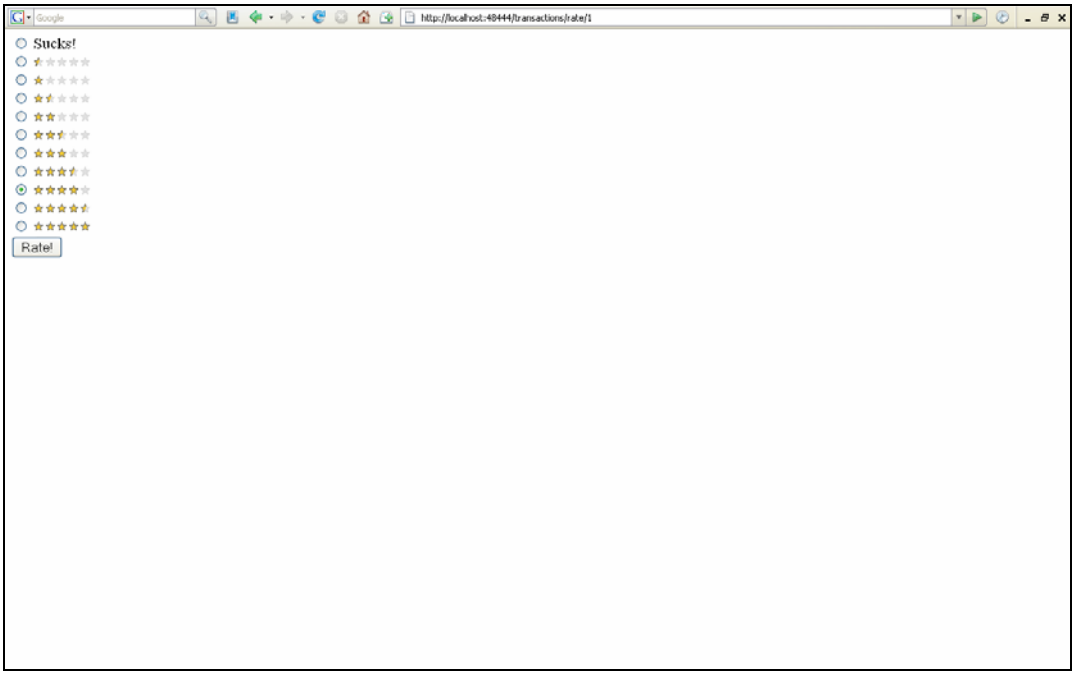
Η οντότητα Β πλοηγείται στην σελίδα όπου παρουσιάζονται οι συναλλαγές που την αφορούν. Από το πλήθος των συναλλαγών που εκκρεμούν για το αγαθό C, η οντότητα Β επιλέγει να συναλλαχθεί με την οντότητα Α, προσφέροντάς της το αγαθό C.

Διαπιστώνουμε ότι:

- Διαγράφονται από τον πίνακα xactions όλες οι εγγραφές που αφορούν τα αγαθά C και K, εκτός από την εγγραφή που αφορά την συναλλαγή ανάμεσα στην οντότητα A και την οντότητα B για το αγαθό C και το αγαθό K.
- Η τιμή του πεδίου state της εγγραφής του πίνακα xactions που αφορά την συναλλαγή ανάμεσα στην οντότητα A και την οντότητα B για το αγαθό C και το αγαθό K μεταβάλλεται από την τιμή pending στην τιμή executed.
- Η οντότητα A ενημερώνεται με mail για την αποδοχή της συναλλαγής.

- Εκτέλεση Συναλλαγής

Σε αυτό το σημείο λαμβάνει χώρα η συνάντηση των δύο μελών της συναλλαγής και τελικά η οντότητα B ανταλλάσσει το αγαθό C με το αγαθό K της οντότητα A. Έπειτα, μία από τις δύο οντότητες, έστω η A, βαθμολογεί την άλλη οντότητα για την συναλλαγή αυτή.



