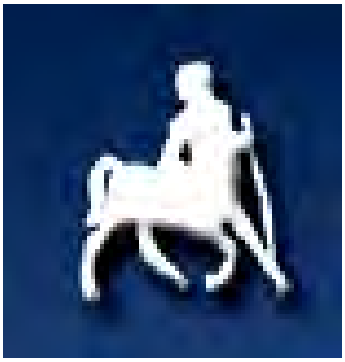
ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΗΛΕΚΤΡΟΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ ΚΑΙ ΔΙΚΤΥΩΝ



Θέμα εργασίας:

Ανάπτυξη web-based εφαρμογής αντιπραγματισμού. Μελέτη αλγορίθμων αξιολόγησης.

ΟΝΟΜΑ: ΦΩΤΗ ΜΑΓΔΑΛΗΝΗ

ΕΠΙΒΛΕΠΟΝΤΕΣ ΚΑΘΗΓΗΤΕΣ: ΠΑΠΑΔΑΚΗΣ ΝΙΚΟΛΑΟΣ

ΣΤΑΜΟΥΛΗΣ ΓΕΩΡΓΙΟΣ

Περιεχόμενα

1 ΓΕΝΙΚΗ ΠΡΟΣΕΓΓΙΣΗ....3
2 Αντιπραγματισμός και οικονομία....4
2.1 Από τον αντιπραγματισμό στην εγχρήματη οικονομία....4
2.2 Από την εγχρήματη οικονομία στον αντιπραγματισμό....8
3 Βάση Δεδομένων....17
3.1 Εννοιολογικό Μοντέλο....17
3.2 Φυσικό Μοντέλο....25
4 ΑΛΓΟΡΙΘΜΟΙ ΑΞΙΟΛΟΓΗΣΗΣ ΠΟΥ ΧΡΗΣΙΜΟΠΟΙΟΥΝΤΑΙ....28
4.1 E-bay....28
4.2 P2P ΔΙΚΤΥΑ....29
5 ΑΛΓΟΡΙΘΜΟΣ ΑΞΙΟΛΟΓΗΣΗΣ....33
5.1 Πως υπολογίζεται ο βαθμός αξιοπιστίας των μελών....33
5.2 Ποσοστό αξιοπιστίας....33
6 Περιγραφή του συστήματος....34
7 Αρχιτεκτονική και περιγραφή συστήματος....43
7.1 Ανάλυση Επιπέδων....43
7.2 Αρχιτεκτονική και Εφαρμογές....47
8 Υλοποίηση συστήματος....48
Παράρτημα Α....50
Παράρτημα Β....55
Παράρτημα Γ....73
9 ΒΙΒΛΙΟΓΡΑΦΙΑ....81

1 ΓΕΝΙΚΗ ΠΡΟΣΕΓΓΙΣΗ

Στόχος της εργασίας αυτής είναι η ανάπτυξη ενός συστήματος συναλλαγής αγαθών στον παγκόσμιο ιστό. Το σύστημα αυτό θα λειτουργεί με βάση ένα σύστημα ανταλλακτικής οικονομίας, όπου οι συναλλαγές γίνονται αποκλειστικά με την ανταλλαγή αγαθών. Τα αγαθά που ανταλλάσσονται μπορεί να είναι είτε προϊόντα είτε υπηρεσίες.

Τα άτομα που εγγράφονται στο σύστημα μπορούν να δημιουργούν «κοινωνίες», αυτό μπορεί να γίνει για παράδειγμα είτε λόγω κάποιου κοινού ενδιαφέροντος π.χ. φίλοι της jazz μουσικής είτε για γεωγραφικούς λόγους.
Οι συναλλαγές εντάσσονται σε τρεις κατηγορίες:

Ανταλλαγή, όπου δύο χρήστες επιλέγουν να ανταλλάξουν δύο αντικείμενα που έχουν στην κατοχή τους.

Δωρεά, ένας χρήστης επιλέγει να δωρίσει ένα αντικείμενο ή μια υπηρεσία σε έναν άλλο χρήστη.

Κοινοχρησία, δύο χρήστες συμφωνούν να χρησιμοποιούν από κοινού ένα αγαθό.

Στις συναλλαγές μπορούν να συμμετέχουν και απλοί χρήστες του συστήματος αλλά και κοινότητες. Για παράδειγμα ένας χρήστης που έτυχε να κερδίσει ένα εισιτήριο για μία συναυλία, την οποία όμως δεν ενδιαφέρεται να παρακολουθήσει, μπορεί να επιλέξει να το χαρίσει σε μία κοινότητα την οποία έχουν δημιουργήσει θαυμαστές του καλλιτέχνη που δίνει τη συναυλία.

Ένα ακόμη σημαντικό συστατικό της κοινωνίας αυτής είναι η υπόληψη των μελών της. Εφόσον δεν επιτρέπονται οι συναλλαγές με βάση το χρήμα θα πρέπει κάθε εμπλεκόμενο μέλος της συναλλαγής να γνωρίζει με τι άτομο συναλλάσσεται και πως έχει συμπεριφερθεί το άτομο αυτό έως τώρα στα πλαίσια της κοινωνίας.

2 Αντιπραγματισμός και οικονομία

2.1 Από τον αντιπραγματισμό στην εγχρήματη οικονομία

Στην οικονομία μιας πρωτόγονης κοινωνίας τα άτομα παράγουν προϊόντα με σκοπό την ιδιοκατανάλωση ή την ιδιοχρησιμοποίηση. Επικρατεί η φυσική παραγωγή, παραγωγή δηλαδή αγαθών που δεν προορίζονται για ανταλλαγή. Αντίθετα η παραγωγή αγαθών για πούλημα στην αγορά, για ανταλλαγή λέγεται εμπορευματική παραγωγή.(Τέτοιες πρωτόγονες κοινότητες εξακολουθούσαν να υπάρχουν ακόμα και πολύ τελευταία σε πολλές απόμερες γωνιές της γης, που δεν επηρεάστηκαν από τις πιο εξελιγμένες χώρες. Η πίεση της ευρωπαϊκής αστικής τάξης, που άρπαξε όλες αυτές της γωνιές της γης, φυσικά κατέστρεψε αυτόν τον τρόπο οργάνωσης). Ο καταμερισμός της εργασίας ήταν ολότελα πρωτόγονος και η εργασία βασικά μοιραζόταν ανάμεσα στα δύο φύλα. Όσο όμως τα άτομα συναναστρέφονται μεταξύ τους και απομακρύνονται από τα όρια της επικράτειάς τους, ανακαλύπτουν ότι υπάρχει ένας αριθμός προϊόντων που δεν έχουν τη δυνατότητα να παράγουν αλλά ταυτόχρονα τους είναι αναγκαία και χρήσιμα. Έτσι μέσα απ' αυτή τη διαδικασία προκύπτει η ανάγκη της ανταλλαγής και μαζί μ' αυτήν η εξειδίκευση και ο καταμερισμός εργασίας στην κοινωνία.

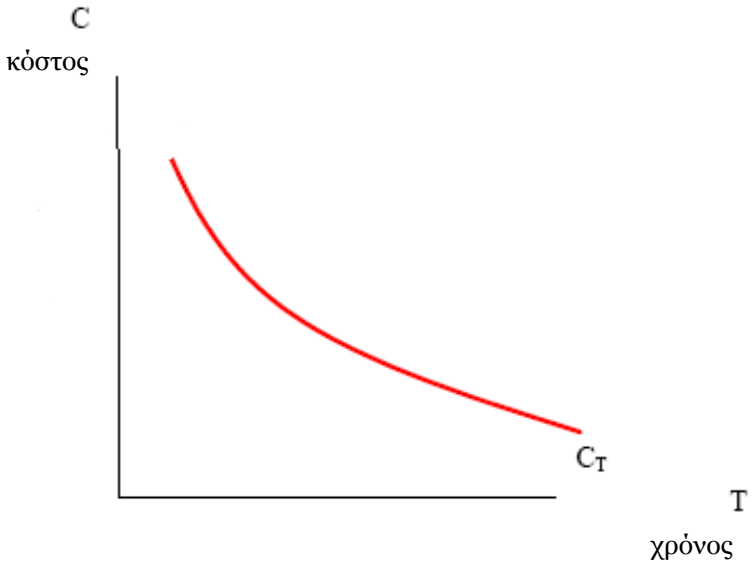
Η εξημέρωση των ζώων οδηγεί στο χωρισμό των κτηνοτροφικών φυλών από τις άλλες μάζες των ομάδων γενών στην πρωτόγονη κοινωνία. Αυτός είναι ο πρώτος μεγάλος κοινωνικός καταμερισμός εργασίας και τα κτηνοτροφικά προϊόντα αποτελούν τη βάση για την ανταλλαγή προϊόντων ανάμεσα στις φυλές.

Με την ταυτόχρονη αύξηση του πληθυσμού γίνονται τα πρώτα βήματα της γεωργίας. Όσο ο άνθρωπος μαθαίνει καινούργιες μορφές και μεθόδους εργασίας, τόσο μεγαλύτερος γίνεται και ο καταμερισμός εργασίας και πλαταίνει πολύ η βάση για την ανάπτυξη της ανταλλαγής.

Ο καταμερισμός εργασίας δημιουργεί πλεονάζοντα προϊόντα, που τα άτομα που τα παράγουν τα ανταλλάσουν με άλλα χρήσιμα για τον εαυτό τους ή παράγουν αποκλειστικά προϊόντα για ανταλλαγή. Η οικονομία που οι συναλλαγές γίνονται αποκλειστικά με ανταλλαγή προϊόντων είναι γνωστή σαν **ανταλλακτική** οικονομία και η διαδικασία της ανταλλαγής καλείται **αντιπραγματισμός**. Ο νόμος της προσφοράς και ζήτησης καθορίζει πλέον τις τιμές ανταλλαγής των προϊόντων οι οποίες είναι γνωστές και σαν σχετικές τιμές.(η αξία ενός προϊόντος εκφράζεται με την αξία χρήσης ενός άλλου που χρησιμεύει σαν ισοδύναμό του σ54 Λεοντίεφ) Για να πραγματοποιηθεί όμως μια συναλλαγή απαιτείται να υπάρξει διπλή σύμπτωση των αναγκών μεταξύ των ενδιαφερόμενων μερών και όταν υπάρξει τότε μιλάμε για **άμεσο αντιπραγματισμό.** Χωρίς όμως εξειδίκευση, γνώση και πληροφόρηση της αγοράς που συνοδεύεται με ένα κόστος αναζήτησης καθίστανται αδύνατες οι συναλλαγές προϊόντων παραγόμενων με σκοπό τον αντιπραγματισμό. Σε μικρές κοινωνίες η γνώση είναι επαρκής και το κόστος χαμηλό και απ' αυτήν την άποψη ο αντιπραγματισμός συνέβαλε στον καταμερισμό και εξειδίκευση της εργασίας.(Όταν η παραγωγή έχει ακόμα πιο πολύ φυσικό χαρακτήρα και η ανταλλαγή γίνεται τυχαία, έχουμε τη στοιχειώδικη, απλή ή τυχαία μορφή της αξίας

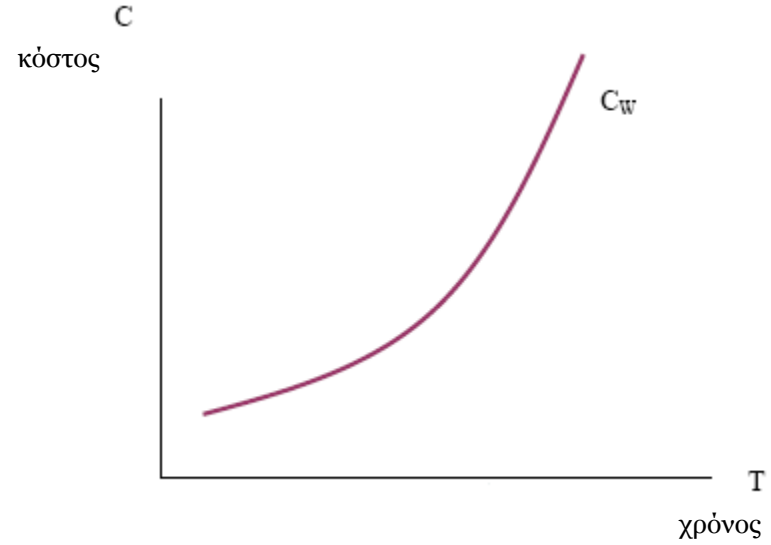
Όσο οι κοινωνίες διευρύνονται, συμμετέχουν όλο και περισσότερα άτομα στην παραγωγική διαδικασία, διογκώνονται οι συναλλαγές, τόσο παράλληλα αυξάνεται ο αριθμός των σχετικών τιμών και το κόστος αναζήτησης και επινοείται ο **έμμεσος αντιπραγματισμός** ως μηχανισμός κατανομής των αγαθών.
Στην περίπτωση του έμμεσου πραγματισμού η επιθυμητή συναλλαγή εμπεριέχει ένα κόστος που συνδέεται με την αναζήτηση των αγαθών και τη συλλογή πληροφοριών και θεωρείται ένα είδος φόρου συναλλαγής. Αν εμφανιστεί ένας ενδιάμεσος το παραπάνω κόστος θα μειωθεί σημαντικά και θα μπορούσε να εξομοιωθεί με το δημοπράτη της βαλρασιανής αγοράς σύμφωνα με την άποψη του Laidler (1990). Ο ενδιάμεσος θέτει τις τιμές ισορροπίας στην αγορά, πληροφορεί γι' αυτές όσους συμμετέχουν και φέρνει κοντά τους παραγωγούς ώστε να πραγματοποιηθούν οι συναλλαγές. Αναλαμβάνει δηλαδή το ρόλο των οργανωμένων κυκλωμάτων διανομής της αναπτυγμένης κοινωνίας. Δεν μηδενίζεται βέβαια η αβεβαιότητα διάθεσης του προϊόντος και εξεύρεσης αγοραστή η οποία χαρακτηρίζει κάθε οικονομική συμπεριφορά.
Αν δεν υπήρχε η αβεβαιότητα και η λήψη πληροφοριών διενεργούνταν χωρίς χρονικό και οικονομικό κόστος, τότε ο έμμεσος αντιπραγματισμός θα αποτελούσε άριστη λύση για την οικονομία.
Η δημιουργία **εκθετηρίων ανταλλαγής** όπου σε κάθε περίπτερο ανταλλάσσεται ένα αγαθό έναντι μιας αναλογίας με κάποιο άλλο, αποτελεί μια περαιτέρω βελτίωση στα πλαίσια της οργανωμένης αγοράς. Στην περίπτωση αυτή αν ο αριθμός των συναλλασσόμενων και επομένως και των προϊόντων τους είναι *ν, τότε* απαιτούνται *ν(ν-1)/2* εκθετήρια. Το κόστος βέβαια αναζήτησης καθώς και η αβεβαιότητα μειώνονται σημαντικά, αλλά είναι υψηλός ο αριθμός των σχετικών τιμών.
Με την εισαγωγή του χρήματος στην κοινωνία ως κοινά αποδεκτό μέσο μειώνονται σημαντικά περιορισμοί χρόνου και πληροφόρησης. Δεν απαιτεί την ύπαρξη διπλής σύμπτωσης αναγκών, απλοποιεί το πρόβλημα του συντονισμού και εξαλείφει την ύπαρξη πολλών σχετικών τιμών με την εμφάνιση χρηματικών ή απόλυτων τιμών. Σε μια οικονομία με *ν* αγαθά, συμπεριλαμβανομένου και του χρήματος, διαμορφώνονται *ν-1* απόλυτες τιμές. (Γενικό ισοδύναμο Με τον όρο χρήμα εννοούμε κάθε αγαθό που έχει τη γενική αποδοχή του μέσου συναλλαγών και διακανονισμού χρεών.)

Παρ' όλα αυτά εξακολουθούν να υπάρχουν προβλήματα που συνδέονται τόσο με την αβεβαιότητα που εξακολουθεί να υφίσταται, όσο και με το συνολικό κόστος ως άθροισμα του κόστους συναλλαγών και του κόστους αναμονής. Σαν κόστος συναλλαγών, Ct, ορίζεται αυτό το οποίο συνδέεται με την αναζήτηση και τη διαπραγμάτευση του εμπορεύματος. Είναι γνωστό και ως φόρος εμπορίου, είναι ανάλογο του αριθμού των συναλλαγών σε δεδομένη χρονική διάρκεια και δεν εξαρτάται από το μέγεθος της συναλλαγής. Όσο λιγότερο συναλλάσσεται κάποιος μέσα σε μια χρονική περίοδο τόσο λιγότερο και το κόστος συναλλαγής. Αν δεν υπάρξουν καθόλου συναλλαγές για ένα μεγάλο χρονικό διάστημα το κόστος θα είναι μηδενικό. Αντίθετα, αν σε μικρό χρονικό διάστημα πραγματοποιούνται πολλαπλές συναλλαγές το κόστος αυξάνει δραματικά. Το χρονικό αυτό διάστημα, Τα, ορίζεται ως περίοδος συναλλαγών. Διαγραμματικά το κόστος συναλλαγών παρουσιάζεται στο σχήμα 1 παρακάτω:



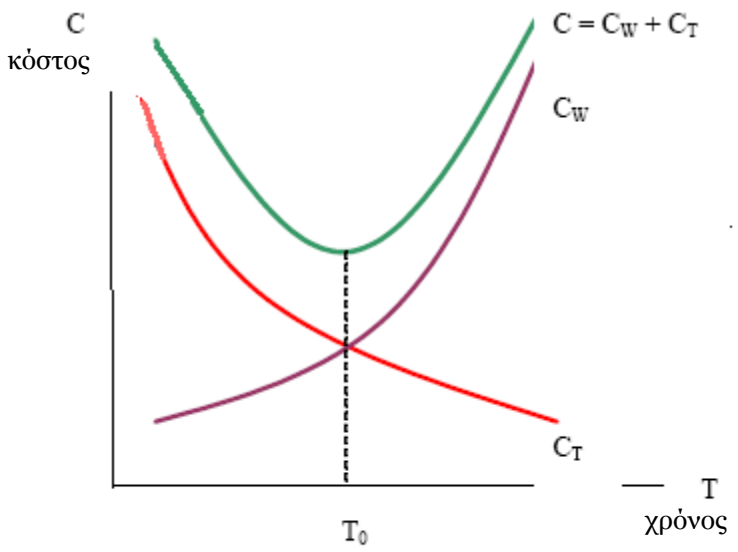
Σχ. 1 Καμπύλη κόστους συναλλαγών

Το κόστος αναμονής, Cw, είναι αποτέλεσμα της αποχής από τις συναλλαγές και διακρίνεται σε αντικειμενικό με την έννοια π.χ του κόστους συσσώρευσης αποθεμάτων και υποκειμενικό με την έννοια της αποχής από την κατανάλωση. Αυτό το κόστος είναι ανάλογο της περιόδου συναλλαγών και παρουσιάζεται διαγραμματικά στο σχήμα 2 .



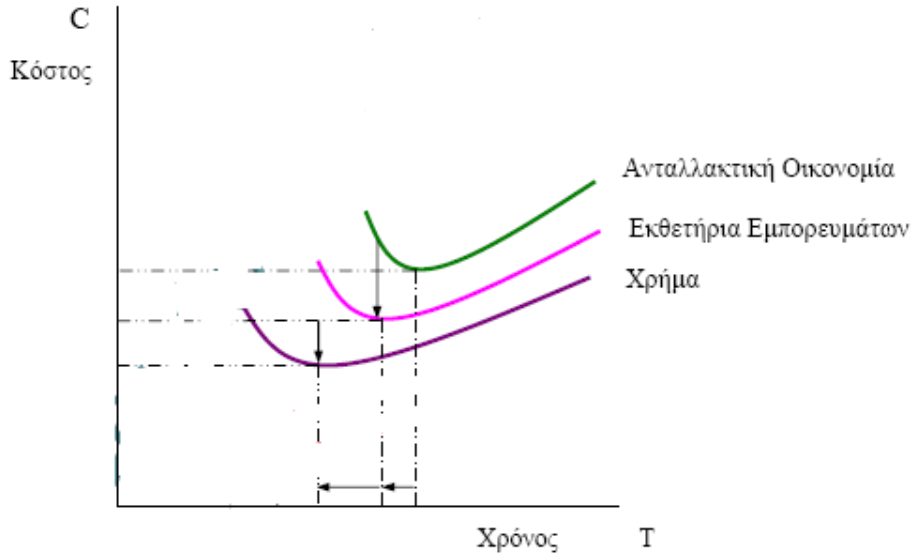
Σχ. 2 Καμπύλη κόστους αναμονής.

Η σύνθεση των στοιχείων κόστους συναλλαγών και αναμονής συντίθεται και απεικονίζεται στο σχήμα 3



Σχ. 3 Καμπύλη συνολικού κόστους συναλλαγών, C=Cw +Ct

Η μετατόπιση των καμπυλών κόστους για κάθε φάση νεωτερισμού που εμφανίζεται στην οικονομία, δηλαδή από το στάδιο του άμεσου αντιπραγματισμού (ανταλλακτική οικονομία) μέχρι την κυκλοφορία του χρήματος στην οικονομία, παρουσιάζεται στο σχήμα 4.



Σχ. 4 Κέρδη για κάθε φάση νεωτερισμού στην οικονομία.

2.2 Από την εγχρήματη οικονομία στον αντιπραγματισμό

Σε οικονομίες όπου καθιερώθηκε η εγχρήματη οικονομία δεν λείπουν και προσπάθειες επιστροφής στον αντιπραγματισμό με διαφορετικούς προσανατολισμούς κάθε φορά. Αποτελούν βέβαια μεμονωμένες καταστάσεις και δεν μπορούν να λάβουν καθολικό χαρακτήρα.
Μια τέτοια περίπτωση αποτελούν ομάδες νεαρών ατόμων ή και ολόκληρων οικογενειών που δε δέχονται τους σύγχρονους κανόνες και αναζητούν έναν πιο σύγχρονο τρόπο ζωής. Ονομάζονται Ελφ (τα ξωτικά από το μυθιστόρημα του Τόλκιν *Ο Άρχοντας των Δακτυλιδιών*-, πλάσματα με απίστευτη μακροβιότητα) του Μεγάλου Φαραγγιού, Σκαφτιάδες χωρίς αφεντικό ή Νταμανχούρ και έχουν ένα και μόνο στόχο, να βγουν από την απρόσωπη καθημερινότητα και να βρουν μια νέα ταυτότητα. Ζουν σε πρωτόγονες καλύβες στα δάση ή σε παλιούς οικισμούς και εγκαταλειμμένες αγροικίες. Μερικές φορές αρνούνται την παροχή νερού ή ρεύματος, ακόμα και τα χρήματα, επιμένοντας στην ανταλλαγή αγαθών.Τους συναντά κανείς όχι στην Καλιφόρνια ή σε κάποια μακρινή βουνοκορφή επαρχίας του Θιβέτ, αλλά εδώ στην Ευρώπη, στην καρδιά των πόλεων, στην ύπαιθρο, ακόμα και στα ελληνικά νησιά.

Εμπνευσμένο από τους Ελφ είναι το Δίκτυο Οικολογικών Χωριών (RIVE) στην Ιταλία με νατουραλιστική τάση. Δημιουργήθηκαν τον Ιούλιο του 1980 στα περίχωρα της Σαμπούκα στα Απέννινα, μεταξύ Τοσκάνης και Εμίλια. Είναι μια αναρχική κοινότητα με περίπου εκατό μέλη, ενηλίκους και παιδιά, που ήρθαν ακόμα κι από το εξωτερικό. Οι Ελφ ζουν σαν παλιοί αγρότες. Αντί για τρακτέρ χρησιμοποιούν ζώα και αντί για φυτοφάρμακα μεικτές καλλιέργειες. Ο τρόπος ζωής τους μοιάζει πολύ μ' εκείνον των χωρικών στα τέλη του 19ου αιώνα. Ζουν σε μια κατάσταση εθελοντικής λιτότητας και συνειδητής φτώχιας, σε σπίτια χωρίς παροχή νερού και ηλεκτρικού, τα οποία θερμαίνονται με ξύλα και φωτίζονται με κεριά. Δεν αποδέχονται την ατομική ιδιοκτησία. Τα αγαθά ανήκουν σε όλους και το εμπόριο γίνεται με αντιπραγματισμό - το χρήμα έχει αντικατασταθεί με αλεύρι από κάστανα, κύριο ανταλλακτικό αγαθό τους. Δεν αρνούνται όμως τα δώρα σε τρόφιμα και ποτά που φέρνουν μαζί τους οι πολυάριθμοι επισκέπτες της κοινότητας.

Πραγματικά οι ιδέες για εναλλακτικούς τρόπους συναλλαγών είναι πάρα πολλές.

Α) Στις Αγγλοσαξονικές χώρες (και όχι μόνο) χρησιμοποιούν τα **Τοπικά Συστήματα Ανταλλακτικού Εμπορίου (Local Exchange Trading Systems – LETS),** τα οποία είναι δίκτυα πολυμερών ανταλλαγών. Κατά βάση έχουν τοπικό χαρακτήρα, λειτουργούν σε επίπεδο γειτονιάς, συνοικίας, χωριού ή πόλης, σε κάποιες περιπτώσεις όμως πολλά δίκτυα ανήκουν σε ένα άλλο μεγαλύτερο δίκτυο, το οποίο επιτρέπει σε μέλη από διαφορετικά δίκτυα να συναλλάσσονται μεταξύ τους. Η λογική ενός Τοπικού Συστήματος Ανταλλακτικού Εμπορίου είναι απλή:

1) Για να συμμετέχει κανείς πρέπει να είναι μέλος, δηλαδή χρειάζεται να εγγραφεί στον κατάλογο των συναλλασσόμενων, να δηλώσει τι προϊόντα και υπηρεσίες προσφέρει και τι χρειάζεται και να ανοίξει λογαριασμό στο Σύστημα.

Τα μέλη έχουν πρόσβαση στον κατάλογο με τα προσφερόμενα προϊόντα και υπηρεσίες και επιλέγουν από εκεί με ποια μέλη θα συναλλαγούν.

2) Οι συναλλαγές γίνονται με τον τρόπο που κάθε Σύστημα αποφασίζει με συλλογικές διαδικασίες. Συνήθως, οι διαμόρφωση των τιμών είναι ελεύθερη μεταξύ των μελών και χρησιμοποιείται μια συμβολική μονάδα (τις περισσότερες φορές την ονομάζουν με κάποιο όνομα που να θυμίζει τον τόπο όπου λειτουργεί το Σύστημα, ή το βασικό προϊόν της περιοχής) για τη μέτρηση της αξίας της συναλλαγής, υπάρχουν όμως και Συστήματα όπου οι τιμές καθορίζονται συλλογικά ή ακόμη χρησιμοποιούν ως μονάδα μέτρησης την εργατοώρα, ώστε κάθε προϊόν και υπηρεσία να αξίζει όσο χρόνο χρειάστηκε για να παραχθεί. Δεν υπάρχει τόκος επί της μονάδας μέτρησης αξίας, δηλαδή εάν κάποιος οφείλει στο Σύστημα δεν πληρώνει τόκο, και εάν κάποιος έχει απόθεμα μονάδων, αυτό δεν αυξάνει με το πέρασμα του χρόνου.

3) Μπορεί κανείς να αρχίσει να συναλλάσσεται χωρίς να έχει κάποιο απόθεμα στο λογαριασμό του, σε ορισμένες όμως περιπτώσεις οι διαχειριστές του Συστήματος παρέχουν σε κάθε νέος μέλος και κάποιες μονάδες, ώστε να το ενθαρρύνουν στην διενέργεια των συναλλαγών. Οι συναλλαγές καταγράφονται πλέον ηλεκτρονικά, σε υπολογιστή του Συστήματος, όπου μπορούν τα μέλη με ένα τηλεφώνημα να αναφέρουν τη συναλλαγή. Κάποια μάλιστα Συστήματα στη Μ.Βρετανία έχουν λογισμικό που επιτρέπει στα μέλη να καταγράφουν απευθείας τη συναλλαγή τους μέσω διαδικτύου χρησιμοποιώντας κωδικούς. Παλαιότερα οι συναλλαγές καταγράφονταν χειρόγραφα ή/και με τη χρήση «επιταγών» που ανακοινώνονταν στους Διαχειριστές, αλλά εν τέλει αυτό το σύστημα αποδείχθηκε δύσχρηστο.

4) Τα προϊόντα που ανταλλάσσονται μέσω των Τοπικών Συστημάτων Ανταλλακτικού Εμπορίου ποικίλλουν, ανάλογα με την περιοχή όπου λειτουργεί το σύστημα και τα μέλη που συμμετέχουν. Βιολογικά προϊόντα από τις αυλές των σπιτιών, μαθήματα ξένων γλωσσών ή μουσικής, συνεργεία αυτοκινήτων ή επιδιορθώσεις ηλεκτρικών συσκευών, μεταποιήσεις ρούχων, ιατρικές ή οδοντιατρικές υπηρεσίες, οικιακές εργασίες, φροντίδα παιδιών αλλά και ερμηνεία αστρολογικού χάρτη ή μαθήματα γιόγκα, είναι κάποιες από τις προσφορές που μπορεί να συναντήσει κανείς σε καταλόγους προσφοράς των Συστημάτων αυτών.

5) Στις περισσότερες περιπτώσεις, τα άτομα που απασχολούνται με τη διαχείριση του Συστήματος αμείβονται επίσης με μονάδες του Συστήματος. Τα έξοδα διαχείρισης του Συστήματος που πληρώνονται σε άτομα που δεν συμμετέχουν στο σύστημα, για παράδειγμα, σε προμηθευτές αναλωσίμων ή ηλεκτρονικού υπολογιστή (εφ' όσον αυτός δεν μπορεί να ενοικιαστεί από άτομο του Συστήματος που δέχεται να πληρωθεί με «μονάδες») ή οι τηλεφωνικοί λογαριασμοί, καλύπτονται από τις ετήσιες εισφορές των μελών ή/και από ποσοστιαίες χρεώσεις επί της αξίας των συναλλαγών.

6) Εάν κανείς θέλει να μάθει περισσότερα για τα Τοπικά Συστήματα Ανταλλακτικού Εμπορίου, μπορεί να κοιτάξει τις ιστοσελίδες www.lets-linkup.com, www.letslinkuk.net, ιστοσελίδες τοπικών Συστημάτων, όπως αυτών του Μπραντφορντ (Αγγλία) www.bradfordlets.org , του Μπρίξτον (Αγγλία) www.brixlets.org.uk , του Νόριτς (Αγγλία) www.norlets.org.uk , του Ρέντιγκ (Αγγλία) www.readinglets.ik.com, του Πήτερμπορο (Καναδάς) http://ptbolets.50webs.com, ή και του Συστήματος του Λονδίνου www.letslinklondon.org.uk , που συνδέει τα Τοπικά Συστήματα που λειτουργούν στις διάφορες συνοικίες. Για τη Γαλλία μπορεί κανείς να δει την ιστοσελίδα www.sel-terre.info, για τη Γερμανία τις ιστοσελίδες www.tauschringe.de, www.regiotauschnetz.de, www.tauschringportal.de, www.tauschringadressen.de , για την Αυστραλία την ιστοσελίδα www.lets.org.au , για την Αυστρία www.tauschkreis.net.

Β) **Στην Ελβετία υπάρχει το Wirtschaftsring (Επιχειρηματικός κύκλος)** το οποίο είναι στην ουσία ένα πιστωτικό ίδρυμα μέσω του οποίου οι επιχειρήσεις και οι ελεύθεροι επαγγελματίες της χώρας προβαίνουν σε πολυμερές ανταλλακτικό εμπόριο. Το Wirtschaftsring χρησιμοποιεί ένα ιδεατό νόμισμα, το WIR, ίσης αξίας με το Ελβετικό φράγκο, και πλέον και τις νέες τεχνολογίες (ηλεκτρονικούς υπολογιστές, διαδίκτυο, χρεωστικές κάρτες) για την καταγραφή των συναλλαγών, αλλά και έχει πολύ αυστηρούς κανόνες για τα μέλη του (για παράδειγμα, παρέχουν οικονομικές εγγυήσεις προκειμένου να γίνουν δεκτά ως μέλη), ακριβώς επειδή οι συναλλαγές μπορεί να είναι μεγάλου ύψους και οι συναλλασσόμενοι επιδιώκουν εμπορικούς σκοπούς, και δραστηριοποιούνται σε όλη την επικράτεια και όχι μόνο στα πλαίσια μιας συνοικίας ή πόλης. Η εν λόγω αυστηρότητα, βέβαια επέτρεψε, σύμφωνα με κάποιες απόψεις, στο Wirtschaftsring να επιβιώσει από το 1934 μέχρι σήμερα. Οι συναλλαγές σε WIR γίνονται χωρίς τόκο, εκτός από δάνεια που δίνει το Wirtschaftsring στα μέλη του, τα οποία έχουν πολύ μικρότερο επιτόκιο από τα επιτόκια χορηγήσεων των υπολοίπων τραπεζών της χώρας. Η ιστοσελίδα του Wirtschaftsring είναι www.wir.ch σε τρεις γλώσσες (Γερμανικά, Γαλλικά και Ιταλικά).

Γ) Στη Βραζιλία, ανάμεσα στα άλλα, υπάρχει και **η Τράπεζα Πάλμας (Banco Palmas),** και μάλιστα είναι μια Τράπεζα που λειτουργεί στη συνοικία (φαβέλα) Παλμέιρας, στην πόλη Φορταλέζα. Η Τράπεζα Πάλμας παρέχει στους κατοίκους της συνοικίας ένα ιδεατό νόμισμα (Πάλμα ή Παλμάρες), το οποίο τίθεται σε κυκλοφορία με τη χορήγηση μικροδανείων σε αυτούς χωρίς τόκο. Με τη χρήση των Πάλμας, οι κάτοικοι της συνοικίας μπορούν να συναλλάσσονται μεταξύ τους ανεξάρτητα εάν έχουν ή όχι επίσημο νόμισμα (ρεάλ) στις τσέπες τους ή τους λογαριασμούς τους) και κυρίως μπορούν να παράγουν προϊόντα και υπηρεσίες μέσω μικρών επιχειρήσεων που μπορούν να δημιουργήσουν με τα χορηγούμενα δάνεια. Οι συναλλαγές γίνονται κυρίως μέσω μιας ηλεκτρονικής κάρτας, η οποία καταγράφει τις συναλλαγές όπως περίπου οι χρεωστικές κάρτες των Τραπεζών. Το ιδεατό νόμισμα χρησιμοποιείται από τους κατοίκους της συνοικίας για τις συναλλαγές τους στην περιοχή τους αλλά και για τη χρηματοδότηση δημόσιων και ιδιωτικών έργων στην περιοχή τους, όπως η υποστήριξη της αστικής γεωργίας στην περιοχή και η εξασφάλιση τροφίμων στη συνοικία, η δημιουργία καταστημάτων όπου οι κάτοικοι μπορούν να πωλούν για

Πάλμας προϊόντα που παράγουν οι ίδιοι στα σπίτια τους, η δημιουργία μικρών βιοτεχνικών μονάδων για παραγωγή ενδυμάτων, καθαριστικών προϊόντων ή ειδών σπιτιού, η εκπαίδευση των κατοίκων, κλπ. Η ιστοσελίδα της Τράπεζας Πάλμας είναι http://www.bancopalmas.org/

Δ) **Η έκδοση και χρήση ενός άλλου, τοπικού, νομίσματος, παράλληλα με το επίσημο νόμισμα**, είναι μια άλλη λύση που την προτιμούν κυρίως σε διάφορες περιοχές στις Ηνωμένες Πολιτείες (αλλά και σε άλλα μέρη του κόσμου).

1) Το τοπικό νόμισμα εκδίδεται μόνο σε χαρτονομίσματα, τα οποία υποχρεωτικά πρέπει να διαφέρουν πολύ από το επίσημο νόμισμα., και έχουν υποδιαιρέσεις. Κυκλοφορεί τοπικά, πράγμα που σημαίνει ότι ο επισκέπτης στην πόλη ή περιοχή πρέπει να αγοράσει τοπικό νόμισμα πληρώνοντας με επίσημο νόμισμα.

2) Δεν υποχρεούται κανείς στην περιοχή να δεχθεί για πληρωμή το τοπικό νόμισμα. Όσοι το δέχονται, θα το χρησιμοποιήσουν, γιατί έτσι κι αλλιώς δεν αποταμιεύεται με τόκο, ούτε μπορεί κανείς να το δανείσει με επιτόκιο. Όσες επιχειρήσεις ή επαγγελματίες θέλουν να συμμετέχουν στο δίκτυο του τοπικού νομίσματος, αφ' ενός αναλαμβάνουν την υποχρέωση να το στηρίξουν με τα προϊόντα ή τις υπηρεσίες που παράγουν και αφ' ετέρου συμμετέχουν στη λήψη αποφάσεων και στη διαχείριση του νομίσματος. Υπάρχουν μάλιστα και ορισμένες περιπτώσεις, όπου τοπικές τράπεζες δέχονται συναλλαγές σε τοπικό νόμισμα (και το αντιμετωπίζουν όπως περίπου το ξένο συνάλλαγμα).

3) Την έκδοση και διαχείριση του νομίσματος και της κυκλοφορίας του έχει η Επιτροπή, η οποία εκλέγεται από όσους συμμετέχουν στην έκδοση του νομίσματος. Το νόμισμα τίθεται σε κυκλοφορία με τη συμμετοχή κάθε νέου μέλους στο δίκτυο, δηλαδή κάθε νέο μέλος λαμβάνει μικρό ποσό νέων νομισμάτων για να αρχίσει τις συναλλαγές του.

4) Για να γίνει πιο ελκυστικό το τοπικό νόμισμα για επιχειρήσεις και επαγγελματίες, πέρα από τη διαφήμιση που λαμβάνουν μέσω του καταλόγου προϊόντων και υπηρεσιών που μπορούν να αποπληρωθούν με τοπικό νόμισμα και όχι με επίσημο, τους επιτρέπεται να καθορίζουν την πληρωμή σε τοπικό νόμισμα μόνο μέχρι ένα ποσοστό της τιμής του προϊόντος/υπηρεσίας και το υπόλοιπο τίμημα να πληρώνεται κανονικά σε επίσημο νόμισμα. Το ποσοστό αυτό ανακοινώνεται επίσης στον κατάλογο προϊόντων και υπηρεσιών.

5) Οι νέες τεχνολογίες επιτρέπουν αφ' ενός να ανακοινώνονται οι κατάλογοι προϊόντων και υπηρεσιών (και να ανανεώνονται σε κάποιες περιπτώσεις πολύ γρήγορα, από τους ίδιους τους ενδιαφερόμενους) και αφ' ετέρου να τυπώνονται χαρτονομίσματα, τα οποία είναι γενικά ασφαλή από παραχάραξη (που έτσι κι αλλιώς δεν έχει ιδιαίτερο κίνητρο, γιατί το νόμισμα αυτό χρησιμοποιείται ουσιαστικά μόνο για συναλλαγές σε τοπικά προϊόντα και υπηρεσίες).

6) Όπως και στα Τοπικά Συστήματα Ανταλλακτικού Εμπορίου, τα προϊόντα και οι υπηρεσίες ποικίλλουν. Για την ενίσχυση του τοπικού νομίσματος

διοργανώνονται και υπαίθριες ή ημιϋπαίθριες αγορές, προκειμένου να γίνονται οι συναλλαγές στο τοπικό νόμισμα αλλά και να μπορούν και όσοι ιδιώτες δεν ενδιαφέρονται να συμμετέχουν ως μέλη, και απλώς έχουν στην τσέπη τους κάποια νομίσματα, να αγοράσουν προϊόντα.

7) Ενδεικτικά μπορεί κανείς να αναφέρει το νόμισμα Ώρες της Ίθακα (ΗΠΑ) www.ithacahours.org , του Κάλγκαρυ (Καναδάς) www.calgarydollars.ca, , του Κορβάλλις (ΗΠΑ) www.hourexchange.org , του Μάντισον (ΗΠΑ) www.madisonhours.org , τα Ισο-δολλάρια της Φιλαδέλφεια (ΗΠΑ) www.rhd.org/equal.html , τα Νομίσματα Συναλλαγών του Χούμπολτ (ΗΠΑ) www.humboltexchange.org , τα δολλάρια του Τορόντο (Καναδάς) www.torontodollar.com, το Πλέντυ (Άφθονο) στη Β.Καρολίνα (ΗΠΑ) www.ncplenty.org το οποίο είναι περιφερειακό νόμισμα και δεν περιορίζεται μόνο σε μια πόλη, , τα νομίσματα Chiemgauer www.chiemgauer.info και Regio www.regiogeld.de στη Γερμανία, τα οποία επίσης είναι περιφερειακά νομίσματα, δηλαδή αφορούν σε ολόκληρες επαρχίες και όχι μόνο σε κάποια πόλη.

Ε) **Στο Ντακάρ της Σενεγάλης κυκλοφορεί το τοπικό νόμισμα Μπον (Bon, Bon d' Échange)**, σε πέντε τοπικά συστήματα συναλλαγών αντίστοιχης δομής με τα ευρωπαϊκά Τοπικά Συστήματα Ανταλλακτικού Εμπορίου, με τη διαφορά ότι η ιδεατή μονάδα μέτρησης αξίας έχει λάβει τη μορφή χαρτονομίσματος. Τα πέντε αυτά τοπικά συστήματα υπάγονται στο δίκτυο Ντουλ (Doole, που σημαίνει Ισχύς εν τη Ενώσει στην τοπική γλώσσα), που είναι δίκτυο συναλλαγών που καλύπτει όλη την πόλη του Ντακάρ.

1) Το Μπον αντιστοιχεί σε μία ώρα εργασίας, δηλαδή είναι η αμοιβή για μια ώρα εργασίας και εκδίδεται σε τρεις υποδιαιρέσεις. Η ιδιαιτερότητα του Μπον είναι ότι έχει αρνητικό επιτόκιο, δηλαδή με το πέρασμα του χρόνου η αξία του μειώνεται και για να επανέλθει, πρέπει ο κάτοχος του χαρτονομίσματος να αγοράσει από το δίκτυο ειδικά χαρτόσημα και να τα επικολλήσει στην πίσω πλευρά του χαρτονομίσματος. Με αυτόν τον τρόπο ενισχύεται η κυκλοφορία του νομίσματος μεταξύ των κατοίκων του Ντακάρ (γιατί κάθε ένας που κερδίζει ένα ποσό σε Μπον θα φροντίσει να τα ξοδέψει όσο το δυνατόν γρηγορότερα ώστε να αποφύγει να πληρώσει τα χαρτόσημα) και καλύπτονται τα έξοδα διαχείρισης του συστήματος. Ένα Μπον αντιστοιχεί σε ποσό 1000 επίσημων νομισμάτων, αλλά δεν επιτρέπεται από το δίκτυο η μετατροπή των Μπον σε επίσημο νόμισμα.