Διαπιστώνουμε ότι:

- Η τιμή του πεδίου state της εγγραφής του πίνακα xactions, η οποία αφορά την συναλλαγή, μεταβάλλεται από την τιμή executed στην τιμή half - rated.
- Διαγράφεται η εγγραφή του πίνακα offers που αφορά το αγαθό C.
- Διαγράφεται η εγγραφή του πίνακα offers που αφορά το αγαθό K.
- Αλλάζει ο ιδιοκτήτης του αγαθού C που σημαίνει ότι μεταβάλλεται το πεδίο owner_id της εγγραφής που αφορά το αγαθό C στον πίνακα commodities της βάσης δεδομένων μας.
- Αλλάζει ο ιδιοκτήτης του αγαθού K που σημαίνει ότι μεταβάλλεται το πεδίο owner_id της εγγραφής που αφορά το αγαθό K στον πίνακα commodities της βάσης δεδομένων μας.
- Ανανεώνεται ο βαθμός υπόληψης της οντότητας B καθώς και τα στοιχεία που αφορούν την δραστηριότητά της στον πίνακα reputations.
- Λαμβάνουν χώρα οι ανανεώσεις των βαθμών υπόληψης που επηρεάζονται από την ανανέωση του βαθμού υπόληψης της οντότητας B στον πίνακα reputations και στον πίνακα vactivity.

Για την ολοκλήρωση αυτού του σεναρίου, η οντότητα B βαθμολογεί την οντότητα A για αυτή τη συναλλαγή. Διαπιστώνουμε ότι:

- Η εγγραφή του πίνακα xactions που αφορά την συναλλαγή διαγράφεται.
- Δημιουργείται εγγραφή για αυτή τη συναλλαγή στον πίνακα swaps.

- Ανανεώνεται ο βαθμός υπόληψης της οντότητας Α καθώς και οι τιμές που αφορούν την δραστηριότητά της, στην αντίστοιχη εγγραφή του πίνακα reputations.
- Λαμβάνουν χώρα όλες οι ανανεώσεις στον πίνακα reputations και vactivities που προκύπτουν από την αλλαγή του βαθμού υπόληψης της οντότητας Α.

6.2.10.1.2 <u>Σενάριο2:</u> Αγαθό που προσφέρεται για Δωρεά

- *Εκδήλωση Επιθυμίας Συναλλαγής:*

Η οντότητα Α εκφράζει την επιθυμία της να συναλλαχθεί με την οντότητα Β, ζητώντας το αγαθό C. Διαπιστώνουμε ότι:

- Δημιουργείται εγγραφή για αυτή τη συναλλαγή στον πίνακα xactions της βάσης δεδομένων μας. Το πεδίο state αυτής της εγγραφής τίθεται στην τιμή pending.
- Η οντότητα Β ενημερώνεται με mail για την επιθυμία της οντότητας Α για συναλλαγή

- *Αποδοχή Συναλλαγής:*

Η οντότητα Β πλοηγείται στην σελίδα όπου παρουσιάζονται οι συναλλαγές που την αφορούν. Από το πλήθος των συναλλαγών που εκκρεμούν για το αγαθό C, η οντότητα Β επιλέγει να δωρίσει το αγαθό C στην οντότητα Α. Διαπιστώνουμε ότι:

- Διαγράφονται από τον πίνακα xactions όλες οι εγγραφές που αφορούν το συγκεκριμένο εκτός από την εγγραφή που αφορά την συναλλαγή ανάμεσα στην οντότητα Α και την οντότητα Β για το αγαθό C.
- Η τιμή του πεδίου state της εγγραφής του πίνακα xactions που αφορά την συναλλαγή ανάμεσα στην οντότητα Α και την οντότητα Β για το αγαθό C μεταβάλλεται από την τιμή pending στην τιμή executed.
- Η οντότητα Α ενημερώνεται με mail για την αποδοχή της συναλλαγής.

- Εκτέλεση Συναλλαγής

Σε αυτό το σημείο λαμβάνει χώρα η συνάντηση των δύο μελών της συναλλαγής και τελικά η οντότητα Β δωρίζει το αγαθό C στην οντότητα Α. Έπειτα, η οντότητα Α βαθμολογεί την συναλλαγή αυτή. Διαπιστώνουμε ότι:

- Η εγγραφή του πίνακα xactions, η οποία αφορά την συναλλαγή, διαγράφεται.
- Δημιουργείται εγγραφή στον πίνακα gifts που αφορά αυτή την συναλλαγή.