2) Τη διαχείριση του νομίσματος την έχει εκλεγόμενο Διοικητικό Συμβούλιο, αλλά υπάρχουν και εξειδικευμένες επιτροπές, που ασχολούνται με ειδικότερα θέματα που αφορούν στη χρήση και κυκλοφορία του νομίσματος. Η χρήση νέων τεχνολογιών επιτρέπει να έχουν τα χαρτονομίσματα αριθμό σειράς, αλλά και να γίνεται η διαχείριση μέσω ηλεκτρονικών υπολογιστών.

3) Μέλος μπορεί να γίνει όποιος μπορεί να πληρώσει μισό Μπον σε επίσημο νόμισμα και ως νέο μέλος λαμβάνει 5 (παλαιότερα 3) Μπον για να αρχίσει τις συναλλαγές. Στις αγορές, όμως, και στα καταστήματα μπορεί κανείς να συναλλαγεί με Μπον και χωρίς να είναι μέλος του δικτύου.

4) Τα προϊόντα που αγοράζει κανείς με το Μπον είναι κυρίως βασικά είδη, δηλαδή τρόφιμα (κυρίως αγροτικά προϊόντα) και είδη ρουχισμού και μπορεί να τα βρει είτε στις υπαίθριες αγορές που οργανώνονται τακτικά, είτε στο κατάστημα του δικτύου Ντουλ.

5) Ένα αγαθό, που επίσης αγοράζει κανείς με Μπον, και το οποίο ήταν αρχικά η βασική αφορμή για τη δημιουργία του δικτύου, είναι η εκπαίδευση. Με το Μπον χρηματοδοτείται η εκπαίδευση στο Λαϊκό Πανεπιστήμιο του Ντακάρ, όπου έμφαση έχει δοθεί σε 5 τομείς: ξένες γλώσσες, χρήση ηλεκτρονικών υπολογιστών, γνώσεις εμπορικών συναλλαγών, γνώσεις για παραγωγή σε τοπικό επίπεδο διαφόρων προϊόντων και ιδίων δημητριακών. Η χρηματοδότηση δεν περιορίζεται μόνο στο Λαϊκό Πανεπιστήμιο, αλλά και σε άλλα δημόσια έργα που χρειάζονται οι κάτοικοι του Ντακάρ. Δεν υπάρχει ιστοσελίδα του δικτύου Ντουλ, αλλά πληροφορίες μπορεί κανείς να βρει στις ιστοσελίδες http://appropriate-economics.org/Africa/Africa.html και www.enda.sn.

ΣΤ) **Στην Αργεντινή** υπήρχε το μεγαλύτερο δίκτυο (το 2003 είχε 6 εκατομμύρια μέλη) ανταλλαγής προϊόντων και υπηρεσιών και μολονότι κατέρρευσε το 2003 (δηλαδή, κυκλοφόρησαν πλαστά χαρτονομίσματα και πολλοί αποσύρθηκαν από μέλη ή τουλάχιστον σταμάτησαν να συμμετέχουν στις υπαίθριες αγορές που διοργάνωνε το δίκτυο), εξακολουθεί να παραμένει αν όχι το μεγαλύτερο, ένα από τα μεγαλύτερα και ίσως μακροβιότερα δίκτυα στον κόσμο (ξεκίνησε το 1995). Ονομάζεται **Γενικό Δίκτυο Ανταλλαγής (Red Global de Trueque)** και ουσιαστικά είναι δίκτυο που συνδέει άλλα μικρότερα δίκτυα ανταλλαγής που υπάρχουν διάσπαρτα στο Μπουένος Άιρες αλλά και σε άλλες πόλεις και περιοχές της Αργεντινής.

1) Μολονότι οι Λέσχες Ανταλλαγής (Clubes de Trueque) έχουν όνομα που παραπέμπει στην ανταλλαγή, χρησιμοποιούνται χαρτονομίσματα ή ιδεατά νομίσματα για τη διενέργεια συναλλαγών. Το νόμισμα λέγεται «μονάδα» (crédito) ή «δέντρο» ή «δεντράκι» (árbol, arbolito) και ονομάζεται έτσι από το δέντρο που υπάρχει πάνω στα χαρτονομίσματα ως εικόνα. Τα χαρτονομίσματα τα εκδίδει κάθε τοπικό δίκτυο (Λέσχη) χωριστά, αλλά γενικά γίνονται δεκτά από το ένα δίκτυο στο άλλο, οπότε τα μέλη μιας Λέσχης μπορούν να συναλλάσσονται στις αγορές και των άλλων Λεσχών, χωρίς να υπάρχει κάποιο πρόβλημα μετατροπής, αφού κάθε «δεντράκι» έχει αξία ενός δολλαρίου ΗΠΑ.

2) Δεν υπάρχει επιτόκιο, ούτε φορολογούνται οι συναλλαγές σε «δεντράκια», για αυτό και δεν επιτρέπεται στις αγορές που οργανώνουν τα δίκτυα να κυκλοφορήσει οποιοδήποτε άλλο νόμισμα, είτε το επίσημο (πέσο) είτε δολλάριο ΗΠΑ, προκειμένου να μην παρέμβουν οι φορολογικές αρχές. Επιτρέπεται όμως να αγοράζει κανείς προϊόντα ή πρώτες ύλες από άλλες αγορές σε άλλο νόμισμα και να τα πουλά αυτούσια ή μεταποιημένα στις αγορές του δικτύου μόνο για «δεντράκια». Για να γίνει κανείς μέλος σε μια Λέσχη πληρώνει ένα πολύ μικρό τέλος συμμετοχής, όμως λόγω της μεγάλης συμμετοχής, τα έσοδα των Λεσχών είναι επαρκή.

3) Με «δεντράκια» μπορεί κανείς να αγοράσει σχεδόν τα πάντα: αγροτικά προϊόντα, όλων των ειδών τα τρόφιμα (και μάλιστα σπιτικά), ρούχα, καθαριστικά, αναλώσιμα για το σχολείο, και διάφορες υπηρεσίες, όπως καθαριότητα, φροντίδα παιδιών, ιατρικές ή νομικές υπηρεσίες, μαθήματα ξένων γλωσσών, κλπ.

4) Οι τοπικές αρχές στήριξαν από την αρχή τα τοπικά δίκτυα συναλλαγών και όταν μάλιστα από το 2000 και μετά η οικονομική κατάσταση στην Αργεντινή και ιδίως στα μεγάλα αστικά κέντρα επιδεινώθηκε, το Υπουργείο Οικονομίας της Αργεντινής συμφώνησε να στηρίξει το Γενικό Δίκτυο στα θέματα διαδικτυακής τεχνολογίας.

5) Περισσότερες πληροφορίες μπορεί να βρει κανείς στην ιστοσελίδα του Γενικού Δικτύου www.trueque.org.ar

Ζ) **Στο Νότιο Μπερκσάιρ (Southern Berkshire) της Μασσαχουσέττης των ΗΠΑ, κυκλοφορεί το τοπικό νόμισμα Μπερκσέαρ (Berkshare)**, το οποίο υιοθετήθηκε στην περιοχή προκειμένου να ενισχυθεί η τοπική οικονομία.

1) Τα Berkshares τίθενται σε κυκλοφορία από τους κατοίκους της περιοχής ή όποιον είναι διατεθειμένος να αγοράσει 100 Berkshares για 90 δολλάρια ΗΠΑ. Όταν όμως τα Berkshares κυκλοφορούν στην αγορά η αξία τους είναι ίση με αυτή του δολλαρίου, επομένως, όποιος αγοράζει Berkshares κερδίζει 10% της αξίας τους σε δολλάρια. Τα Berkshares τα αγοράζει κανείς από όποιο τραπεζικό κατάστημα βρίσκεται στην περιοχή του Μπερκσάιρ και κυκλοφορούν σε 3 χαρτονομίσματα διαφορετικής αξίας.

2) Οι επιχειρήσεις της περιοχής δέχονται τα Berkshares γιατί με αυτόν τον τρόπο κερδίζουν πελάτες. Επιπλέον, επειδή για να τα επιστρέψουν/καταθέσουν στην Τράπεζα, θα τους κοστίσει περίπου 10% της αξίας τους (δηλαδή, εάν πουλήσουν στην Τράπεζα 100 Berkshares, θα πάρουν στα χέρια τους 90 δολλάρια ΗΠΑ) είναι προτιμότερο να φροντίσουν να αγοράσουν κάτι με αυτά από τη διπλανή επιχείρηση ή υπηρεσίες από κάποιον επιχειρηματία/επαγγελματία της περιοχής που δέχεται Berkshares.

3) Δεν είναι υποχρεωτικό για κανέναν και καμία επιχείρηση να δέχεται Berkshares, ούτε να συμμετέχει στο δίκτυο Berkshares (που έχει μορφή εταιρείας), αλλά εάν τα δέχεται, έχει πλεονέκτημα στις συναλλαγές και εάν συμμετέχει στο δίκτυο, έχει ουσιαστικά δωρεάν διαφήμιση και κατά κάποιον τρόπο εξασφαλισμένη πελατεία.

4) Περισσότερες πληροφορίες για το νόμισμα Berkshares μπορεί να βρει κανείς στην σχετική ιστοσελίδα www.berkshares.org .

Η) Μια άλλη μορφή δικτύων συναλλαγών είναι οι **Χρονοτράπεζες (Timebanks).** Η Χρονοτράπεζα είναι ένα δίκτυο για την ανταλλαγή κατά βάση υπηρεσιών. Αυτό συμβαίνει γιατί βασική αρχή της Χρονοτράπεζας είναι η ισότητα του χρόνου όλων

των ανθρώπων, επομένως είναι υποχρεωτικό να χρησιμοποιείται ως μονάδα μέτρησης αξίας η ώρα εργασίας, δηλαδή κάθε υπηρεσία ή κάθε προϊόν αξίζει όσες ώρες χρειάστηκε ο παραγωγός για να το παραγάγει. Κατά συνέπεια, δεν είναι εύκολο να ενσωματωθούν στην «τιμή» πρώτες ύλες που αγοράστηκαν σε επίσημο νόμισμα.

1) Σε κάθε περίπτωση, η χρήση της ώρας εργασίας ως μονάδας μέτρησης αξίας επιτρέπει να αμείβονται το ίδιο εργασίες, οι οποίες στην συμβατική αγορά αμείβονται με πολύ μεγάλες αποκλίσεις τιμής. Δηλαδή, οι οικιακές εργασίες αμείβονται το ίδιο με τις υπηρεσίες ενός γιατρού ή ενός προγραμματιστή ηλεκτρονικών υπολογιστών.

2) Φυσικά δεν υπάρχουν επιτόκια, αλλά στους ηλεκτρονικούς λογαριασμούς κάθε μέλους χρεώνονται οι ώρες που εργάστηκε και οι ώρες που έλαβε υπηρεσίες ή αγαθά. Επίσης, δεν υπάρχει δυνατότητα σύνδεσης των «ωρών» με οποιοδήποτε νόμισμα, επίσημο ή όχι. Είναι ο μόνος τύπος δικτύου συναλλαγής που θεωρείται επισήμως ότι εξαιρείται από φορολογικό έλεγχο, ούτε χρειάζεται κανείς να δηλώνει «εισοδήματα» ή «έξοδα» από την Χρονοτράπεζα. Αυτό ισχύει, όχι μόνο στις ΗΠΑ, από όπου ξεκίνησε, και στη Μ.Βρετανία, αλλά και στην Ελλάδα, δηλαδή οι φορολογικές αρχές θεωρούν ότι οι υπηρεσίες παρέχονται εθελοντικά και δεν έχουν μορφή κανονικής συναλλαγής.

3) Για τις Χρονοτράπεζες στις ΗΠΑ μπορεί κανείς να επισκεφθεί την ιστοσελίδα www.timebanks.org, στη Μ.Βρετανία www.timebanks.co.uk, www.timebanking.uk αλλά και www.wicc.org.uk .

4) Στην Ελλάδα λειτουργεί η **Χρονοτράπεζα του Ελληνικού Δικτύου Γυναικών Ευρώπης** (www.enow.gr)**.** Συμμετέχει όποιος θέλει, συμπληρώνοντας μια απλή αίτηση και έχει καθιερωθεί προθεσμία λήξης των πιστωμένων μονάδων, ώστε να έχει κίνητρο ο δικαιούχος να τις ξοδέψει αντί να τις αποταμιεύει.

5) Στην Ιαπωνία υπάρχει το δίκτυο **Φουρεάι Κίππου (Fureai Kippu)**, μέσω του οποίου παρέχονται υπηρεσίες φροντίδας σε ηλικιωμένους και σε άτομα με ειδικές ανάγκες. Το δίκτυο είναι ουσιαστικά σύνδεσμος τοπικών δικτύων (περίπου 400 τον αριθμό) που υπάρχουν σε όλη την Ιαπωνία και επιτρέπει σε άτομα που δεν έχουν συγγενείς στο άμεσο περιβάλλον τους για να τα φροντίσουν, να έχουν υψηλό επίπεδο διαβίωσης και να αποφύγουν τον εγκλεισμό σε γηροκομεία και κλινικές ή άλλα ιδρύματα. Την ίδια στιγμή, επιτρέπει σε ανθρώπους που είναι μακριά από τους ηλικιωμένους συγγενείς τους να κερδίσουν μονάδες φροντίζοντας κάποιον γείτονα και να μεταφέρουν τις μονάδες αυτές στους συγγενείς τους που κατοικούν μακριά, ώστε να μπορούν να τους φροντίσουν άλλα άτομα εκεί που μένουν. Έχει γίνει μάλιστα ειδική ρύθμιση, ώστε όσοι νέοι συμμετέχουν στο Φουρεάι Κίππου, να μπορούν με τις μονάδες που κερδίζουν, να πληρώνουν τα κατά τα άλλα πανάκριβα πανεπιστημιακά δίδακτρα για τις σπουδές τους.

Όλα τα παραπάνω είναι μερικές μόνο προσπάθειες από όσες λαμβάνουν χώρα αυτήν την εποχή στον κόσμο. Εάν κανείς θέλει να μάθει περισσότερα, μπορεί να ανατρέξει στη βάση δεδομένων παράλληλων νομισμάτων www.complementarycurrency.org αλλά θα πρέπει να έχει υπ' όψη του ότι πολλά δίκτυα λειτουργούν χωρίς ιστοσελίδες, είτε λόγω έλλειψης τεχνολογίας, είτε λόγω του ότι οι χρήστες δεν χρειάζονται το διαδίκτυο για να προβούν στις συναλλαγές που θέλουν.

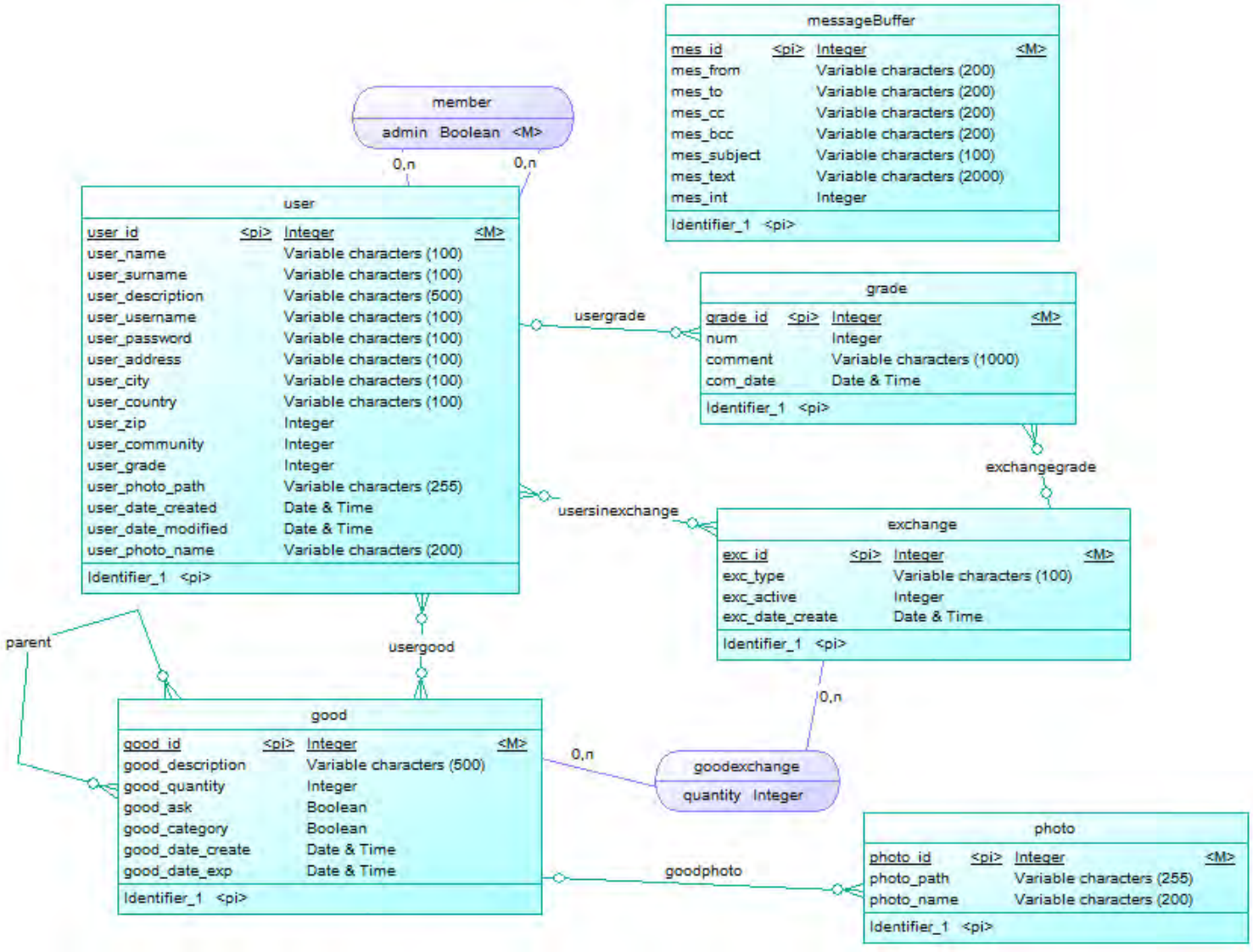
Για το θεωρητικό υπόβαθρο των παραπάνω προσπαθειών, αλλά και πολλά άλλα ενημερωτικά και ιστορικά στοιχεία, μπορεί κανείς να ανατρέξει στη βιβλιοθήκη της ιστοσελίδας www.appropriate-economics.org , στο επιστημονικό περιοδικό www.uea.ac.uk/env/ijccr , στην ιστοσελίδα www.feasta.org, όπου μπορεί να βρει ανάμεσα στα άλλα το βιβλίο Short Circuit του Richard Douthwaite, στην ιστοσελίδα της Μη Κυβερνητικής Οργάνωσης Στρόχαλμ www.strohalm.nl, στην ιστοσελίδα του Σωματείο E.F.Schumacher www.schumachersociety.org , στην ιστοσελίδα του Ινστιτούτου της Ουαλίας για τα Κοινοτικά Νομίσματα www.wicc.org.uk και στην ιστοσελίδα του Ινστιτούτου Νέας Οικονομίας www.neweconomics.org . Θα παρατηρήσει κανείς την έντονα περιβαλλοντική κατεύθυνση των τοπικών ή παράλληλων νομισμάτων ή δικτύων ανταλλαγής, και την σχέση τους με την αειφόρο οικονομία και παραγωγή.

Θα παρατηρήσει επίσης, ότι οι προσπάθειες είναι τόσες όσες και οι τοπικές κοινωνίες, δηλαδή ότι κάθε χώρα, περιοχή, κάθε κοινότητα ή κάθε ομάδα ανθρώπων προσαρμόζει τις λύσεις και τις πληροφορίες που υπάρχουν στις συγκεκριμένες ανάγκες και ότι δεν υπάρχουν έτοιμες λύσεις. Αλλά ζήτησε κανείς έτοιμες λύσεις;

3 Βάση Δεδομένων

3.1 Εννοιολογικό Μοντέλο

Το εννοιολογικό μοντέλο της βάσης δεδομένων του συστήματος φαίνεται παρακάτω:



Οι πίνακες

Το σχήμα αποτελείται από 6 πίνακες:

- Τον πίνακα User στον οποίο καταχωρείται όποιο νέο μέλος επισκέπτεται για πρώτη φορά το site και εγγράφεται σε αυτό,όπως επίσης και όλες οι κοινωνίες οι οποίες δημιουργούνται από τους χρήστες. Ο πίνακας αυτό αποτελείται από 16 πεδία.

  Το πεδίο user_id είναι το πρωτεύον κλειδί του πίνακα. Είναι τύπου int και ανατίθεται αυτόματα σε κάθε εγγραφή που εισάγεται στον πίνακα.
  Στο πεδίο user_name, τύπου string, αποθηκεύεται το όνομα που θα δώσει ο χρήστης. Μπορεί να είναι είτε το πραγματικό του όνομα είτε ένα ψευδώνυμο που θα επιλέξει ο χρήστης. Το όνομα αυτό θα αντιπροσωπεύει το χρήστη στην εφαρμογή(εμφανιζόμενο όνομα)

  Το πεδίο user_surname,τύπου string, έχει προστεθεί για την αποθήκευση του επωνύμου του χρήστη η συμπλήρωση του οποίου είναι προαιρετική.

  Στα πεδία user_username και user_password αποθηκεύονται οι προσωπικοί κωδικοί με τους οποίους θα κάνει login το νέο μέλος κάθε φορά που θα επισκέπτεται το site. Και τα δύο πεδία είναι τύπου string. Στην περίπτωση που ο χρήστης είναι μία κοινωνία το στοιχεία user_username και user_password ορίζονται από το χρήστη που δημιουργεί την κοινωνία αυτή και τα χρησιμοποιεί στην περίπτωση που θέλει να κάνει αλλαγές που αφορούν την κοινωνία.

  Στα πεδία user_address, user_city, user_zip και user_country αποθηκεύεται η ακριβής διεύθυνση του μέλους. Ο χρήστης δεν είναι αναγκασμένος να δώσει όλα αυτά τα στοιχεία αν δεν το επιθυμεί. Τα στοιχεία αυτά μπορούν να φανούν χρήσιμα στην περίπτωση που δύο χρήστες αποφασίσουν μία ανταλλαγή αγαθών να μπορέσουν να αποστείλουν για παράδειγμα ταχυδρομικώς τα αντικείμενα ο ένας στον άλλο. Τα πεδία user_address, user_city και user_country είναι τύπου string, περιλαμβάνουν αντίστοιχα την διεύθυνση, την πόλη και την χώρα διαμονής του χρήστη ενώ το πεδίο per_zip είναι τύπου int και περιλαμβάνει τον ταχυδρομικό κωδικό της πόλης στην οποία διαμένει ο χρήστης.

  Το πεδίο user_grade είναι τύπου integer και περιέχει τη βαθμολογία που έχει συγκεντρώσει ο χρήστης μετά τις ανταλλαγές τις οποίες έχει πραγματοποιήσει με άλλους χρήστες του συστήματος.

  Το πεδίο user_photo_path περιέχει τη διαδρομή στο σύστημα αρχείων του server στο οποίο αποθηκεύονται οι φωτογραφίες που έχει εισάγει ο συγκεκριμένος χρήστης. Ενώ το πεδίο user_photo_name περιέχει το όνομα της φωτογραφίας η οποία εμφανίζεται στο προφίλ του χρήστη.

Το πεδίο user_date_created περιέχει την χρονική στιγμή που ο χρήστης δημιούργησε το λογαριασμό του στο σύστημα και είναι τύπου date & time. Στο πεδίο user_date_modified αποθηκεύεται η χρονική στιγμή που χρήστης άλλαξε για τελευταία φορά τα στοιχεία του λογαριασμού του.

Τέλος το πεδίο user_community είναι τύπου Boolean αποθηκεύεται σε αυτό η τιμή true αν η συγκεκριμένη εγγραφή του πίνακα αφορά μια κοινωνία ενώ false αποθηκεύεται στην περίπτωση που εγγραφή αφορά έναν απλό χρήστη. Στην περίπτωση που η εγγραφή στον πίνακα user αφορά μία κοινωνία τα πεδία user_name, user_surname, user_address, user_city, user_zip και user_country είναι κενά

- Τον πίνακα Good, στον οποίο καταχωρούνται οι προσφορές που κάνει οποιοσδήποτε χρήστης, τα αγαθά τα οποία έχει δηλώσει ότι επιθυμεί να αποκτήσει αλλά και οι κατηγορίες στις οποίες ταξινομούνται τα αγαθά . Ο πίνακας αποτελείται από τα παρακάτω πεδία:

Το πεδίο good_id, τύπου int, που είναι το πρωτεύον κλειδί του πίνακα και ανατίθεται αυτόματα σε κάθε εγγραφή.

Το πεδίο good_description, τύπου String, περιέχει την περιγραφή που έχει δώσει ο χρήστης για το στοιχείο το οποίο εισάγει στο σύστημα.

Το πεδίο good_quantity, τύπου int, όπου αποθηκεύεται η ποσότητα του αγαθού την οποία ο χρήστης προσφέρει.

Στο πεδίο good_date_create,τύπου date & time, περιέχεται η χρονική στιγμή κατά την οποία ο χρήστης εισήγαγε την προσφορά του, ενώ το πεδίο good_date_exp περιέχει τη χρονική στιγμή μέχρι την οποία το συγκεκριμένο αγαθό θα είναι διαθέσιμο στους υπόλοιπους χρήστες για να υποβάλουν προσφορά ώστε να το ανταλλάξουν με κάποιο αντικείμενο που αυτοί διαθέτουν.

Τα πεδία good_ask και good_category είναι τύπου Boolean. Το πεδίο good_ask έχει την τιμή true στην περίπτωση που η εγγραφή του πίνακα αναφέρεται σε μία επιθυμία κάποιου χρήστη και όχι σε κάποια προσφορά. Ενώ το πεδίο good_category είναι true αν η εγγραφή αποτελεί μία κατηγορία αγαθών και false εάν είναι ένα αγαθό.

- Τον πίνακα photo. Στον πίνακα αυτό αποθηκεύονται τα στοιχεία των φωτογραφιών τις οποίες εισάγει ο χρήστης για κάθε αγαθό που καταχωρεί.

Το πεδίο photo_id είναι το πρωτεύον κλειδί του πίνακα και δίνεται αυτόματα σε κάθε εγγραφή. Είναι τύπου Integer.

Τα πεδία photo_path και photo_name περιέχουν τη διαδρομή στο σύστημα αρχείων του server όπου αποθηκεύονται οι φωτογραφίες και το όνομα της κάθε φωτογραφίας αντίστοιχα

- Τον πίνακα exchange. Στον πίνακα αυτό καταχωρούνται όλες οι συναλλαγές που είναι σε εξέλιξη. Ο πίνακας αποτελείται από 4 πεδία:

Το πεδίο exc_id το οποίο είναι το πρωτεύον κλειδί του πίνακα και ανατίθεται αυτόματα στην κάθε εγγραφή.

Το πεδίο exc_type,τύπου String, το οποίο περιέχει τον τύπο της κάθε συναλλαγής. Κάθε συναλλαγή μπορεί να είναι είτε τύπου συναλλαγής, είτε τύπου δωρεάς, είτε κοινοχρησίας.

Το πεδίο exc_active, τύπου Boolean. Είναι true στην περίπτωση που συναλλαγή δεν έχει ακόμη ολοκληρωθεί και false στην περίπτωση που η συναλλαγή ολοκληρώθηκε(είτε έγινε αποδεκτή είτε απορρίφθηκε).

Στο πεδίο exc_date_create,τύπου date & time, αποθηκεύεται η χρονική στιγμή κατά την οποία ο χρήστης υπέβαλε την πρόταση συναλλαγής.

- Τον πίνακα grade. Μόλις μία συναλλαγή ολοκληρωθεί επιτυχώς οι δύο χρήστες που συμμετείχαν σε αυτήν βαθμολογούν ο ένας τον άλλο. Ο πίνακας αυτός αποτελείται από 4 πεδία:

Το πεδίο grade_id, το πρωτεύον κλειδί του πίνακα.

Το πεδίο num το οποίο περιέχει τη βαθμολογία που πήρε ο χρήστης συμμετέχοντας σε μία συναλλαγή.

Το πεδίο comment, στο οποίο αποθηκεύεται το σχόλιο που έχει δεχθεί ο χρήστης μαζί με το βαθμό μετά από τη συμμετοχή του σε μία συναλλαγή από το δεύτερο μέλος της συναλλαγής.

Και τέλος το πεδίο com_date το οποίο περιέχει τη χρονική στιγμή κατά την οποία καταχωρήθηκε το σχόλιο.

- Τον πίνακα messageBuffer. Στον πίνακα αυτό αποθηκεύονται τα μηνύματα που θέλουν να στείλουν οι χρήστες μεταξύ τους. Τα μηνύματα αυτά στέλνονται με τη μορφή e-mail και μόλις σταλούν διαγράφονται. Ο πίνακας αποτελείται από 7 πεδία.

  Το πεδίο mes_id το οποίο είναι το πρωτεύον κλειδί του πίνακα

  Τα πεδία mes_from, mes_to, mes_cc, mes_bcc, mes_subject και mes_text τα οποία περιέχουν όλα τα στοιχεία για την αποστολή το μηνύματος και το μήνυμα που πρέπει να αποσταλεί.

Σχέσεις μεταξύ πινάκων

- member

  Η σχέση member είναι μία συσχέτιση του πίνακα User με τον εαυτό του. Η σχέση αυτή έχει λόγο πληθικότητας Many-To-Many.

  Η σχέση αυτή ορίζει τα μέλη της κάθε κοινωνίας. Από τον λόγο πληθικότητας μπορούμε να συμπεράνουμε ότι κάθε άτομο μπορεί να είναι μέλος σε 0 έως N κοινωνίες και κάθε κοινωνία έχει από 0 έως N μέλη. Η συσχέτιση member περιέχει το γνώρισμα admin τύπου Boolean. Το γνώρισμα αυτό είναι true εάν το μέλος της κοινωνίας που ορίζει η σχέση είναι και administrator αυτής.

- parent

  Η σχέση member συνδέει τον πίνακα good με τον εαυτό του και έχει λόγο πληθικότητας Many-To-Many.

  Η σχέση αυτή συνδέει τα προσφερόμενα αγαθά με την κατηγορία στην οποία ανήκουν, αλλά και την κάθε υποκατηγορία με την κατηγορία στην οποία ανήκει.

  Η σχέση έχει λόγο πληθικότητας Many-To-Many γιατί κάθε κατηγορία μπορεί να έχει από 0 έως N αγαθά αλλά και κάθε αντικείμενο μπορεί να ανήκει σε πολλές κατηγορίες.

- goodexchange

  Η σχέση goodexchange συνδέει τους πίνακες good και exchange. Ο λόγος πληθικότητας είναι Many-To-Many.

  Κάθε συναλλαγή μπορεί να περιλαμβάνει 1 έως N αντικείμενα ενώ κάθε αντικείμενο μπορεί να συμμετέχει σε 0 έως N συναλλαγές.

  Η συσχέτιση goodexchange περιέχει ένα το γνώρισμα quantity το οποίο αποθηκεύει την προσφερόμενη ή ζητούμενη ποσότητα του κάθε αγαθού που συμμετέχει στη συναλλαγή.

- goodphoto

  Η σχέση goodphoto συνδέει τους πίνακες good και photo με λόγο πληθικότητας One-To-Many.

  Κάθε αντικείμενο μπορεί να έχει από 0 εώς Ν φωτογραφίες ενώ κάθε φωτογραφία πρέπει να ανήκει το πολύ σε ένα αντικείμενο.

- usergood

  Η σχέση usergood συνδέει τους πίνακες user και good. Ο λόγος πληθικότητας είναι Many-To-Many.

  Κάθε χρήστης της υπηρεσίας, είτε άτομο είτε κοινωνία, μπορεί να προσφέρει από 0 εώς Ν αντικείμενα ή υπηρεσίες, ενώ κάθε αντικείμενο ή υπηρεσία που προσφέρεται (εγγραφή του πίνακα good) ανήκει σε ένα ή περισσότερους χρήστες οι οποίοι και την προσφέρουν.

- usersinexchange

  Η σχέση usersinexchange συνδέει τους πίνακες user και exchange. Ο λόγος πληθικότητας είναι Many-To-Many.

  Κάθε χρήστης(user), μπορεί να συμμετέχει σε 0 εώς Ν συναλλαγές ενώ κάθε συναλλαγή αφορά 1 εώς Ν χρήστες.

- exchangegrade

  Η σχέση exchangegrade συνδέει τους πίνακες exchange και grade. Ο λόγος πληθικότητας είναι One-To-Many.
  Σε κάθε ανταλλαγή αντιστοιχούν δύο βαθμολογίες γιατί κάθε μέλος της συναλλαγής βαθμολογεί το δεύτερο μέλος.

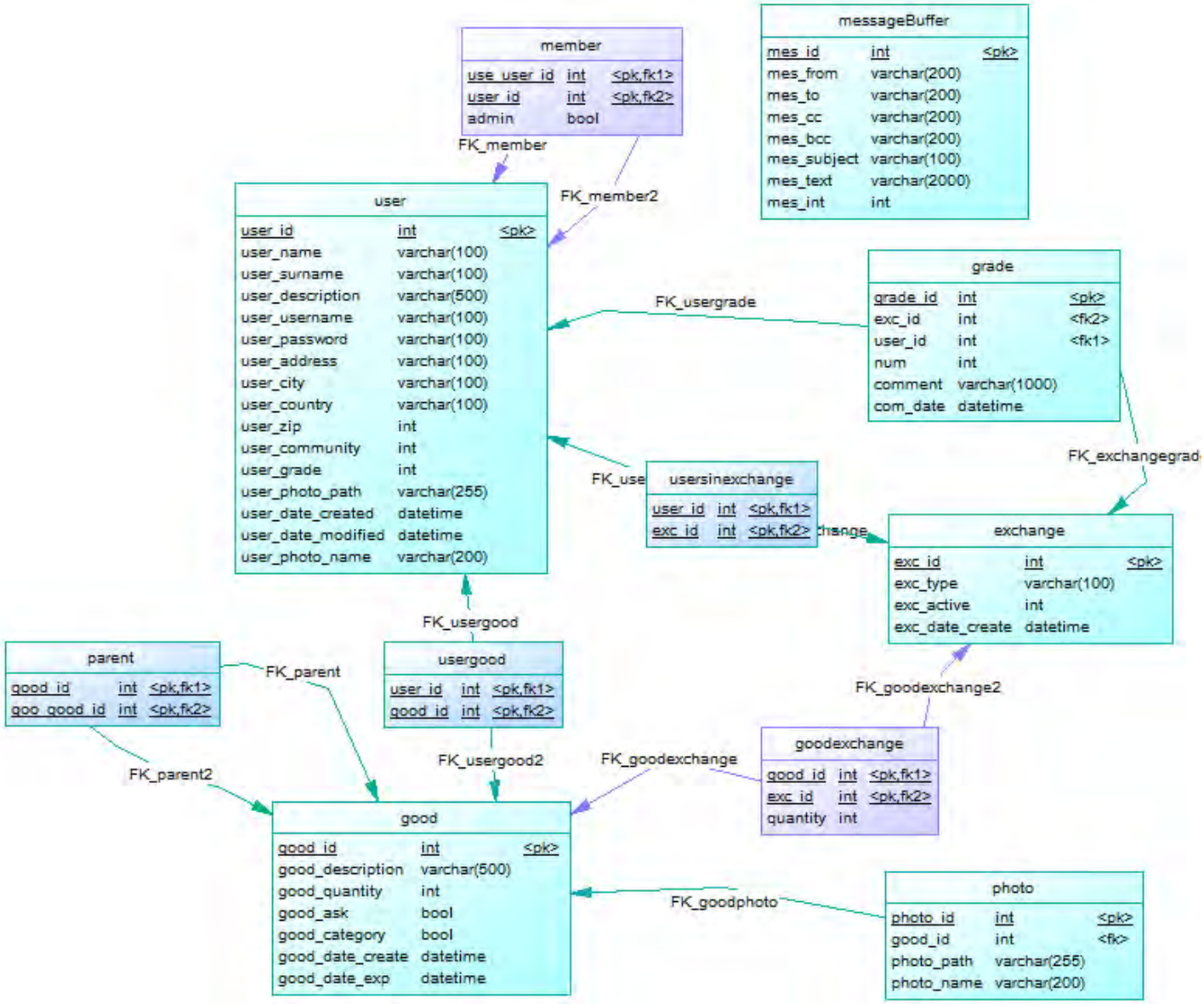
- usergrade

Η σχέση usergrade συνδέει τους πίνακες user και grade. Ο λόγος πληθικότητας της σχέσης αυτής είναι One-To-Many.

Κάθε χρήστης έχει λάβει από 0 έως Ν βαθμούς ανάλογα με τον αριθμό των συναλλαγών στις οποίες συμμετείχε ενώ κάθε βαθμός έχει δοθεί για ένα και μόνο χρήστη.

3.2 Φυσικό Μοντέλο

Από το εννοιολογικό μοντέλο της βάσης δεδομένων προκύπτει το παρακάτω φυσικό μοντέλο:



Στο παραπάνω σχήμα παρατηρούμε ότι κάθε σχέση με λόγο πληθικότητα Many-To-Many έχει αντικατασταθεί από ένα πίνακα ο οποίος αποτελείται από δύο ξένα κλειδιά(τα πρωτεύοντα κλειδιά των δύο πινάκων που συνδέει).

- Ο πίνακας member αποτελείται από τα πεδία user_id και use_user_id. Ορίζουμε αυθαίρετα ότι το πεδίο user_id περιέχει το πρωτεύον κλειδί του χρήστη, ο οποίος ανήκει στην κοινωνία με πρωτεύον κλειδί το use_user_id. Από αυτόν τον πίνακα μπορούμε να καταλάβουμε κάθε άτομο σε πόσες και ποιες κοινωνίες είναι μέλος και κάθε κοινωνία πόσα και ποια μέλη έχει δεδομένου ότι γνωρίζουμε το αναγνωριστικό του ατόμου ή της κοινωνίας που μας ενδιαφέρει. Ακόμη μπορούμε να ανακτήσουμε τον διαχειριστή της κάθε κοινωνίας αλλά και τις κοινωνίες των οποίων είναι διαχειριστής ο κάθε χρήστης.

- Ο πίνακας usersinexchange ενώνει τους πίνακες user και exchange και αποτελείται από τα πεδία user_id και exc_id.

  Από τον πίνακα αυτό μπορούμε να δούμε ο κάθε χρήστης σε ποιες συναλλαγές συμμετείχε και κάθε συναλλαγή μεταξύ ποιων χρηστών έχει γίνει, χρησιμοποιώντας το αναγνωριστικό του χρήστη που μας ενδιαφέρει ή το αναγνωριστικό της ανταλλαγής.

- Ο πίνακας goodexchange ενώνει τους πίνακες good και exchange και αποτελείται από τα πεδία good_id και exc_id.

  Από τον πίνακα αυτό μπορούμε να εξάγουμε την πληροφορία για ένα αντικείμενο σε ποιες συναλλαγές συμμετείχε. Ενώ μπορούμε ακόμη να δούμε για κάθε συναλλαγή ποια αντικείμενα συμπεριλαμβανόταν σε αυτή αλλά και την ποσότητα του κάθε αντικειμένου που είτε ζητήθηκε είτε προσφέρθηκε.

- Ο πίνακας parent αποτελείται από τα πεδία good_id και goo_good_id. Το good_id αποτελεί το πρωτεύον κλειδί του αγαθού το οποίο ανήκει στην κατηγορία με πρωτεύον κλειδί το goo_good_id.

- Ο πίνακας usergood αποτελείται από τα πεδία user_id και good_id. Ο πίνακας περιέχει μία εγγραφή για κάθε αγαθό που έχει καταχωρηθεί στο σύστημα και η εγγραφή αυτή αποτελείται από το πρωτεύον κλειδί του αγαθού και το πρωτεύον κλειδί του χρήστη στον οποίο ανήκει.

Στο φυσικό μοντέλο της βάσης έχουν προστεθεί και τα ξένα κλειδιά για τη συσχέτιση των πινάκων που συνδέονται με σχέσεις τύπου One-To-Many.

- Στον πίνακα grade έχουν προστεθεί τα ξένα κλειδιά user_id και exc_id. Τα ξένα αυτά κλειδιά μας δείχνουν σε ποιον χρήστη έχει δοθεί η συγκεκριμένη βαθμολογία και μετά από ποια συναλλαγή.
- Στον πίνακα photo έχει προστεθεί το ξένο κλειδί good_id το οποίο δηλώνει σε ποιο αντικείμενο αναφέρεται η συγκεκριμένη φωτογραφία

4 ΑΛΓΟΡΙΘΜΟΙ ΑΞΙΟΛΟΓΗΣΗΣ ΠΟΥ ΧΡΗΣΙΜΟΠΟΙΟΥΝΤΑΙ

4.1 E-bay

Το EBay χρησιμοποιεί ένα σύστημα ανάδρασης, το οποίο επιτρέπει τους χρήστες του συστήματος δημοπρασίας, πωλητές και αγοραστές,να αξιολογήσουν ο ένας τον άλλο βασισμένοι στην έκβαση των συναλλαγών. Υπάρχουν τρεις βαθμολογίες τις οποίες μπορεί να αφήσει ο κάθε χρήστης σαν αξιολόγηση της ανταλλαγής:

- Θετικό: +1 βαθμοί
- Ουδέτερο: 0 βαθμοί
- Αρνητικό: -1 βαθμοί

Θετική αξιολόγηση σημαίνει ότι η συναλλαγή ήταν επιτυχημένη και όλα κύλησαν ομαλά.

Ουδέτερη αξιολόγηση σημαίνει ότι η συναλλαγή έγινε κανονικά αλλά με κάποια προβλήματα.

Και τέλος αρνητική αξιολόγηση σημαίνει ότι υπήρξαν σοβαρά προβλήματα κατά την προσπάθεια συναλλαγής.

Μόλις ένας χρήστης μαζέψει ένα ορισμένο σύνολο βαθμών, παίρνει ένα αστεράκι το οποίο αντικατοπτρίζει την αξιοπιστία του. Υπάρχουν διάφορα αστέρια διαφορετικών χρωμάτων τα οποία μπορεί να λάβει ο χρήστης σύμφωνα με τους βαθμούς που έχει λάβει από τις συναλλαγές στις οποίες συμμετείχε.

- Κίτρινο αστέρι( ) = 10 εώς 49 πόντοι
- Μπλέ αστέρι ( ) = 50 εώς 99 πόντοι
- Τιρκουάζ αστέρι( ) = 100 εώς 499 πόντοι
- Μωβ αστέρι ( ) = 500 εώς 999 πόντοι
- Κόκκινο αστέρι ( ) = 1,000 εώς 4,999 πόντοι
- Πράσινο αστέρι ( ) = 5,000 εώς 9,999 πόντοι
- Κίτρινο γραμμοσκιασμένο αστέρι ( ) = 10,000 εώς 24,999 πόντοι

- Τιρκουάζ γραμμοσκιασμένο αστέρι () = 25,000 εώς 49,999 πόντοι
- Μωβ γραμμόσκιασμένο αστέρι () = 50,000 εώς 99,999 πόντοι
- Κόκκινο γραμμοσκιασμένο αστέρι () = 100,000 και υψηλότερα

Βασιζόμενος στο σύστημα αυτό, κάθε χρήστης του συστήματος μπορεί να επιλέξει με ποιους χρήστες θα συνεργαστεί κοιτώντας απλά τη βαθμολογία τους.