- Διαγράφεται η εγγραφή του πίνακα offers, η οποία αφορά το αγαθό C.
- Αλλάζει ο ιδιοκτήτης του αγαθού C που σημαίνει ότι μεταβάλλεται το πεδίο owner_id της εγγραφής που αφορά το αγαθό C στον πίνακα commodities της βάσης δεδομένων μας.
- Ανανεώνεται ο βαθμός υπόληψης και τα υπόλοιπα στοιχεία που αφορούν την δραστηριότητά της οντότητας B στον πίνακα reputations.
- Λαμβάνουν χώρα οι ανανεώσεις των βαθμών υπόληψης που επηρεάζονται από την ανανέωση του βαθμού υπόληψης της οντότητας B στους πίνακες reputations και vactivities.

6.2.10.1.3 <u>Σενάριο3:</u> Αγαθό που προσφέρεται για Κοινοχρησία

- *Εκδήλωση Επιθυμίας Συναλλαγής:*

Η οντότητα A εκφράζει την επιθυμία της να συναλλαχθεί με την οντότητα B, ζητώντας το αγαθό C. Διαπιστώνουμε ότι:

- Δημιουργείται εγγραφή για αυτή τη συναλλαγή στον πίνακα xactions της βάσης δεδομένων μας. Το πεδίο state αυτής της εγγραφής τίθεται στην τιμή pending.
- Η οντότητα B ενημερώνεται με mail για την επιθυμία της οντότητας A για συναλλαγή

- *Αποδοχή Συναλλαγής:*

Η οντότητα B πλοηγείται στην σελίδα όπου παρουσιάζονται οι συναλλαγές που την αφορούν. Από το πλήθος των συναλλαγών που εκκρεμούν για το αγαθό C, η οντότητα B επιλέγει να δωρίσει το αγαθό C στην οντότητα A. Διαπιστώνουμε ότι:

- Διαγράφονται από τον πίνακα xactions όλες οι εγγραφές που αφορούν το συγκεκριμένο εκτός από την εγγραφή που αφορά την συναλλαγή ανάμεσα στην οντότητα A και την οντότητα B για το αγαθό C.
- Η τιμή του πεδίου state της εγγραφής του πίνακα xactions που αφορά την συναλλαγή ανάμεσα στην οντότητα A και την οντότητα B για το αγαθό C μεταβάλλεται από την τιμή pending στην τιμή executed.
- Η οντότητα A ενημερώνεται με mail για την αποδοχή της συναλλαγής.

- Εκτέλεση Συναλλαγής

Σε αυτό το σημείο λαμβάνει χώρα η συνάντηση των δύο μελών της συναλλαγής και τελικά η οντότητα Β δωρίζει το αγαθό C στην οντότητα Α. Έπειτα, η οντότητα Α βαθμολογεί την συναλλαγή αυτή. Διαπιστώνουμε ότι:

- Η εγγραφή του πίνακα xactions, η οποία αφορά την συναλλαγή, διαγράφεται.
- Δημιουργείται εγγραφή στον πίνακα share που αφορά αυτή την συναλλαγή.
- Διαγράφεται η εγγραφή του πίνακα offers, η οποία αφορά το αγαθό C.
- Ανανεώνεται ο βαθμός υπόληψης και τα υπόλοιπα στοιχεία που αφορούν την δραστηριότητά της οντότητας Β στον πίνακα reputations.
- Λαμβάνουν χώρα οι ανανεώσεις των βαθμών υπόληψης που επηρεάζονται από την ανανέωση του βαθμού υπόληψης της οντότητας Β στους πίνακες reputations και vactivities.

6.2.10.2 Η οντότητα Α προσφέρει αγαθό το οποίο ζητείται από την οντότητα Β

Η οντότητα Α έχει στην διάθεσή του μία λίστα με αγαθά τα οποία ζητούνται. Εκφράζει την επιθυμία της να συναλλαχθεί, δηλώνοντας ότι θέλει να προσφέρει αγαθό C στην οντότητα Β, η οποία ζητάει αγαθό Κ.