4.2 P2P ΔΙΚΤΥΑ

Kazaa

Το Kazaa, ένα πολύ διάσημο P2P file sharing σύστημα διαθέτει έναν αλγόριθμο αξιολόγησης.

Το Kazaa διαθέτει ένα χαρακτηριστικό γνώρισμα που το ονομάζει βαθμό ακεραιότητας (Integrity Rating) που επιτρέπει στους χρήστες που μοιράζονται αρχεία να βαθμολογήσουν τα αρχεία τους από την άποψη των τεχνικών αξιών τους, όπως εάν τα αρχεία έχουν τα ακριβή μεταδεδομένα και είναι υψηλής ποιότητας. Προκειμένου να καθοδηγηθούν άλλοι χρήστες προς τα ποιοτικότερα αρχεία διαθέσιμα για download, οι χρήστες του Kazaa ενθαρρύνονται να δώσουν ένα βαθμό ακεραιότητας στα αρχεία τους και να διαγράφουν τα αρχεία που δεν πρέπει να μοιραστούν αλλά δεν είναι απαραίτητο ένας χρήστης να αξιολογήσει τα αρχεία που μοιράζεται προκειμένου να συμμετέχει στο δίκτυο του Kazaa.

Υπάρχουν τέσσερις βαθμοί ακεραιότητας για τη βαθμολογία των αρχείων:

- Άριστος: Το αρχείο έχει πλήρη μεταδεδομένα και είναι μιας υψηλής ποιότητας.
- Μέσος όρος: Το αρχείο έχει μερικά μεταδεδομένα και το αρχείο είναι μέτριας ποιότητας.
- Φτωχός: Το αρχείο είναι κακής ποιότητας και δεν έχει κανένα μεταδεδομένο.
- Διαγράψτε το αρχείο: Το αρχείο δεν πρέπει να μοιραστεί

Στο σύστημα διαχείρισης φήμης του Kazaa, κάθε χρήστης έχει ένα επίπεδο συμμετοχής που είναι βασισμένο στην ποιότητα και τον αριθμό αρχείων που μοιράζεται. Το επίπεδο συμμετοχής ενός χρήστη είναι ένας αριθμός που απεικονίζει τους τρόπους με τους οποίους ο χρήστης έχει χρησιμοποιήσει το Kazaa για να «ανεβάσει» και να «κατεβάσει»

αρχεία. Το επίπεδο συμμετοχής ενός χρήστη μπορεί να είναι μέσα στη μια από έξι κατηγορίες και προορίζεται να ανταμείψει τους χρήστες που μοιράζονται αρχεία στα οποία έχουν δώσει βαθμό ακεραιότητας παρέχοντάς τους αυξημένο εύρος ζώνης που μπορούν να χρησιμοποιήσουν για να «κατεβάσουν» αρχεία από άλλους χρήστες του δικτύου.

Το επίπεδο συμμετοχής ενός χρήστη υπολογίζεται σύμφωνα με τον παρακάτω τύπο:

$$plevel_i = \frac{uploaded_i}{downloaded_i} \times 100$$

$plevel_i$ είναι το επίπεδο συμμετοχής του χρήστη i

$uploaded_i$ είναι η ποσότητα των δεδομένων (σε megabytes) που ο χρήστης i έχει διαθέσει στους άλλους χρήστες

$downloaded_i$ είναι η ποσότητα των δεδομένων (σε megabytes) που ο χρήστης i έχει «κατεβάσει» από άλλους χρήστες

Στον παραπάνω τύπο, αν ο χρήστης «ανεβάσει» ένα αρχείο στο οποίο δεν έχει αναθέσει ένα βαθμό ακεραιότητας, αυτό το αρχείο θα ληφθεί υπ'όψην με το μισό του κανονικού του μεγέθους κατά τον υπολογισμό του επιπέδου συμμετοχής του συγκεκριμένου χρήστη.

Στο σύστημα του Kazaa ο βαθμός συμμετοχής ενός χρήστη μπορεί να πάρει τιμές από 0 εώς 1000 πόντους.

Μετά την εγκατάσταση του λογισμικού για τη διαμοίραση αρχείων του Kazaa, κάθε χρήστης ξεκινά να χρησιμοποιεί το σύστημα με ένα μέσο βαθμό συμμετοχής που έχει οριστεί στους 100 πόντους και η βαθμολογία αυτή μπορεί να κινηθεί ανοδικά ή καθοδικά ανάλογα με το πως χρησιμοποιεί ο χρήστης το δίκτυο όπως περιγράφεται ανωτέρω.

Ένα σημαντικό θέμα με το σύστημα διαχείρισης φήμης του Kazaa είναι ότι έχει ως σκοπό να ανταμείψει τους χρήστες που καταδεικνύουν καλή συμπεριφορά αλλά δεν τιμωρεί εκείνους που δεν το κάνουν ή δεν μπορούν να το κάνουν. Το σύστημα επίσης δεν παρέχει έναν μηχανισμό από τον οποίο οι χρήστες μπορούν να εκτιμήσουν ο ένας τα αρχεία του άλλου βασισμένοι στην ποιότητά τους, αντ' αυτού στηρίζεται στους χρήστες να βαθμολογήσουν τα δικά τους αρχεία. Αυτή η προσέγγιση στη διαχείριση φήμης αφήνει το δίκτυο τρωτό στους κακόβουλους χρήστες επειδή αυτοί οι χρήστες μπορούν να δώσουν σε κάθε αρχείο που μοιράζονται μια υψηλή βαθμολογία ακεραιότητας ακόμα κι αν όλα τα αρχεία τους είναι ψευδή αρχεία. Δεδομένου ότι αυτοί οι κακόβουλοι χρήστες δεν τιμωρούνται ποτέ για τη διανομή των ψευδών αρχείων, μπορούν να συνεχίσουν να παράγουν έναν απεριόριστο αριθμό υπέρ-εκτιμημένων, ψευδών αρχείων χωρίς να τους απαγορεύσει ποτέ κανείς να τα διακινούν.

EigenTrust

Το EigenTrust είναι ένα τα λίγα προτεινόμενα συστήματα διαχείρισης φήμης για τα peer-to-peer δίκτυα. Αυτό το σύστημα αναπτύχθηκε από τους Sepandar D. Kamvar, Mario T. Schlosser, και Hector Garcia-Molina στο Πανεπιστήμιο του Stanford.

Στο EigenTrust, κάθε χρήστης i βαθμολογεί έναν άλλο χρήστη j από τον οποίο προσπαθεί να «κατεβάσει» αρχεία βαθμολογώντας κάθε μεταφορά αρχείου είτε ως θετική (*tr(i,j)* = 1) είτε ως αρνητική (*tr(i,j)* = -1), ανάλογα με το εάν το αρχείο που του παρείχε ο j ήταν αυθεντικό ή όχι. Το σύνολο των βαθμών που προέρχονται από τις μεταφορές που έκανε ο i από τον j καλείται τοπική τιμή εμπιστοσύνης (s_{ij}).

Για να αθροίσουμε αυτές τις τοπικές τιμές εμπιστοσύνης γύρω από το δίκτυο, τις ομαλοποιούμε έτσι ώστε οι κακόβουλοι χρήστες να μην είναι σε θέση να ορίσουν αυθαίρετα υψηλές τιμές εμπιστοσύνης σε άλλους κακόβουλους χρήστες και να υπονομεύσουν το σύστημα του EigenTrust.

Η ομαλοποιημένη τοπική αξία εμπιστοσύνης (cij) καθορίζεται ως εξής:

$$c_{ij} = \frac{\max(s_{ij}, 0)}{\sum_j \max(s_{ij}, 0)}$$

Το ανωτέρω άθροισμα, μαζί με άλλα αθροίσματα στον αλγόριθμο του EigenTrust, χρησιμοποιεί μια μαθηματική σύμβαση αποκαλούμενη άθροισμα Einstein, που υπονοεί ότι όταν εμφανίζεται περισσότερο από μία φορά ένας δείκτης σε μια έκφραση, η έκφραση αθροίζεται σε όλες τις πιθανές τιμές για εκείνο τον δείκτη. Προκειμένου να χρησιμοποιηθεί αυτή η σύμβαση, πρέπει να είναι σαφές πέρα από ποιο σημείο οι δείκτες πρέπει να αθροιστούν. Παραδείγματος χάριν, στον παρονομαστή του cij ανωτέρω, αθροίζουμε σιωπηρά όλες τις ανώτατες (sij, 0) τιμές σε όλες τις πιθανές τιμές για το δείκτη j, οι οποίες είναι όλοι οι χρήστες j που ο χρήστης i έχει αλληλεπιδράσει μαζί τους.

Κανονικοποιώντας την συνολική τιμή της εμπιστοσύνης ενός χρήστη με αυτόν τον τρόπο εξασφαλίζεται ότι όλες οι τιμές, ακόμη και των κακόβουλων χρηστών θα είναι μεταξύ 0 και 1. Πρέπει να σημειωθεί ότι εάν:

$$\Sigma_j \max(s_{ij}) = 0$$

το cij είναι απροσδιόριστο. Εάν αυτό συμβαίνει, ξέρουμε ότι ο χρήστης i είτε δεν «κατεβάζει» από οποιοδήποτε άλλο χρήστη είτε δίνει ένα αποτέλεσμα 0 σε όλους τους άλλους χρήστες επειδή δεν εμπιστεύεται κανέναν άλλο. Σε αυτήν την περίπτωση, είναι χρήσιμο να υπάρξει ένα σύνολο προ-εμπιστευόμενων χρηστών. Κατόπιν μπορούμε να καθορίσουμε:

$$p_i = 1/P \text{ if } i \in P, \text{ else } p_i = 0$$

Εάν το c_{ij} δεν ορίζεται τότε ο χρήστης i θα επιλέξει να εμπιστευθεί το σύνολο των προ-εμπιστευόμενων χρηστών.

Επιθυμούμε να αθροίσουμε τις κανονικοποιημένες τοπικές τιμές εμπιστοσύνης. Ένας φυσικός τρόπος να γίνει αυτό σε ένα κατανεμημένο περιβάλλον είναι για το κόμβο i να ρωτήσει τους γνωστούς του για τις απόψεις τους για άλλους κόμβους. Θα ήταν λογικό να σταθμίσει τις απόψεις τους σύμφωνα με την εμπιστοσύνη που έχει στον καθένα

$$\mathbf{t_{ik} = \Sigma_j c_{ij} c_{jk}}$$

όπου t_{ij} είναι η εμπιστοσύνη που θα δείξει ο χρήστης i στον χρήστη k βασιζόμενος στους γνωστούς του χρήστες.
Μπορούμε να παραστήσουμε την πληροφορία αυτή με έναν πίνακα:
Ορίζουμε τον πίνακα C ο οποίος περιέχει τις τιμές $[c_{ij}]$ και t_i είναι το διάνυσμα που περιέχει τις τιμές t_{ik}. Στη συνέχεια $\vec{t} = (C^T)\vec{c_i}$. Με αυτόν τον τρόπο κάθε χρήστης κερδίζει μία άποψη για το δίκτυο η οποία είναι ευρύτερη της δικής του εμπειρίας. Παρόλα αυτά, οι τιμές εμπιστοσύνης που είναι αποθηκευμένες στο χρήστη i ακόμη αντανακλούν μόνο την εμπειρία του χρήστη i και των φίλων του. Για να αποκτήσει ο χρήστης i μια ευρύτερη άποψη μπορεί να επιθυμεί να ρωτήσει και τους φίλους των φίλων του $\vec{t} = (C^T)^2\vec{c_i}$. Εάν συνεχίσει με αυτόν τον τρόπο $\vec{t} = (C^T)^n\vec{c_i}$ θα αποκτήσει μια πλήρη άποψη του δικτύου μετά από n επαναλήψεις (δεχόμενοι ότι ο C δεν μειώνεται και είναι απεριοδικός)

5 ΑΛΓΟΡΙΘΜΟΣ ΑΞΙΟΛΟΓΗΣΗΣ

Στην κοινωνία μας συναλλάσσονται οντότητες, άτομα ή κοινωνίες, που τις περισσότερες φορές δεν γνωρίζονται μεταξύ τους(μπορεί να μιλάμε για άτομα που ζουν σε διαφορετικές πόλεις και δεν έχουν συναντηθεί ποτέ). Στην περίπτωση αυτή δεν μπορούν να εμπιστεύονται τυφλά ο ένας τον άλλο για την αξιοπιστία της συναλλαγής.

Κάθε οντότητα, άτομο ή κοινωνία, που συμμετέχει στο σύστημά μας διατηρεί ένα βαθμό αξιοπιστίας. Ο βαθμός αυτός διαμορφώνεται κατά την δραστηριοποίηση της οντότητας αυτής στο σύστημα. Μετά από κάθε συναλλαγή τα μέλη που μετείχαν στη συναλλαγή αυτή έχουν τη δυνατότητα να βαθμολογήσουν το χρήστη με τον οποίο έκαναν τη συναλλαγή αλλά και να αφήσουν ένα μικρό σχόλιο βασιζόμενοι στη εμπειρία τους από τη συναλλαγή αυτή.

Πιο συγκεκριμένα, ανάλογα με το είδος της συναλλαγής είτε θα βαθμολογηθούν και τα δύο μέρη, οντότητες, της συναλλαγής είτε το ένα.

1. Στην περίπτωση της ανταλλαγής, όπου ανταλλάσσονται αντικείμενα ή υπηρεσίες μεταξύ της οντότητας Α και Β, μετά το πέρας της δοσοληψίας η οντότητα Α βαθμολογεί την οντότητα Β και το αντίστροφο.

2. Στην περίπτωση της δωρεάς, όπου μία οντότητα Α δωρίζει ένα αντικείμενο ή μία υπηρεσία που κατέχει στην οντότητα Β τότε η οντότητα Α λαμβάνει ένα βαθμό από την οντότητα Β αλλά η Β δεν βαθμολογείται από την Α.

3. Στην περίπτωση της κοινοχρησίας, βαθμολογούνται και οι δύο οντότητες.

5.1 Πως υπολογίζεται ο βαθμός αξιοπιστίας των μελών

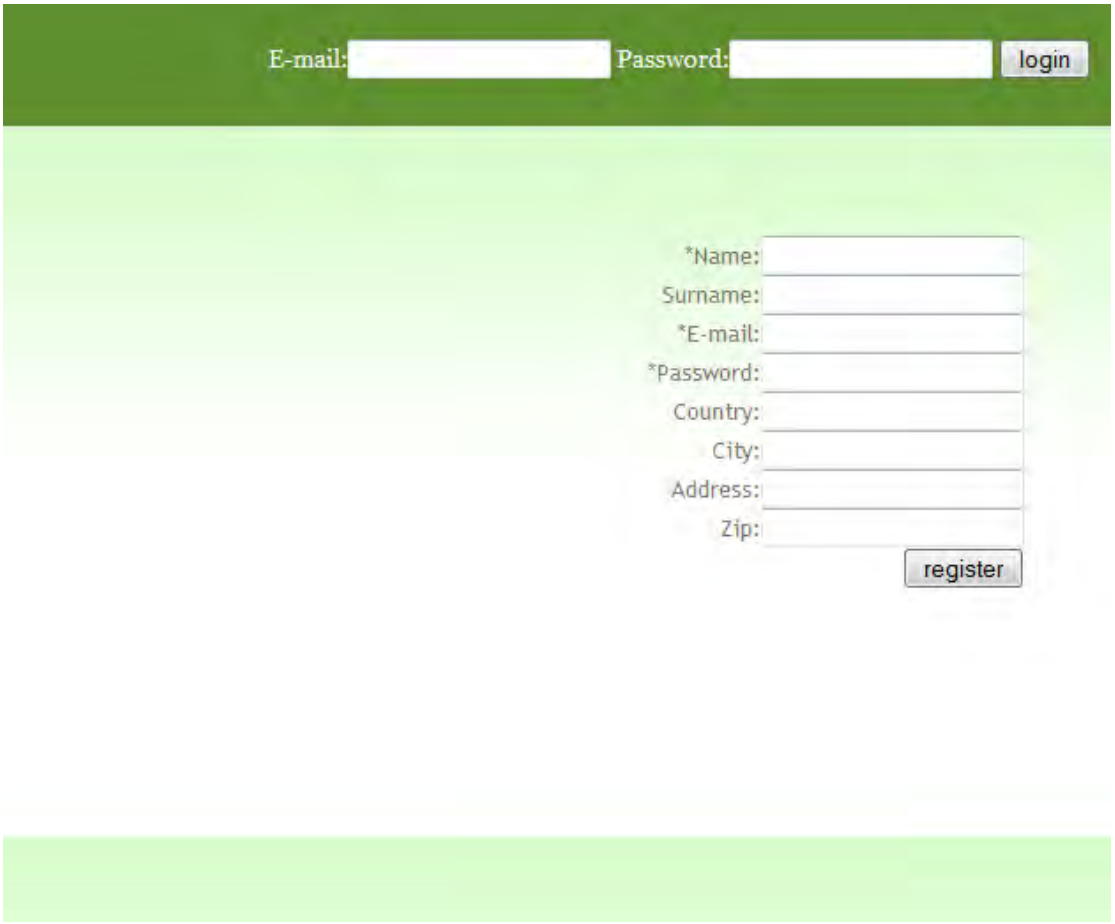
Η βαθμολογία λαμβάνει ένας χρήστης μπορεί να είναι θετική, ουδέτερη ή αρνητική. Η αξιοπιστία του κάθε χρήστη προκύπτει από το άθροισμα των βαθμολογιών που έχει λάβει από τις συναλλαγές στις οποίες συμμετείχε.
Πιο συγκεκριμένα κάθε μέλος λαμβάνει:
+1 στη συνολική του βαθμολογία για κάθε θετική αξιολόγηση που δέχτηκε
0 για κάθε ουδέτερη και
-1 για κάθε αρνητική αξιολόγηση

5.2 Ποσοστό αξιοπιστίας

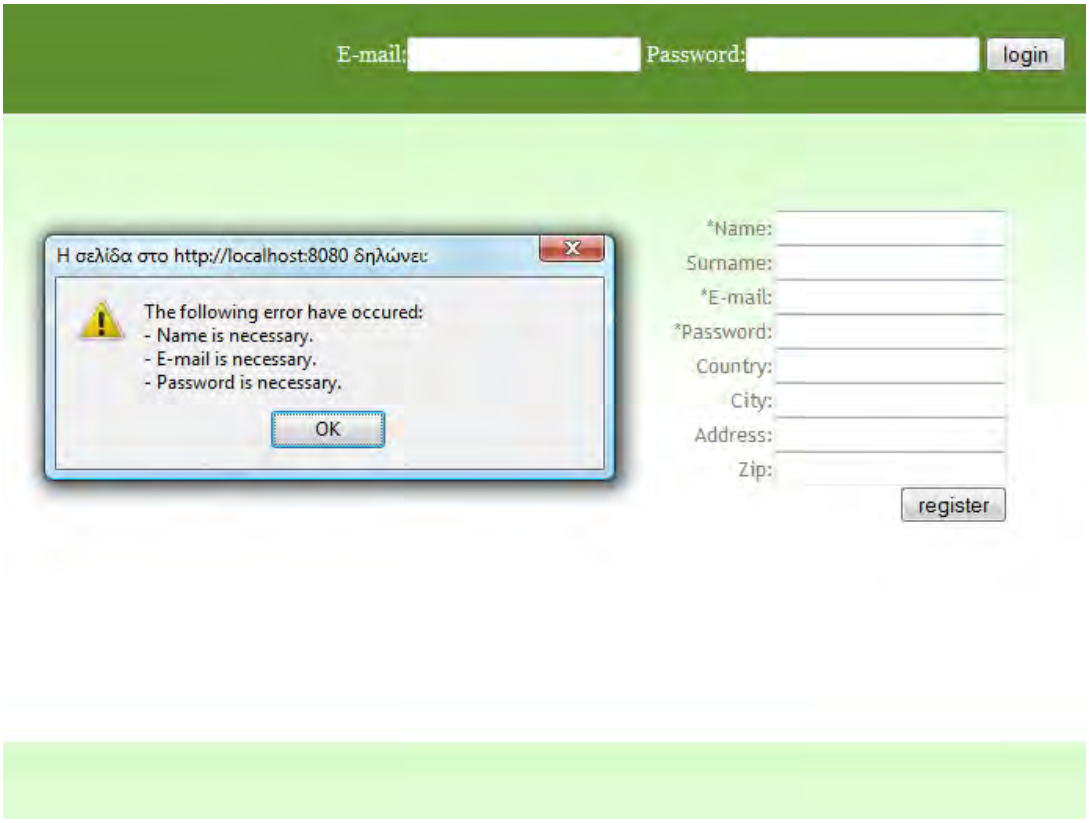
Εκτός από το βαθμό αξιοπιστίας υπολογίζεται και το ποσοστό αξιοπιστίας του κάθε χρήστη. Το ποσοστό αξιοπιστίας υπολογίζεται διαιρώντας τον συνολικό αριθμό των θετικών αξιολογήσεων με το άθροισμα των θετικών και αρνητικών αξιολογήσεων που έχει λάβει ο χρήστης.
(Θετικές)÷(Θετικές + Αρνητικές)

6 Περιγραφή του συστήματος

Στην αρχική σελίδα του συστήματος υπάρχουν δύο φόρμες, μία για την είσοδο μελών που είναι ήδη εγγεγραμμένα στο σύστημα, με την οποία δίνοντας το e-mail και το password ο χρήστης μπορεί να κάνει login στο σύστημα και μία δεύτερη φόρμα με την οποία κάθε νέος χρήστης μπορεί να εγγραφεί.

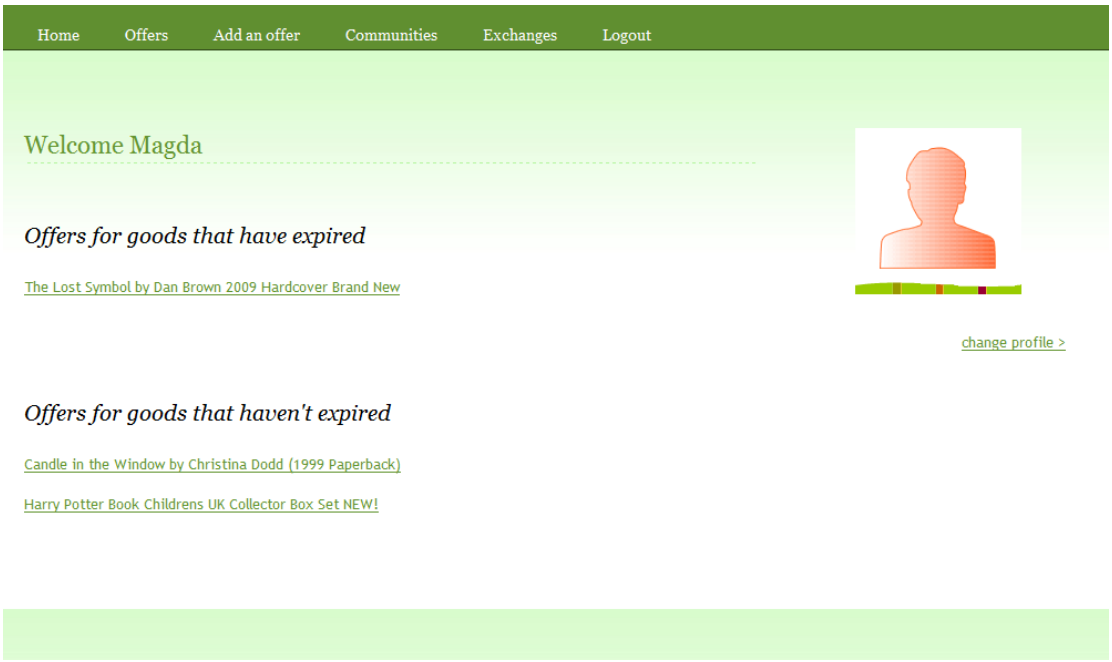


Για την εγγραφή στο σύστημα απαιτείται από το χρήστη να δώσει το όνομά του, το επίθετό του, το e-mail, ένα password, τη χώρα στην οποία διαμένει, την πόλη, τη διεύθυνση και τον ταχυδρομικό κώδικα. Από αυτά απαραίτητα είναι μόνο το όνομα, το οποίο μπορεί να είναι κάποιο ψευδώνυμο αλλά θα είναι αυτό που θα αντιπροσωπεύει το χρήστη στο σύστημα, το e-mail και το password. Τα υπόλοιπα στοιχεία ζητούνται από το χρήστη γιατί θα φανούν χρήσιμα στην περίπτωση μίας συναλλαγής με κάποιο άλλο χρήστη. Στην περίπτωση που προσπαθήσει ένας χρήστης είτε να δώσει λιγότερα στοιχεία για την εγγραφή του είτε να εισέλθει στο σύστημα χωρίς να δώσει κάποιο από τα στοιχεία εισόδου ανοίγει ένα ενημερωτικό παράθυρο που τον ενημερώνει για το λάθος του.



Για κάθε αντικείμενο που εισάγει ο χρήστης στο σύστημα με σκοπό να το προσφέρει για ανταλλαγή δίνει και μια ημερομηνία λήξης. Η ημερομηνία αυτή δηλώνει τη μέρα και ώρα μέχρι την οποία οι υπόλοιποι χρήστες μπορούν να καταθέτουν προσφορές για το αντικείμενο αυτό.

Μόλις επιτευχθεί η εισαγωγή του χρήστη στο σύστημα, εμφανίζονται στην οθόνη τα αντικείμενα τα οποία έχει εισάγει στο σύστημα για ανταλλαγή ταξινομημένα ανάλογα με την ημερομηνία λήξης τους.



Δεξιά της σελίδας εμφανίζεται η φωτογραφία του προφίλ του χρήστη και ο σύνδεσμος “change profile” . Πατώντας αυτό το σύνδεσμο ο χρήστης μεταβαίνει στη σελίδα από την οποία μπορεί να αλλάξει τα στοιχεία που έχει δώσει στο σύστημα. Στα πεδία της φόρμας έχουν εισαχθεί τα στοιχεία που είχε εισάγει ο χρήστης κατά την τελευταία ανανέωση του προφίλ του. Μεταβάλει τα στοιχεία που επιθυμεί και στη συνέχεια υποβάλει τις αλλαγές.

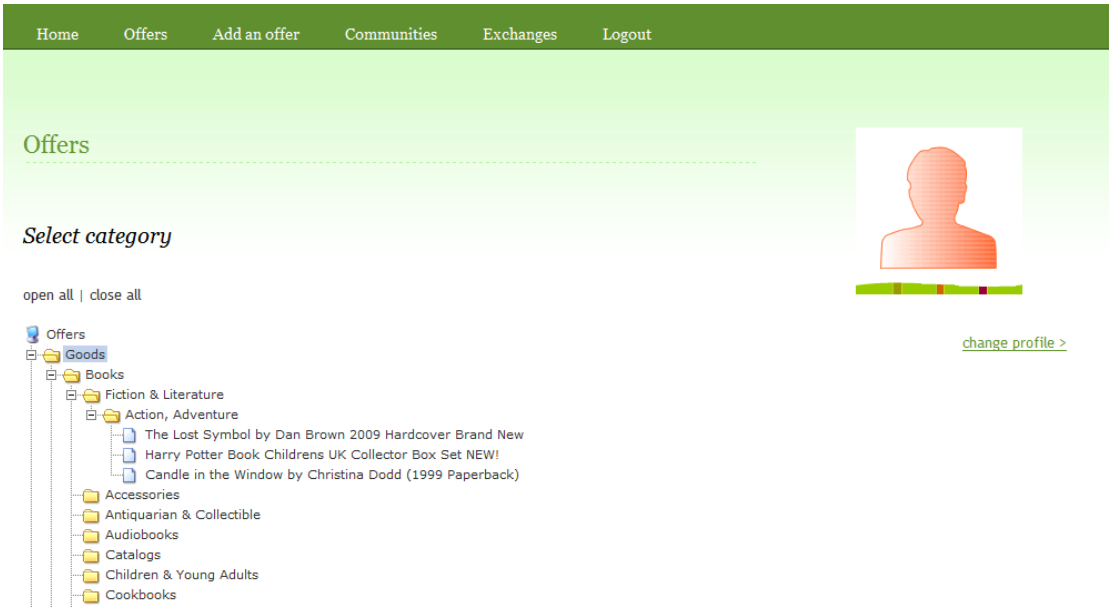
Χρησιμοποιώντας τους συνδέσμους που βρίσκονται κάτω από τον τίτλο Offers for goods that have expired ο χρήστης μπορεί να δει τις προσφορές που έχουν κάνει οι χρήστες για το κάθε αντικείμενο και επιλέγει ποια από αυτές θέλει να πραγματοποιηθεί.

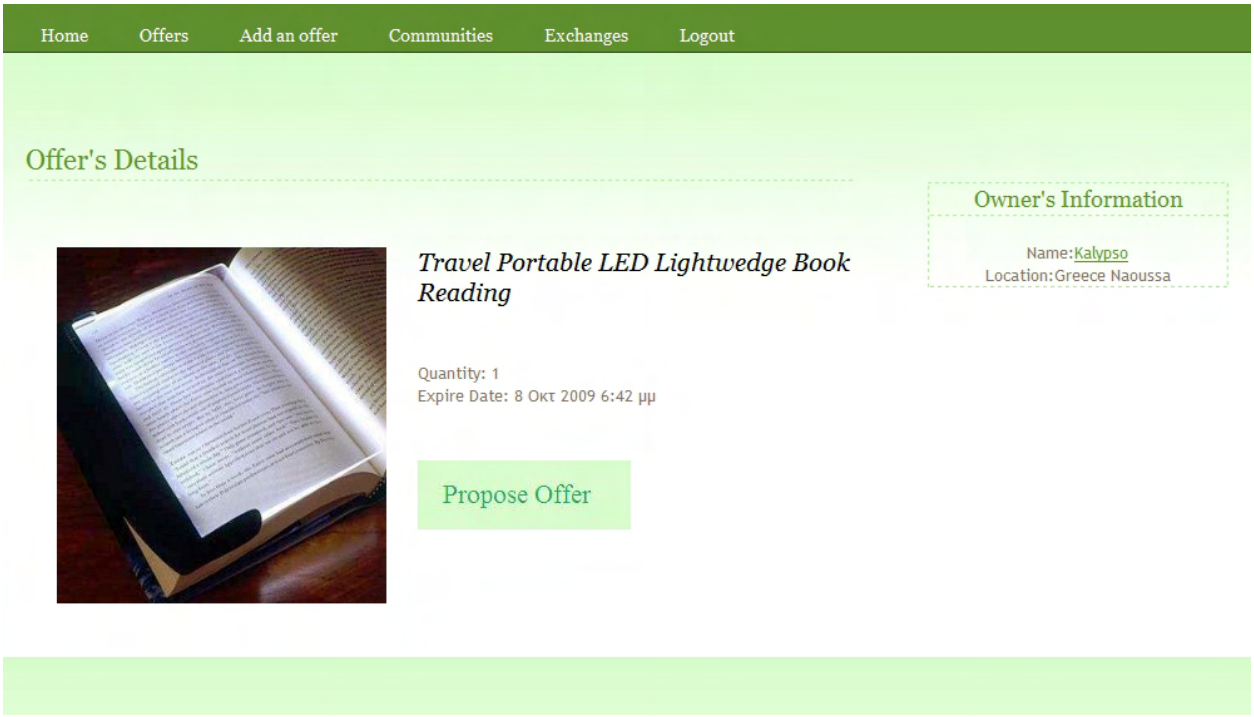
Για να μπορέσει ο χρήστης να πάρει την απόφασή του πιο εύκολα, τα ονόματα των αντικειμένων είναι σύνδεσμοι οι οποίοι οδηγούν στη σελίδα περιγραφής τους, όπως και το όνομα του χρήστη που προσφέρει το αντικείμενο είναι σύνδεσμος που οδηγεί στη σελίδα περιγραφής του χρήστη. Δίπλα στο όνομα του χρήστη σε παρένθεση βρίσκεται η βαθμολογία που έχει συγκεντρώσει ο χρήστης από τις ανταλλαγές στις οποίες συμμετείχε μέχρι τώρα.

Με τους συνδέσμους που βρίσκονται κάτω από τον τίτλο Offers for goods that haven't expired έχει τη δυνατότητα να δει ποιες προσφορές έχουν κατατεθεί μέχρι αυτή τη στιγμή για το κάθε αντικείμενο που προσφέρει, χωρίς όμως να μπορεί να επιλέξει μία από αυτές τις προσφορές γιατί δεν έχει περάσει ακόμη η ημερομηνία μέχρι την οποία οι υπόλοιποι χρήστες μπορούν να καταθέτουν τις προσφορές τους(ημερομηνία λήξης).
Στο menu που βρίσκεται στην κορυφή της αρχικής σελίδας υπάρχουν οι επιλογές:

- Home
- Offers
- Add an offer
- Communities
- Exchanges
- Logout

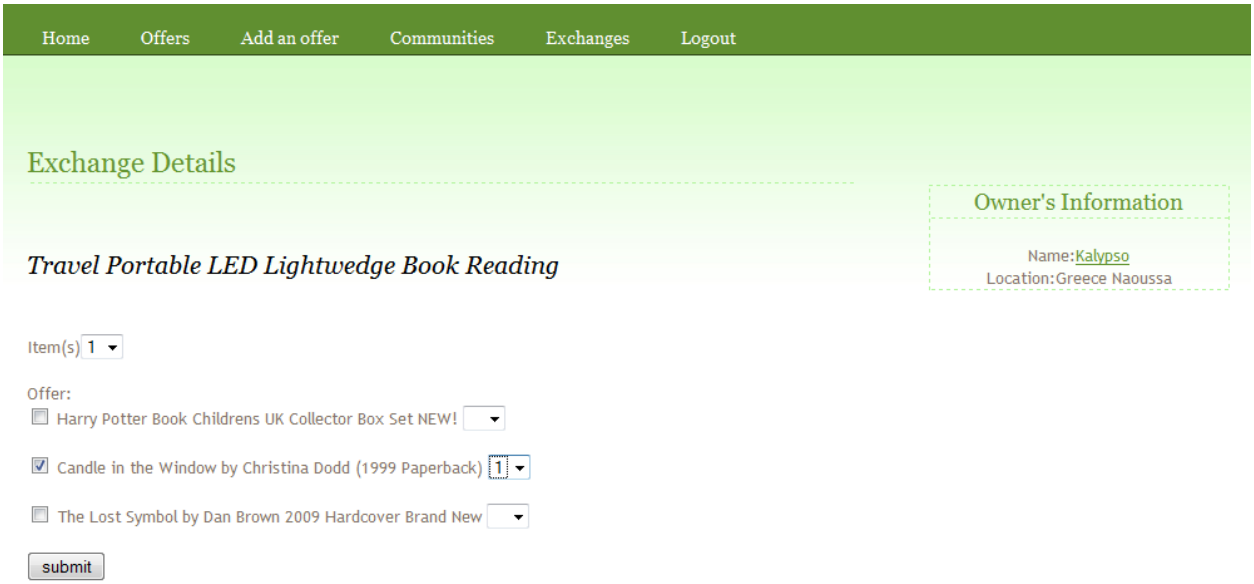
Πατώντας στην επιλογή Offers εμφανίζονται τα αντικείμενα που προσφέρονται από τους χρήστες ταξινομημένα σε κατηγορίες. Κάθε αντικείμενο είναι ένας σύνδεσμος ο οποίος οδηγεί στην σελίδα περιγραφής του αντικειμένου.





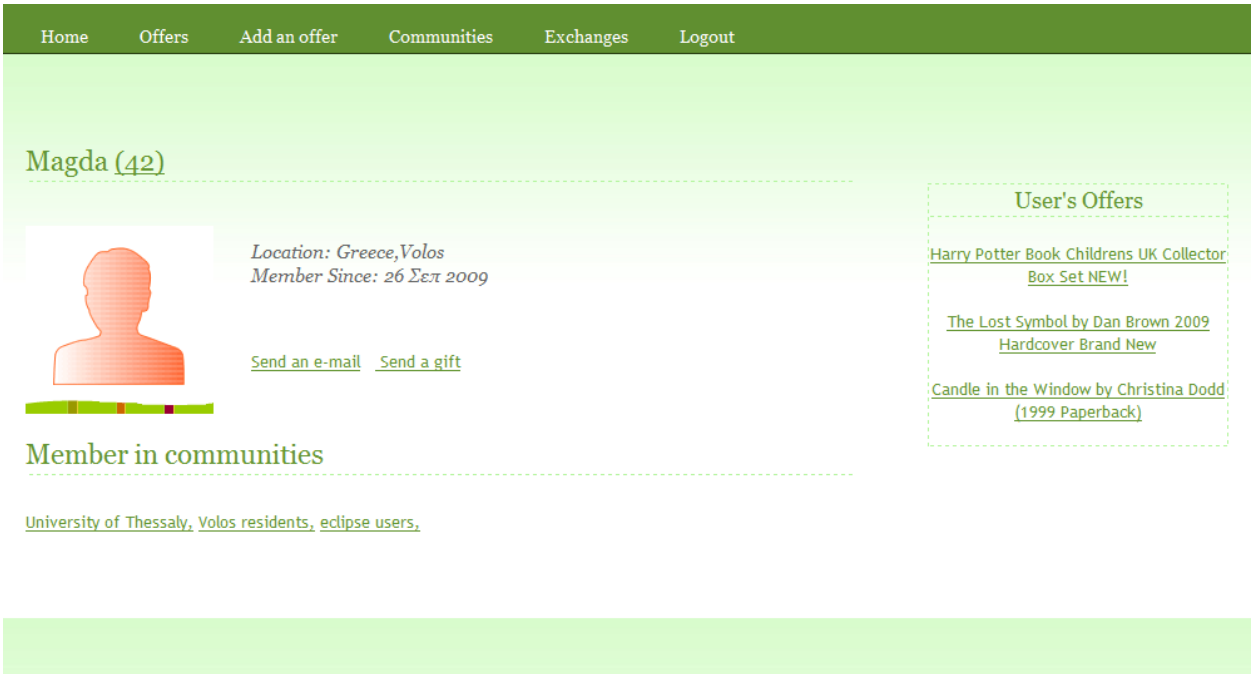
Στη σελίδα περιγραφής του αντικειμένου υπάρχει μία φωτογραφία του αντικειμένου που προσφέρεται, η ποσότητα, η ημερομηνία λήξης της προσφοράς και πληροφορίες για τον ιδιοκτήτη του αντικειμένου. Το όνομα του ιδιοκτήτη είναι σύνδεσμος που οδηγεί στην σελίδα του προφίλ του χρήστη ενώ πατώντας Propose Offer μπορεί ο χρήστης να καταθέσει προσφορά για το συγκεκριμένο αντικείμενο.

Πατώντας το Propose Offer οδηγούμαστε στην παρακάτω σελίδα:



Στην επιλογή Item βάζουμε την ποσότητα του αγαθού που ζητάμε. Κάτω από το Offer εμφανίζονται τα αντικείμενα που έχουμε στην κατοχή μας και επιλέγουμε αυτά που θέλουμε να προσφέρουμε και στην ποσότητα που θέλουμε.

Πατώντας πάνω στο όνομα του ιδιοκτήτη του αντικειμένου εμφανίζεται η σελίδα του προφίλ του.

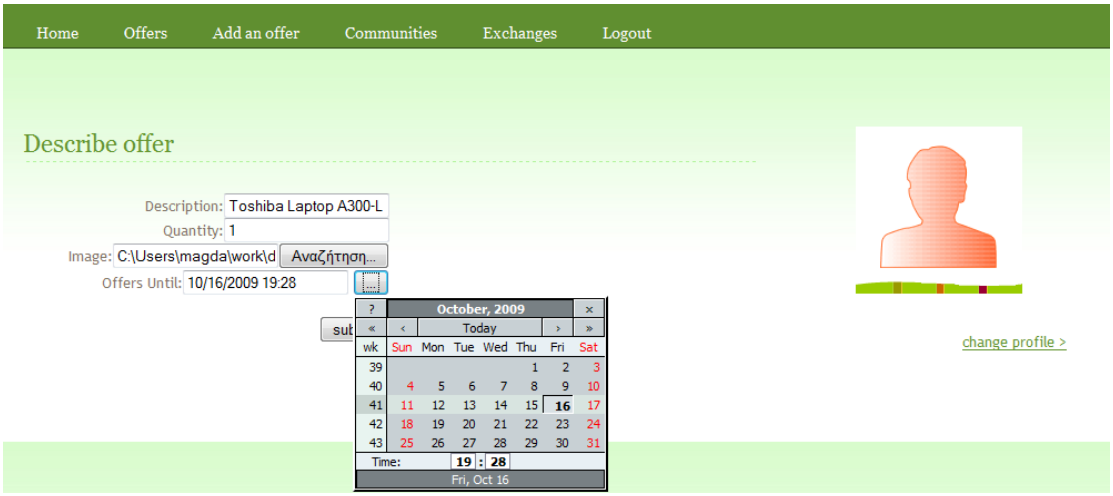


Από τη σελίδα αυτή μπορούμε να δούμε τον τόπο διαμονής του χρήστη, την ημερομηνία εγγραφής του στο σύστημα, τις κοινωνίες στις οποίες ανήκει αλλά και τα αντικείμενα τα οποία προσφέρει. Ακόμη πατώντας πάνω στο send an e-mail μπορούμε να στείλουμε

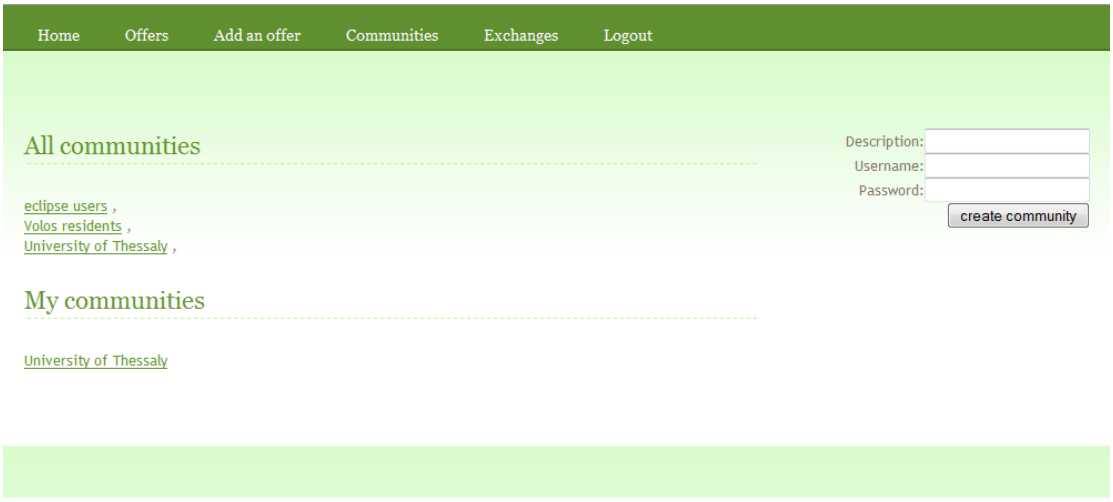
μήνυμα στο χρήστη μέσα από την εφαρμογή, ενώ πατώντας πάνω στο send a gift εμφανίζεται λίστα των αγαθών που κατέχουμε και επιλέγουμε ποια θέλουμε να χαρίσουμε στο χρήστη.