6.2.10.1.1 <u>Σενάριο1:</u> Αγαθό που ζητείται για Ανταλλαγή

- *Εκδήλωση Επιθυμίας Συναλλαγής:*

Η οντότητα Α εκφράζει την επιθυμία της να συναλλαχθεί με την οντότητα Β, ζητώντας το αγαθό C και προσφέροντας ως αντάλλαγμα το αγαθό Κ, το οποίο της ανήκει. Διαπιστώνουμε ότι:

- Δημιουργείται εγγραφή για αυτή τη συναλλαγή στον πίνακα xactions της βάσης δεδομένων μας. Το πεδίο state αυτής της εγγραφής τίθεται στην τιμή pending.
- Η οντότητα Β ενημερώνεται με mail για την επιθυμία της οντότητας Α για συναλλαγή

- *Αποδοχή Συναλλαγής:*

Η οντότητα Β πλοηγείται στην σελίδα όπου παρουσιάζονται οι συναλλαγές που την αφορούν. Από το πλήθος των συναλλαγών που εκκρεμούν για το αγαθό C, η οντότητα Β επιλέγει να συναλλαχθεί με την οντότητα Α, προσφέροντάς της το αγαθό Μ. Διαπιστώνουμε ότι:

- Διαγράφονται από τον πίνακα xactions όλες οι εγγραφές που αφορούν τα αγαθά C, εκτός από την εγγραφή που αφορά την συναλλαγή ανάμεσα στην οντότητα A και την οντότητα B για το αγαθό C και το αγαθό K.
- Η τιμή του πεδίου state της εγγραφής του πίνακα xactions που αφορά την συναλλαγή ανάμεσα στην οντότητα A και την οντότητα B για το αγαθό C και το αγαθό K μεταβάλλεται από την τιμή pending στην τιμή semi - executed.
- Η οντότητα A ενημερώνεται με mail για την αποδοχή της συναλλαγής.

Η οντότητα A πλοηγείται στην σελίδα όπου παρουσιάζονται οι συναλλαγές που την αφορούν και από το πλήθος των συναλλαγών που εκκρεμούν για το αγαθό K, επιλέγει να συναλλαχθεί με την οντότητα B, λαμβάνοντας ως αντάλλαγμα το αγαθό M. Διαπιστώνουμε ότι:

- Διαγράφονται από τον πίνακα xactions όλες οι εγγραφές που αφορούν τα αγαθά K και M εκτός από την εγγραφή που αφορά την συναλλαγή μεταξύ των οντοτήτων A και B για αυτά τα αγαθά.
- Η τιμή του πεδίου state της εγγραφής του πίνακα xactions που αφορά την συναλλαγή ανάμεσα στην οντότητα A και την οντότητα B για το αγαθό C και το αγαθό K μεταβάλλεται από την τιμή semi – executed στην τιμή executed.
- Η οντότητα B ενημερώνεται με mail για την αποδοχή της συναλλαγής.

- Εκτέλεση Συναλλαγής

Σε αυτό το σημείο λαμβάνει χώρα η συνάντηση των δύο μελών της συναλλαγής και τελικά η οντότητα B ανταλλάσσει το αγαθό C με το αγαθό K της οντότητα A. Έπειτα, μία από τις δύο οντότητες, έστω η A, βαθμολογεί την άλλη οντότητα για την συναλλαγή αυτή. Διαπιστώνουμε ότι:

- Η τιμή του πεδίου state της εγγραφής του πίνακα xactions, η οποία αφορά την συναλλαγή, μεταβάλλεται από την τιμή executed στην τιμή half rated.
- Διαγράφεται η εγγραφή του πίνακα demands που αφορά το αγαθό C.
- Διαγράφεται η εγγραφή του πίνακα offers που αφορά το αγαθό K.
- Διαγράφεται η εγγραφή του πίνακα offers που αφορά το αγαθό M.
- Αλλάζει ο ιδιοκτήτης του αγαθού C που σημαίνει ότι μεταβάλλεται το πεδίο owner_id της εγγραφής που αφορά το αγαθό C στον πίνακα commodities της βάσης δεδομένων μας.

- Αλλάζει ο ιδιοκτήτης του αγαθού Μ που σημαίνει ότι μεταβάλλεται το πεδίο owner_id της εγγραφής που αφορά το αγαθό Μ στον πίνακα commodities της βάσης δεδομένων μας.
- Ανανεώνεται ο βαθμός υπόληψης της οντότητας Β.
- Λαμβάνουν χώρα οι ανανεώσεις των βαθμών υπόληψης που επηρεάζονται από την ανανέωση του βαθμού υπόληψης της οντότητας Β.