Ακόμη δίπλα από το όνομα του χρήστη εμφανίζεται η βαθμολογία του η οποία είναι σύνδεσμος ο οποίος μας οδηγεί σε σελίδα στην οποία μπορούμε να δούμε τον αριθμό των θετικών, αρνητικών και ουδέτερων βαθμολογιών που έχει λάβει ο χρήστης αλλά και τα σχόλια τα οποία έχουν αφήσει άλλοι χρήστες οι οποίοι έχουν πραγματοποιήσει κάποια συναλλαγή με αυτό το χρήστη.

Πατώντας την επιλογή Add an Offer εμφανίζεται η σελίδα με το δέντρο των κατηγοριών και πατώντας πάνω σε μία κατηγορία ο χρήστης μπορεί να εισάγει ένα αντικείμενο στην κατηγορία αυτή. Μόλις επιλεγεί η κατηγορία ο χρήστης περνάει στη σελίδα όπου δίνει την περιγραφή του αντικειμένου, την ποσότητα, τη φωτογραφία του αντικειμένου και την ημερομηνία μέχρι την οποία οι υπόλοιποι χρήστες μπορούν να υποβάλουν τις προσφορές τους.



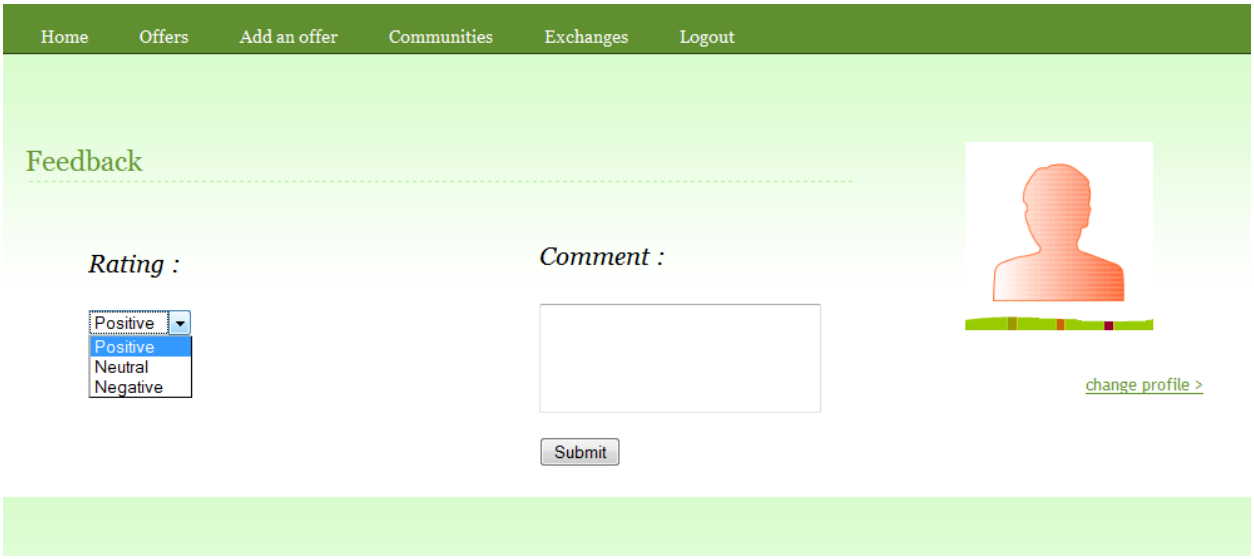
Αν ο χρήστης πατήσει την επιλογή Communities εμφανίζονται όλες οι κοινωνίες που έχουν σχηματιστεί από τους χρήστες στο σύστημα και όλες οι κοινωνίες στις οποίες είναι μέλος ο χρήστης.



Σε αυτή τη σελίδα δίνεται η δυνατότητα στο χρήστη να δημιουργήσει μία νέα κοινωνία. Αν ο χρήστης πατήσει πάνω στο όνομα κάποιας κοινωνίας μεταβαίνει στην σελίδα με την περιγραφή και το σκοπό για τον οποίο δημιουργήθηκε η κοινωνία. Ακόμη μπορεί να δει ποιοι χρήστες είναι μέλη αυτής της κοινωνίας, ποιος είναι ο διαχειριστής της αλλά και να γίνει μέλος της.

Ο χρήστης πατώντας την επιλογή Exchanges μπορεί να δει την πορεία των προσφορών που έχει κάνει αυτός σε άλλους χρήστες(αν τις αποδέχτηκαν ή εάν τις απέρριψαν)και τις προσφορές που κατέθεσαν άλλοι χρήστες σε αυτόν. Ακόμη μπορεί να δει τις δωρεές και τις προτάσεις για κοινοχρησία αγαθών που είτε πρότειναν άλλοι χρήστες σε αυτόν είτε το αντίθετο και την πορεία αυτών των προτάσεων.

Επιλέγοντας μία ανταλλαγή η οποία έχει ολοκληρωθεί (επιβεβαιώθηκε και από τα δύο μέλη) μπορεί να προχωρήσει στην βαθμολόγηση του χρήστη με τον οποίο πραγματοποίησε τη συναλλαγή. Για να καταχωρηθεί η βαθμολογία του πρέπει να χαρακτηρίσει τη συναλλαγή ως αρνητική, θετική ή ουδέτερη και να αφήσει ένα μήνυμα αξιολόγησης.



Τέλος με την επιλογή Logout ο χρήστης μπορεί να εξέλθει από το σύστημα

7 Αρχιτεκτονική και περιγραφή συστήματος

Το σύστημα θα ακολουθεί την n-Tier Αρχιτεκτονική ώστε να επιτυγχάνεται η βέλτιστη λειτουργία και επεκτασιμότητα του συστήματος.

Συγκεκριμένα η αρχιτεκτονική του συστήματος θα ακολουθεί την 3-tier λογική. Με αυτό τον σχεδιασμό το σύστημα είναι στον μέγιστο βαθμό:

- Ευέλικτο σε μελλοντικές αλλαγές,
- Αρθρωτό και επεκτάσιμο,
- Άμεσο στην απόκριση του κατά την διάρκεια υψηλής ζήτησης,
- Συντηρήσιμο,
- Ασφαλές.

Με τη διαστρωμάτωση αυτή, επίσης, διαχωρίζεται η διεπαφή με τον χρήστη από τη λογική του συστήματος και αυτή με τη σειρά της από την αποθήκευση των δεδομένων.
Τα πλεονεκτήματα μια τέτοιας αρχιτεκτονικής περιλαμβάνουν:

- Ανεξαρτησία διεπαφής από την υπόλοιπη εφαρμογή.

- Το πληροφοριακό σύστημα γίνεται κατάλληλο για ετερογενή περιβάλλοντα (μπορούν να υπάρχουν διαφορετικές διεπαφές για διαφορετικούς χρήστες και πελάτες -π.χ. χρήστες που χρησιμοποιούν πρωτόκολλο WAP ή thin clients).

- Ελαχιστοποιούνται οι απαιτήσεις στον υπολογιστή του χρήστη, καθώς εκτελείται μόνο κώδικας διεπαφής χωρίς να απαιτούνται εγκαταστάσεις και εξαρτήσεις από βιβλιοθήκες λογισμικού τρίτων (απαιτείται μονάχα ένα φυλλομετρητής -Browser).

- Γίνεται καλύτερη διαχείριση πόρων των συστημάτων, καθώς μπορεί να καταμεριστεί και η πρόσβαση στα δεδομένα και η επιχειρησιακή λογική.

7.1 Ανάλυση Επιπέδων

Το πρώτο επίπεδο είναι υπεύθυνο για την εμφάνιση του περιβάλλοντος της Πύλης. Στο επίπεδο αυτό εδρεύουν οι rendering μηχανισμοί του συστήματος που είναι υπεύθυνοι για την άμεση και εξειδικευμένη εμφάνιση των αποτελεσμάτων που αιτούνται οι χρήστες των εφαρμογών (client tier).

Να σημειωθεί ότι δεδομένου της ανεξαρτησίας του τρόπου παραγωγής και rendering της πληροφορίας προς τον “έξω κόσμο” καθώς και της αρθρωτής αρχιτεκτονικής της Πύλης είναι πολύ εύκολη η μελλοντική επέκταση και υποστήριξη σε Thin Clients (wireless συσκευές, PDA, Κινητά τηλέφωνα, κλπ).

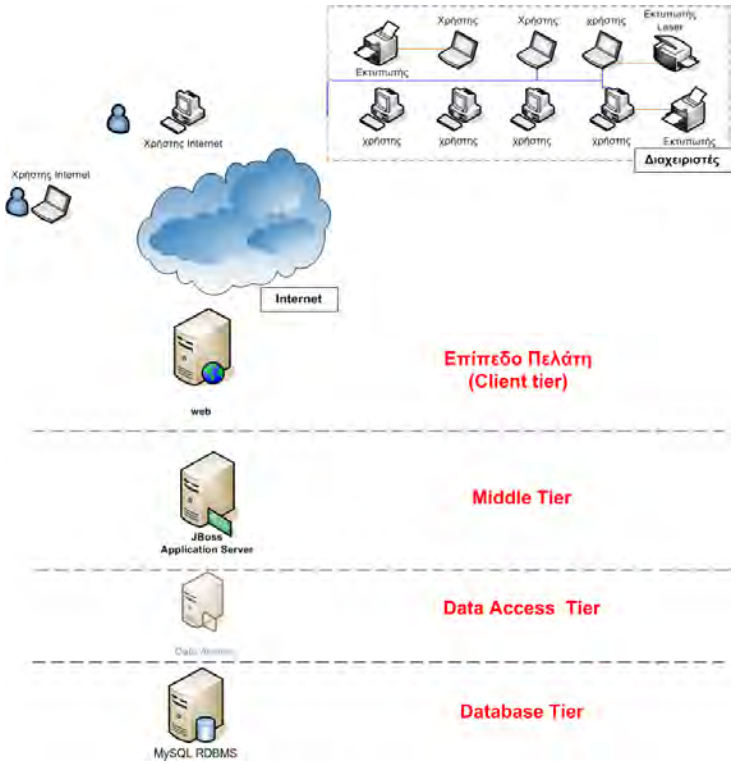
Στο Tier αυτό της προτεινόμενης αρχιτεκτονικής θα υλοποιηθούν οι διαδικασίες συλλογής δεδομένων εισόδου (από τους χρήστες), ελέγχου των δεδομένων αυτών, παράδοσης των επιχειρησιακών δεδομένων εισόδου στην επιχειρησιακή λογική, λήψης των δεδομένων εξόδου από την επιχειρησιακή λογική και εν τέλει δυναμικής παραγωγής των σελίδων του user Interface.

Στο επίπεδο αυτό θα υλοποιηθεί και η παραγωγή δυναμικού περιεχομένου που αφορά σε μη επιχειρησιακά δεδομένα και διαδικασίες, π.χ. περιεχομένου σχετικού με τη δυναμική πλοήγηση, τους καταλόγους και την προσωποποίηση του user Interface. Οι μηχανισμοί γένεσης του δυναμικού user interfaces θα βασιστούν στις δυνατότητες της επιλεγείσας πλατφόρμας Portal (JBoss Server). Οι υπηρεσίες και το λογισμικό του Web tier θα αξιοποιήσουν τις σχετικές J2EE τεχνολογίες (Java Server Pages, Java Servlets) και frameworks (Java Server Faces, Struts, webwork, κλπ).

Οι διεπαφές με τους χρήστες του πληροφοριακού συστήματος μπορούν να διατίθενται σε δυο διαφορετικές εκδόσεις:

Web-based διεπαφές

Η πρώτη κατηγορία διεπαφών μπορεί να εμφανιστεί στον χρήστη με την χρήση ενός φυλλομετρητή (browser) που θα συνδέεται στο πληροφοριακό σύστημα διαμέσου το δικτύου. Με αυτό τον τρόπο ο τελικός χρήστης θα μπορεί να έχει πρόσβαση στο σύστημα από οποιοδήποτε υπολογιστή ανεξάρτητα του λειτουργικού συστήματος που χρησιμοποιεί π.χ. Windows, Mac ή Linux. Η σύνδεση των χρηστών στο σύστημα θα είναι δυνατή τόσο από το τοπικό δίκτυο διαχειριστών (intranet), όσο και από το διαδίκτυο (Internet) χωρίς να απαιτείται η εγκατάσταση κάποιου ειδικού λογισμικού και χρησιμοποιώντας ως πρωτόκολλο επικοινωνίας το HTTP/HTTPS (HTTPS θα χρησιμοποιείται στις φάσεις που ανταλλάσσονται ευαίσθητου περιεχομένου δεδομένα που απαιτούν κρυπτογράφηση).



Η διαστρωμάτωση της Πύλης σε 3 επίπεδα

Middle Tier ή επίπεδο της επιχειρησιακής λογικής

Το δεύτερο επίπεδο περιέχει τον JBoss Application Server που είναι υπεύθυνος για την δημιουργία των αποτελεσμάτων (από τις αιτήσεις των χρηστών του συστήματος διαμέσου των φορμών και ιστοσελίδων της εφαρμογής αλλά και των διαφόρων SOAP μηνυμάτων που ανταλλάσσονται) καθώς και την εκτέλεση της Επιχειρησιακής Λογικής του πληροφοριακού συστήματος.

Σε αυτό το επίπεδο εδρεύει και η λογική που αφορά στο υποσύστημα διαχείρισης επικοινωνίας και συγκέντρωσης των δεδομένων. Μια σειρά από ειδικά και γενικά WebServices είναι υπεύθυνα για την διαχείριση της επικοινωνίας καθώς και για την συλλογή και διανομή των δεδομένων προς τον έξω κόσμο.

Το επίπεδο της επιχειρησιακής λογικής θα περιλάβει συλλογές από επιχειρησιακά συστατικά στοιχεία λογισμικού (π.χ. business components / EJB), κάθε μια από τις οποίες θα υλοποιήσει τις επιχειρησιακές διαδικασίες που περιλαμβάνονται στα υποσυστήματα που περιλαμβάνει το σύστημα

Επίπεδο δεδομένων (Database και Data Access tier)

Σε αυτό το επίπεδο βρίσκονται οι βάσεις δεδομένων (αν και αυτό στο σχήμα αυτό εμφανίζεται σε δυο tiers καθότι το Data Access αποτελεί ένα abstraction του Middle tier που έχει σκοπό να υλοποιεί το mapping της σχεσιακής βάσης δεδομένων για χρήση της από το Object Oriented περιβάλλον της Java με τον πιο φυσικό τρόπο).

Με βάση τις δυνατότητες του συστήματος βάσεων δεδομένων (Sybase Database) θα αναπτυχθεί λογισμικό για την εξασφάλιση της ακεραιότητας των δεδομένων αλλά και της ταχείας και αποδοτικής πρόσβασης σε αυτά.

Εδώ επίσης επιτυγχάνεται η σύνδεση μεταξύ του επιπέδου της Επιχειρησιακής Λογικής με τη Βάση Δεδομένων. Να σημειωθεί ότι κοινή πρακτική είναι το επίπεδο αυτό να μην διαχωρίζεται από το επίπεδο της επιχειρησιακής Λογικής. Στην παρούσα εφαρμογή λαμβάνοντας υπόψη την φύση των λειτουργιών που καλούνται τα υπό ανάπτυξη συστήματα να επιτελέσουν καθώς και τις ιδιαιτερότητες που εμφανίζονται, ο σχεδιασμός περιλαμβάνει ξεχωριστή υλοποίηση για το επίπεδο της Πρόσβασης στη Βάση Δεδομένων (Data Access Tier) και της Επιχειρησιακής Λογικής.

Με τον τρόπο αυτό επιτυγχάνουμε:

Κλιμάκωση – Από τη στιγμή που η επιχειρησιακή λογική αναπτύσσεται ξεχωριστά από την πρόσβαση στη Βάση δεδομένων, έχουμε την δυνατότητα να αναβαθμίζουμε και να τροποποιούμε τον κώδικα του επιπέδου της Πρόσβασης στη Βάση δεδομένων χωρίς να επεμβαίνουμε καθόλου στο επίπεδο της επιχειρησιακής λογικής.

Διαθεσιμότητα – Ο διαχωρισμός αυτός μας επιτρέπει τον καλύτερο έλεγχο της εφαρμογής, μιας και σε περίπτωση που προκύψει κάποιο πρόβλημα, μπορούμε να απομονώσουμε το κομμάτι του λογισμικού που έχει πρόβλημα, και η υπόλοιπη εφαρμογή να μείνει ανεπηρέαστη.

Συντηρησιμότητα – Όπως αναφέρεται και στην κλιμάκωση, αν χρειαστεί να γίνει μία αλλαγή στο επίπεδο της Πρόσβασης στη Βάση Δεδομένων, τότε το επίπεδο της επιχειρησιακής λογικής θα μείνει ανεπηρέαστο.

Ουσιαστικά αυτό το Tier που περιλαμβάνει τον Database Server. Στο επίπεδο αυτό επιτυγχάνεται η φυσική καταχώρηση των δεδομένων στη Βάση και η ανάκτηση των δεδομένων αυτών. Η καρδιά του συστήματος του επιπέδου αυτού είναι η Sybase.

7.2 Αρχιτεκτονική και Εφαρμογές

Η αρχιτεκτονική θα επιτρέψει την ανεξάρτητη υλοποίηση, εγκατάσταση ή/και παραμετροποίηση των διαφορετικών εφαρμογών-υποσυστημάτων της Πύλης. Η αρχιτεκτονική που έχει περιγραφεί επιβάλλει φυσικά την κοινή χρήση κάποιων βιβλιοθηκών (jar) με χαρακτηριστικότερα παραδείγματα τις κεντρικές-οριζόντιες υπηρεσίες, και τις βιβλιοθήκες πρόσβασης στα δεδομένα (μια και οι εφαρμογές είναι πιθανό να χρησιμοποιούν κοινά δεδομένα).

Οι εφαρμογές στο βαθμό που θα απαιτηθεί θα είναι δυνατό να επικοινωνούν μεταξύ τους μέσω μηχανισμών επικοινωνίας-ολοκλήρωσης κατανεμημένων συστημάτων που παρέχει η προδιαγραφή J2EE1.4 και η πλατφόρμα (AS), και ειδικότερα διαμέσου:

J2EE Web Services

JMS (Java Messaging Service)

EJB

Η επιλογή του κατάλληλου μηχανισμού θα λάβει σε κάθε περίπτωση υπ' όψη τις απαιτήσεις επιδόσεων και ασφάλειας για τη ζητούμενη ολοκλήρωση, καθώς και το βαθμό στον οποίο απαιτείται περισσότερο ή λιγότερο στενή διασύνδεση των συστημάτων (loose vs. tight coupling).

Σημειώνεται ότι οι πλατφόρμες J2EE / JBoss AS είναι ανοιχτές, μια και υλοποιούν τις προδιαγραφές J2EE 1.4 και W3C Web Services. Κατά συνέπεια είναι δυνατή στα πρότυπα των παραπάνω αρχιτεκτονικών προσεγγίσεων η διασύνδεση της Πύλης με άλλα συστήματα και ομάδες χρηστών, όταν και εφ' όσον αυτό απαιτηθεί.

8 Υλοποίηση συστήματος

Για την υλοποίηση του συστήματος χρησιμοποιήθηκε η πλατφόρμα του Eclipse σε συνδυασμό με την έκδοση jdk6 της java.

Το Eclipse είναι μία ευρύτατα διαδεδομένη πλατφόρμα ανάπτυξης εφαρμογών η οποία διατίθεται στην default έκδοσή της για Java developers. Ισχυρό εργαλείο του Eclipse είναι η δυνατότητα προσθήκης plugins ώστε να υποστηρίζει δεκάδες επιπλέον γλώσσες και δυνατότητες.

Για το mapping της βάσης σε Java χρησιμοποιήθηκε το εργαλείο Hibernate το οποίο προστίθεται ως plug-in στο Eclipse.

Το Hibernate Framework είναι λογισμικό ανοιχτού κώδικα (ελεύθερο λογισμικό) που σκοπό έχει να συνδέσει τα αντικείμενα που δημιουργούνται σε μία αντικειμενοστραφή γλώσσα προγραμματισμού (Java) με τους πίνακες μιας σχεσιακής βάσης δεδομένων. Η σύνδεση αυτή επιτυγχάνεται με την χρήση επιπρόσθετης πληροφορίας (metadata) που τοποθετείται κατάλληλα (μαζί με τον κώδικα Java ή σε ξεχωριστά xml αρχεία) και περιγράφει την αντιστοιχία μεταξύ των αντικειμένων και της βάσης δεδομένων. Γενικά το Hibernate προσφέρει την αυτόματη μετατροπή της μίας μορφής (αντικείμενα) στην άλλη (σχεσιακή βάση δεδομένων).