Για την ολοκλήρωση αυτού του σεναρίου, η οντότητα Β βαθμολογεί την οντότητα Α για αυτή τη συναλλαγή. Διαπιστώνουμε ότι:

- Η εγγραφή του πίνακα xactions που αφορά την συναλλαγή διαγράφεται.
- Δημιουργείται εγγραφή για αυτή τη συναλλαγή στον πίνακα swaps.
- Ανανεώνεται ο βαθμός υπόληψης της οντότητας Α καθώς και οι τιμές που αφορούν την δραστηριότητά της, στην αντίστοιχη εγγραφή του πίνακα reputations.
- Λαμβάνουν χώρα όλες οι ανανεώσεις στον πίνακα reputations και vactivities που προκύπτουν από την αλλαγή του βαθμού υπόληψης της οντότητας Α.

6.2.10.1.2 <u>Σενάριο2:</u> Αγαθό που ζητείται για Δωρεά

- *Εκδήλωση Επιθυμίας Συναλλαγής:*

Η οντότητα Α εκφράζει την επιθυμία της να συναλλαχθεί με την οντότητα Β, προσφέροντάς της ως δωρεά το αγαθό C, έναντι του αγαθού Κ το οποίο ζητάει η οντότητα Β. Διαπιστώνουμε ότι:

- Δημιουργείται εγγραφή για αυτή τη συναλλαγή στον πίνακα xactions της βάσης δεδομένων μας. Το πεδίο state αυτής της εγγραφής τίθεται στην τιμή pending
- Η οντότητα Β ενημερώνεται με mail για την επιθυμία της οντότητας Α για συναλλαγή

- *Αποδοχή Συναλλαγής:*

Η οντότητα Β πλοηγείται στην σελίδα όπου παρουσιάζονται οι συναλλαγές που την αφορούν. Από το πλήθος των συναλλαγών που εκκρεμούν για το αγαθό Κ, η οντότητα Β επιλέγει να δεχθεί την δωρεά από την οντότητα Α. Διαπιστώνουμε ότι:

- Διαγράφονται από τον πίνακα xactions όλες οι εγγραφές που αφορούν το αγαθό K και το αγαθό C, εκτός από την εγγραφή που αφορά την συναλλαγή ανάμεσα στην οντότητα A και την οντότητα B για το αγαθό C και το αγαθό K.
- Η τιμή του πεδίου state της εγγραφής του πίνακα xactions που αφορά την συναλλαγή ανάμεσα στην οντότητα A και την οντότητα B για το αγαθό C και K μεταβάλλεται από την τιμή pending στην τιμή executed.
- Η οντότητα A ενημερώνεται με mail για την αποδοχή της συναλλαγής.

- Εκτέλεση Συναλλαγής

Σε αυτό το σημείο λαμβάνει χώρα η συνάντηση των δύο μελών της συναλλαγής και τελικά η οντότητα A δωρίζει το αγαθό C στην οντότητα B. Έπειτα, η οντότητα B βαθμολογεί την συναλλαγή αυτή. Διαπιστώνουμε ότι:

- Η εγγραφή του πίνακα xactions, η οποία αφορά την συναλλαγή, διαγράφεται.
- Δημιουργείται εγγραφή στον πίνακα gifts που αφορά αυτή την συναλλαγή.
- Διαγράφεται η εγγραφή του πίνακα offers, η οποία αφορά το αγαθό C
- Διαγράφεται η εγγραφή του πίνακα demands, η οποία αφορά το αγαθό K.
- Αλλάζει ο ιδιοκτήτης του αγαθού C που σημαίνει ότι μεταβάλλεται το πεδίο owner_id της εγγραφής που αφορά το αγαθό C στον πίνακα commodities της βάσης δεδομένων μας.
- Ανανεώνεται ο βαθμός υπόληψης και τα υπόλοιπα στοιχεία που αφορούν την δραστηριότητά της οντότητας A στον πίνακα reputations.
- Λαμβάνουν χώρα οι ανανεώσεις των βαθμών υπόληψης που επηρεάζονται από την ανανέωση του βαθμού υπόληψης της οντότητας A.

6.2.10.1.3 <u>Σενάριο3:</u> Αγαθό που ζητείται για Κοινοχρησία

- *Εκδήλωση Επιθυμίας Συναλλαγής:*

Η οντότητα B ζητάει αγαθό K για κοινοχρησία και η οντότητα A δηλώνει την επιθυμία της να διαθέσει το αγαθό C, το οποίο κατέχει, για κοινοχρησία με την οντότητα B. Διαπιστώνουμε ότι:

- Δημιουργείται εγγραφή για αυτή τη συναλλαγή στον πίνακα xactions της βάσης δεδομένων μας. Το πεδίο state αυτής της εγγραφής τίθεται στην τιμή pending.
- Η οντότητα B ενημερώνεται με mail για την επιθυμία της οντότητας A για

συναλλαγή

- *Αποδοχή Συναλλαγής:*

Η οντότητα Β πλοηγείται στην σελίδα όπου παρουσιάζονται οι συναλλαγές που την αφορούν. Από το πλήθος των συναλλαγών που εκκρεμούν για το αγαθό Κ, η οντότητα Β επιλέγει να δεχθεί το αγαθό C από την οντότητα Α για κοινοχρησία. Διαπιστώνουμε ότι:

- Διαγράφονται από τον πίνακα xactions όλες οι εγγραφές που αφορούν τα αγαθά Κ και C εκτός από την εγγραφή που αφορά την συναλλαγή ανάμεσα στην οντότητα Α και την οντότητα Β για τα αγαθά C και Κ.
- Η τιμή του πεδίου state της εγγραφής του πίνακα xactions που αφορά την συναλλαγή ανάμεσα στην οντότητα Α και την οντότητα Β για το αγαθό C μεταβάλλεται από την τιμή pending στην τιμή executed.
- Η οντότητα Α ενημερώνεται με mail για την αποδοχή της συναλλαγής.

- Εκτέλεση Συναλλαγής

Έπειτα, η οντότητα Β βαθμολογεί την οντότητα Α. Διαπιστώνουμε ότι:

- Η εγγραφή του πίνακα xactions, η οποία αφορά την συναλλαγή, διαγράφεται.
- Δημιουργείται εγγραφή στον πίνακα share που αφορά αυτή την συναλλαγή.
- Διαγράφεται η εγγραφή του πίνακα demands που αφορά το αγαθό C
- Διαγράφεται η εγγραφή του πίνακα offers, η οποία αφορά το αγαθό Κ.
- Ανανεώνεται ο βαθμός υπόληψης και τα στοιχεία που αφορούν την δραστηριότητα της οντότητας Α στον πίνακα reputations.
- Λαμβάνουν χώρα οι ανανεώσεις των βαθμών υπόληψης που επηρεάζονται από την ανανέωση του βαθμού υπόληψης της οντότητας Α.

7

Επίλογος

7.1 Σύνοψη και συμπεράσματα

Το σύστημα που παρουσιάσαμε στην διπλωματική μας μπορεί να εφαρμοστεί πρακτικά σε ένα σύνολο ομάδων χρηστών, όπως για παράδειγμα ένας πανεπιστημιακός χώρος, για να μοντελοποιήσει την μεταξύ τους αλληλεπίδραση. Μπορούν να το χρησιμοποιήσουν ως ένα ευέλικτο και αποτελεσματικό εργαλείο για την ολοκλήρωση των συναλλαγών τους παρέχοντας τους αξιοπιστία και ασφάλεια.

Παρόμοια συστήματα συναλλαγών που δραστηριοποιούνται στον χώρο του Διαδικτύου συστηματοποιούν τον τρόπο εκτέλεσης μιας συναλλαγής. Με βοηθό την σύγχρονη τεχνολογία πλαισιώσαμε την φυσική διαδικασία αντιπραγματισμού με όρους και κανόνες, με αποτέλεσμα να μεγιστοποιήσουμε την απόδοση της.

Η αξία των προϊόντων και των υπηρεσιών δεν ορίζεται από ένα απόλυτα καθορισμένο μετρικό σύστημα. Η διαχείριση των προϊόντων και υπηρεσιών αποδίδει την τελική τους αξία, η οποία μάλιστα μεταβάλλεται, λόγω της ακολουθίας των συναλλαγών οι οποίες επενεργούν επί αυτών. Χαρακτηριστικό παράδειγμα αποτελεί η περίπτωση ενός ατόμου να αρχίσει με την κατοχή ενός σπιρτόκουτου και μέσα από αλλεπάλληλες ανταλλαγές να αποκτήσει ένα σπίτι.

Είδαμε επίσης πόσο σύνθετο θέμα αποτελεί η προσπάθεια να προσδώσουμε στους χρήστες έναν δείκτη που να εκφράζει την αξιοπιστία τους απέναντι στο σύστημά μας. Αν προσθέσουμε και την συμμετοχή τους σε κοινότητες βλέπουμε πως υπάρχει μια συνεχής

αλληλεπίδραση μεταξύ της μονάδας και του συνόλου, την οποία προσπαθήσαμε να αποτυπώσουμε όσο καλύτερα γίνεται. Απώτερος στόχος μας ήταν το σύστημα της υπόληψης να λειτουργήσει ως κίνητρο για την περαιτέρω συμμετοχή των χρηστών στη VCommunity μέσω συναλλαγών που θα βελτιώνουν τον ατομικό τους βαθμό καθώς και των κοινοτήτων στις οποίες ανήκουν.

Κανένα σύστημα βέβαια δεν μπορεί να επιτύχει το σκοπό του αν δεν υπάρχουν πρόθυμα μέλη να συμμετάσχουν ενεργά, καταχωρώντας στην περίπτωσή μας αγαθά και συμμετέχοντας σε κοινότητες. Βασιζόμαστε δηλαδή στην βούληση των χρηστών για την επιτυχία της εφαρμογής μας, έχοντας την όμως εξοπλίσει με ένα μεγάλο και σταθερό σύστημα που μας παρέχει ο δομημένος προγραμματισμός.

7.2 Μελλοντικές επεκτάσεις

- Δημιουργία προσωπικού blog για κάθε χρήστη
- Δημιουργία forum/chat room.
- Περαιτέρω εξατομίκευση της σελίδας στις προτιμήσεις του χρήστη(Profiling)
- Εργαλείο αυτόματου matching demands και offers.
- Προσθήκη AJAX features
- Μελέτη της απόδοσης της Βάσης Δεδομένων

8

Βιβλιογραφία

[1]. Thomas, Dave and Hanson ,David Heinemeier. Agile Web Development with Rails 2nd ed., Raleigh, North Carolina :The Pragmatic Bookshelf, Version: 2006-11-22

[2]. Thomas, Dave. Programming Ruby The Pragmatic Programmers' Guide 2nd ed., Raleigh, North Carolina : The Pragmatic Bookshelf, Version: 2004-9-30

[3]. Black, David A., foreword by Hanson ,David Heinemeier. Ruby for Rails Ruby Techniques for Rails Developers, Manning Publications Version: 2006

[4]. Silberschatz, Abraham, Henry F. Korth, and Sudarshan S., Database Systems Concepts 4th ed. Trans Γκλαβά Μαίρη, New York: The MacGraw-Hill Companies Inc. Version : 2002

[5]. Wikipedia The Free Encyclopedia <http://en.wikipedia.org/wiki/Web_2.0>

[6]. Wikipedia The Free Encyclopedia <http://en.wikipedia.org/wiki/Social_networks>

[7]. Wikipedia The Free Encyclopedia <http://en.wikipedia.org/wiki/Reputation>

[8]. Wikipedia The Free Encyclopedia <http://en.wikipedia.org/wiki/Barter>

[9]. Wikipedia The Free Encyclopedia <http://en.wikipedia.org/wiki/Ruby_on_rails>

[10]. Wikipedia The Free Encyclopedia <http://en.wikipedia.org/wiki/Motivations_for_contributing_to_online_communities>

[11]. <http://www.rubyonrails.org/>

[12]. eBay <http://www.ebay.com>

[13]. iKarma <http://www.ikarma.com>

[14]. RapLeaf <http://www.rapleaf.com>

[15]. FaceBook <http://www.facebook.com>

[16]. MySpace <http://www.myspace.com>

[17]. SwapThing <http://www.swapthing.com>