Το Hibernate συγκαταλέγεται στην κατηγορία του ORM (*object/relational mapping*) λογισμικού. Το ORM λογισμικό στοχεύει στην δημιουργία μιας διεπαφής (*interface*) μεταξύ των διαδεδομένων σχεσιακών βάσεων δεδομένων και του αντικειμενοστραφούς προγραμματισμού. Με απλά λόγια, προσφέρει την χρησιμοποίηση μιας σχεσιακής βάσης δεδομένων ως αντικειμενοστραφή. Για να το επιτύχει αυτό δημιουργεί αντιστοιχίες μεταξύ των εννοιών του αντικειμενοστραφούς προγραμματισμού (συσχετίσεις, κληρονομικότητα, πολυμορφισμός) - που δεν υπάρχουν σε μία σχεσιακή βάση δεδομένων - και των πινάκων και σχέσεων μεταξύ των πινάκων μιας σχεσιακής βάσης. Με αυτό τον τρόπο ο προγραμματιστής *βλέπει* τελικά μία αντικειμενοστραφή βάση δεδομένων, παρόλο που στην ουσία χρησιμοποιεί μια σχεσιακή. Έτσι ο προγραμματιστής χρησιμοποιεί τα αντικείμενα της συγκεκριμένης εφαρμογής, τα τροποποιεί σχετικά με τη λογική της εφαρμογής που αναπτύσσει και τα αποθηκεύει (τροποποιεί, διαγράφει και αναζητά) στην βάση *ως αντικείμενα*, σκεπτόμενος δηλαδή με αντικειμενοστραφείς έννοιες και όχι με βάση το σχήμα της σχεσιακής βάσης δεδομένων. Σε αυτό το σημείο είναι το Hibernate που, γνωρίζοντας την αντιστοιχία μεταξύ βάσης και λογικής της εφαρμογής, αναλαμβάνει να κατασκευάσει την κατάλληλη εντολή της SQL η οποία και στέλνεται τελικά στην βάση δεδομένων. Έπειτα, τα αποτελέσματα που επιστρέφει η βάση το Hibernate τα επιστρέφει στον προγραμματιστή ως αντικείμενα της εφαρμογής. Είναι δηλαδή ένα ενδιάμεσο επίπεδο μεταξύ εφαρμογής και βάσης δεδομένων.

Το Hibernate προσφέρει τα παρακάτω στον προγραμματιστή:

- Παραγωγικότητα:

  Στην ανάπτυξη λογισμικού ένα μεγάλο μέρος της προγραμματιστικής προσπάθειας αφιερώνεται στην διεπαφή της εφαρμογής με τη βάση δεδομένων. Το Hibernate αυτοματοποιώντας τις βασικές λειτουργίες (CRUD – Create Read Update Delete) επιτρέπει αρχικά στον προγραμματιστή να επικεντρώνει την προσπάθειά του στη λογική της εφαρμογής (business logic). Επίσης, υπάρχει η δυνατότητα να ακολουθηθούν δύο στρατηγικές ανάπτυξης λογισμικού: είτε αρχίζοντας από το μοντέλο δεδομένων είτε από τη βάση δεδομένων. Αυτό μειώνει σε μεγάλο βαθμό το χρόνο ανάπτυξης.

- Συντηρησιμότητα:

  Με τη χρήση του Hibernate γράφονται σημαντικά λιγότερες γραμμές κώδικα και ο κώδικας είναι πιο κατανοητός και καλογραμμένος. Αυτό κάνει την συντήρηση της εφαρμογής ευκολότερη.

- Ανεξαρτησία από τη βάση δεδομένων:

  Με τη συμβατότητα του Hibernate με διαφορετικές βάσεις δεδομένων και τη δυνατότητα σύνδεσής του με τη βάση μέσω δηλώσεων οριζομένων σε ειδικό αρχείο η αναπτυσσόμενη εφαρμογή μπορεί με ελάχιστες τροποποιήσεις να χρησιμοποιηθεί με βάσεις δεδομένων διαφορετικών κατασκευαστών. Το γεγονός αυτό στερεί μεν από το Hibernate την εκμετάλλευση των ιδιαίτερων χαρακτηριστικών της χρησιμοποιούμενης βάσης, όμως, και σε αυτή την περίπτωση, δίνεται η δυνατότητα χρήσης πηγαίας SQL μέσα στο Hibernate που εκμεταλλεύεται τα ιδιαίτερα αυτά χαρακτηριστικά. Αυτό βέβαια μειώνει την ανεξαρτησία του Hibernate.

Παράρτημα Α

Υλοποίηση της βάσης δεδομένων

Ο κώδικας MySQL που δημιουργεί την βάση δεδομένων του συστήματός μας στον MySQL Server είναι:

```
/*==============================================================*/
/* DBMS name:      MySQL 5.0                                    */
/*==============================================================*/


drop table if exists exchange;

drop table if exists good;

drop table if exists goodexchange;

drop table if exists grade;

drop table if exists member;

drop table if exists messageBuffer;

drop table if exists parent;

drop table if exists photo;

drop table if exists user;

drop table if exists usergood;

drop table if exists usersinexchange;

/*==============================================================*/
/* Table: exchange                                              */
/*==============================================================*/
create table exchange
(
   exc_id               int not null,
   exc_type             varchar(100),
   exc_active           int,
   exc_date_create      datetime,
   primary key (exc_id)
);
```

```
/*==============================================================*/
/* Table: good                                                  */
/*==============================================================*/
create table good
(
  good_id           int not null AUTO_INCREMENT,
  good_description    varchar(500),
  good_quantity       int,
  good_ask          bool,
  good_category       bool,
  good_date_create    datetime,
  good_date_exp       datetime,
  primary key (good_id)
);

/*==============================================================*/
/* Table: goodexchange                                          */
/*==============================================================*/
create table goodexchange
(
  good_id           int not null,
  exc_id            int not null,
  quantity          int,
  primary key (good_id, exc_id)
);

/*==============================================================*/
/* Table: grade                                                 */
/*==============================================================*/
create table grade
(
  grade_id          int not null AUTO_INCREMENT,
  exc_id            int,
  user_id           int,
  num               int,
  comment           varchar(1000),
  com_date          datetime,
  primary key (grade_id)
);

/*==============================================================*/
/* Table: member                                                */
/*==============================================================*/
create table member
(
  use_user_id       int not null,
  user_id           int not null,
```

```
  admin             bool not null,
  primary key (use_user_id, user_id)
);

/*==============================================================*/
/* Table: messageBuffer                                         */
/*==============================================================*/
create table messageBuffer
(
  mes_id            int not null AUTO_INCREMENT,
  mes_from            varchar(200),
  mes_to            varchar(200),
  mes_cc            varchar(200),
  mes_bcc            varchar(200),
  mes_subject         varchar(100),
  mes_text           varchar(2000),
  mes_int           int,
  primary key (mes_id)
);

/*==============================================================*/
/* Table: parent                                                */
/*==============================================================*/
create table parent
(
  good_id           int not null,
  goo_good_id         int not null,
  primary key (good_id, goo_good_id)
);

/*==============================================================*/
/* Table: photo                                                 */
/*==============================================================*/
create table photo
(
  photo_id           int not null AUTO_INCREMENT,
  good_id           int,
  photo_path          varchar(255),
  photo_name          varchar(200),
  primary key (photo_id)
);
```

```
/*==============================================================*/
/* Table: user                                                  */
/*==============================================================*/
create table user
(
  user_id           int not null AUTO_INCREMENT,
  user_name         varchar(100),
  user_surname      varchar(100),
  user_description  varchar(500),
  user_username     varchar(100),
  user_password     varchar(100),
  user_address      varchar(100),
  user_city         varchar(100),
  user_country      varchar(100),
  user_zip          int,
  user_community    int,
  user_grade        int,
  user_photo_path   varchar(255),
  user_date_created datetime,
  user_date_modified datetime,
  user_photo_name   varchar(200),
  primary key (user_id)
);

/*==============================================================*/
/* Table: usergood                                              */
/*==============================================================*/
create table usergood
(
  user_id           int not null,
  good_id           int not null,
  primary key (user_id, good_id)
);

/*==============================================================*/
/* Table: usersinexchange                                       */
/*==============================================================*/
create table usersinexchange
(
  user_id           int not null,
  exc_id            int not null,
  primary key (user_id, exc_id)
);

alter table goodexchange add constraint FK_goodexchange foreign key (good_id)
    references good (good_id) on delete restrict on update restrict;
```

```
alter table goodexchange add constraint FK_goodexchange2 foreign key (exc_id)
    references exchange (exc_id) on delete restrict on update restrict;

alter table grade add constraint FK_exchangegrade foreign key (exc_id)
    references exchange (exc_id) on delete restrict on update restrict;

alter table grade add constraint FK_usergrade foreign key (user_id)
    references user (user_id) on delete restrict on update restrict;

alter table member add constraint FK_member foreign key (use_user_id)
    references user (user_id) on delete restrict on update restrict;

alter table member add constraint FK_member2 foreign key (user_id)
    references user (user_id) on delete restrict on update restrict;

alter table parent add constraint FK_parent foreign key (good_id)
    references good (good_id) on delete restrict on update restrict;

alter table parent add constraint FK_parent2 foreign key (goo_good_id)
    references good (good_id) on delete restrict on update restrict;

alter table photo add constraint FK_goodphoto foreign key (good_id)
    references good (good_id) on delete restrict on update restrict;

alter table usergood add constraint FK_usergood foreign key (user_id)
    references user (user_id) on delete restrict on update restrict;

alter table usergood add constraint FK_usergood2 foreign key (good_id)
    references good (good_id) on delete restrict on update restrict;

alter table usersinexchange add constraint FK_usersinexchange foreign key (user_id)
    references user (user_id) on delete restrict on update restrict;

alter table usersinexchange add constraint FK_usersinexchange2 foreign key (exc_id)
    references exchange (exc_id) on delete restrict on update restrict;
```

Παράρτημα Β

Δημιουργία των αντικειμένων στην εφαρμογή

Για κάθε αντικείμενο της εφαρμογής δημιουργείται μία κλάση για τη δημιουργία των στιγμιοτύπων τους.
Για το αντικείμενο **User** δημιουργούμε την κλάση User.java

```
import java.util.Date;
import java.util.HashSet;
import java.util.Set;

public class User implements java.io.Serializable {

	private Integer userId;
	private String userName;
	private String userSurname;
	private String userDescription;
	private String userUsername;
	private String userPassword;
	private String userAddress;
	private String userCity;
	private String userCountry;
	private Integer userZip;
	private Integer userCommunity;
	private Integer userGrade;
	private String userPhotoPath;
	private Date userDateCreated;
	private Date userDateModified;
	private String userPhotoName;
	private Set<Good> goods = new HashSet<Good>(0);
	private Set<Grade> grades = new HashSet<Grade>(0);
	private Set<Exchange> exchanges = new HashSet<Exchange>(0);
	private Set<Member> membersForUserId = new HashSet<Member>(0);
	private Set<Member> membersForUseUserId = new HashSet<Member>(0);

	public User() {
	}

	public User(String userName, String userSurname, String userDescription,
String userUsername, String userPassword, String userAddress, String userCity, String
userCountry, Integer userZip, Integer userCommunity, Integer userGrade, String
userPhotoPath, Date userDateCreated, Date userDateModified, String userPhotoName,
Set<Good> goods, Set<Grade> grades, Set<Exchange> exchanges, Set<Member>
membersForUserId,Set<Member> membersForUseUserId) {
		this.userName = userName;
		this.userSurname = userSurname;
```

```
        this.userDescription = userDescription;
        this.userUsername = userUsername;
        this.userPassword = userPassword;
        this.userAddress = userAddress;
        this.userCity = userCity;
        this.userCountry = userCountry;
        this.userZip = userZip;
        this.userCommunity = userCommunity;
        this.userGrade = userGrade;
        this.userPhotoPath = userPhotoPath;
        this.userDateCreated = userDateCreated;
        this.userDateModified = userDateModified;
        this.userPhotoName = userPhotoName;
        this.goods = goods;
        this.grades = grades;
        this.exchanges = exchanges;
        this.membersForUserId = membersForUserId;
        this.membersForUseUserId = membersForUseUserId;
}

public Integer getUserId() {
        return this.userId;
}

public void setUserId(Integer userId) {
        this.userId = userId;
}

public String getUserName() {
        return this.userName;
}

public void setUserName(String userName) {
        this.userName = userName;
}

public String getUserSurname() {
        return this.userSurname;
}

public void setUserSurname(String userSurname) {
        this.userSurname = userSurname;
}

public String getUserDescription() {
        return this.userDescription;
}
```

```
public void setUserDescription(String userDescription) {
        this.userDescription = userDescription;
}

public String getUserUsername() {
        return this.userUsername;
}

public void setUserUsername(String userUsername) {
        this.userUsername = userUsername;
}

public String getUserPassword() {
        return this.userPassword;
}

public void setUserPassword(String userPassword) {
        this.userPassword = userPassword;
}

public String getUserAddress() {
        return this.userAddress;
}

public void setUserAddress(String userAddress) {
        this.userAddress = userAddress;
}

public String getUserCity() {
        return this.userCity;
}

public void setUserCity(String userCity) {
        this.userCity = userCity;
}

public String getUserCountry() {
        return this.userCountry;
}

public void setUserCountry(String userCountry) {
        this.userCountry = userCountry;
}

public Integer getUserZip() {
        return this.userZip;
}
```

```
public void setUserZip(Integer userZip) {
	this.userZip = userZip;
}

public Integer getUserCommunity() {
	return this.userCommunity;
}

public void setUserCommunity(Integer userCommunity) {
	this.userCommunity = userCommunity;
}

public Integer getUserGrade() {
	return this.userGrade;
}

public void setUserGrade(Integer userGrade) {
	this.userGrade = userGrade;
}

public String getUserPhotoPath() {
	return this.userPhotoPath;
}

public void setUserPhotoPath(String userPhotoPath) {
	this.userPhotoPath = userPhotoPath;
}

public Date getUserDateCreated() {
	return this.userDateCreated;
}

public void setUserDateCreated(Date userDateCreated) {
	this.userDateCreated = userDateCreated;
}

public Date getUserDateModified() {
	return this.userDateModified;
}

public void setUserDateModified(Date userDateModified) {
	this.userDateModified = userDateModified;
}

public String getUserPhotoName() {
	return this.userPhotoName;
```

```
	}

	public void setUserPhotoName(String userPhotoName) {
		this.userPhotoName = userPhotoName;
	}

	public Set<Good> getGoods() {
		return this.goods;
	}

	public void setGoods(Set<Good> goods) {
		this.goods = goods;
	}

	public Set<Grade> getGrades() {
		return this.grades;
	}

	public void setGrades(Set<Grade> grades) {
		this.grades = grades;
	}

	public Set<Exchange> getExchanges() {
		return this.exchanges;
	}

	public void setExchanges(Set<Exchange> exchanges) {
		this.exchanges = exchanges;
	}

	public Set<Member> getMembersForUserId() {
		return this.membersForUserId;
	}

	public void setMembersForUserId(Set<Member> membersForUserId) {
		this.membersForUserId = membersForUserId;
	}

	public Set<Member> getMembersForUseUserId() {
		return this.membersForUseUserId;
	}

	public void setMembersForUseUserId(Set<Member> membersForUseUserId) {
		this.membersForUseUserId = membersForUseUserId;
	}

}
```

Για το αντικείμενο **Good** δημιουργούμε την κλάση Good.java

```
import java.util.Date;
import java.util.HashSet;
import java.util.Set;
import java.util.LinkedList;
import java.util.List;

public class Good implements java.io.Serializable {

	private Integer goodId;
	private String goodDescription;
	private Integer goodQuantity;
	private Boolean goodAsk;
	private Boolean goodCategory;
	private Date goodDateCreate;
	private Date goodDateExp;
	private Set<Good> goodsForGoodId = new HashSet<Good>(0);
	private Set<Goodexchange> goodexchanges = new HashSet<Goodexchange>(0);
	private Set<User> users = new HashSet<User>(0);
	private Set<Good> goodsForGooGoodId = new HashSet<Good>(0);
	private Set<Photo> photos = new HashSet<Photo>(0);

	public Good() {
	}

	public Good(String goodDescription, Integer goodQuantity, Boolean goodAsk,
Boolean goodCategory, Date goodDateCreate, Date goodDateExp, Set<Good>
goodsForGoodId, Set<Goodexchange> goodexchanges, Set<User> users, Set<Good>
goodsForGooGoodId, Set<Photo> photos) {
		this.goodDescription = goodDescription;
		this.goodQuantity = goodQuantity;
		this.goodAsk = goodAsk;
		this.goodCategory = goodCategory;
		this.goodDateCreate = goodDateCreate;
		this.goodDateExp = goodDateExp;
		this.goodsForGoodId = goodsForGoodId;
		this.goodexchanges = goodexchanges;
		this.users = users;
		this.goodsForGooGoodId = goodsForGooGoodId;
		this.photos = photos;
	}

	public Integer getGoodId() {
		return this.goodId;
	}
```

```
public void setGoodId(Integer goodId) {
        this.goodId = goodId;
}

public String getGoodDescription() {
        return this.goodDescription;
}

public void setGoodDescription(String goodDescription) {
        this.goodDescription = goodDescription;
}

public Integer getGoodQuantity() {
        return this.goodQuantity;
}

public void setGoodQuantity(Integer goodQuantity) {
        this.goodQuantity = goodQuantity;
}

public Boolean getGoodAsk() {
        return this.goodAsk;
}

public void setGoodAsk(Boolean goodAsk) {
        this.goodAsk = goodAsk;
}

public Boolean getGoodCategory() {
        return this.goodCategory;
}

public void setGoodCategory(Boolean goodCategory) {
        this.goodCategory = goodCategory;
}

public Date getGoodDateCreate() {
        return this.goodDateCreate;
}

public void setGoodDateCreate(Date goodDateCreate) {
        this.goodDateCreate = goodDateCreate;
}

public Date getGoodDateExp() {
        return this.goodDateExp;   }
```

```
	public void setGoodDateExp(Date goodDateExp) {
		this.goodDateExp = goodDateExp;
	}

	public Set<Good> getGoodsForGoodId() {
		return this.goodsForGoodId;
	}

	public void setGoodsForGoodId(Set<Good> goodsForGoodId) {
		this.goodsForGoodId = goodsForGoodId;
	}

	public Set<Goodexchange> getGoodexchanges() {
		return this.goodexchanges;
	}

	public void setGoodexchanges(Set<Goodexchange> goodexchanges) {
		this.goodexchanges = goodexchanges;
	}

	public Set<User> getUsers() {
		return this.users;
	}

	public void setUsers(Set<User> users) {
		this.users = users;
	}


	public Set<Good> getGoodsForGooGoodId() {
		return this.goodsForGooGoodId;
	}

	public void setGoodsForGooGoodId(Set<Good> goodsForGooGoodId) {
		this.goodsForGooGoodId = goodsForGooGoodId;
	}

	public Set<Photo> getPhotos() {
		return this.photos;
	}

	public void setPhotos(Set<Photo> photos) {
		this.photos = photos;
	}

}
```

Για το αντικείμενο **Exchange** δημιουργούμε την κλάση Exchange.java

```
import java.util.Date;
import java.util.HashSet;
import java.util.Set;
import java.util.LinkedList;
import java.util.List;

public class Exchange implements java.io.Serializable {

	private int excId;
	private String excType;
	private Integer excActive;
	private Date excDateCreate;
	private Set<Grade> grades = new HashSet<Grade>(0);
	private Set<Goodexchange> goodexchanges = new HashSet<Goodexchange>(0);
	private Set<User> users = new HashSet<User>(0);

	public Exchange() {
	}

	public Exchange(int excId) {
		this.excId = excId;
	}

	public Exchange(int excId, String excType, Integer excActive, Date
excDateCreate, Set<Grade> grades, Set<Goodexchange> goodexchanges, Set<User>
users) {
		this.excId = excId;
		this.excType = excType;
		this.excActive = excActive;
		this.excDateCreate = excDateCreate;
		this.grades = grades;
		this.goodexchanges = goodexchanges;
		this.users = users;
	}

	public int getExcId() {
		return this.excId;
	}

	public void setExcId(int excId) {
		this.excId = excId;
	}

	public String getExcType() {
		return this.excType;
	}
```

```
	public void setExcType(String excType) {
		this.excType = excType;
	}

	public Integer getExcActive() {
		return this.excActive;
	}

	public void setExcActive(Integer excActive) {
		this.excActive = excActive;
	}

	public Date getExcDateCreate() {
		return this.excDateCreate;
	}

	public void setExcDateCreate(Date excDateCreate) {
		this.excDateCreate = excDateCreate;
	}

	public Set<Grade> getGrades() {
		return this.grades;
	}

	public void setGrades(Set<Grade> grades) {
		this.grades = grades;
	}

	public Set<Goodexchange> getGoodexchanges() {
		return this.goodexchanges;
	}

	public void setGoodexchanges(Set<Goodexchange> goodexchanges) {
		this.goodexchanges = goodexchanges;
	}

	public Set<User> getUsers() {
		return this.users;
	}

	public void setUsers(Set<User> users) {
		this.users = users;
	}


}
```

Για τη δημιουργία των στιγμιοτύπων του αντικειμένου **Goodexchange** δημιουργούμε την κλάση Goodexchange.java

```
public class Goodexchange implements java.io.Serializable {

	private GoodexchangeId id;
	private Exchange exchange;
	private Good good;
	private Integer quantity;
	private Integer propose;

	public Goodexchange() {
	}

	public Goodexchange(GoodexchangeId id, Exchange exchange, Good good) {
		this.id = id;
		this.exchange = exchange;
		this.good = good;
	}

	public Goodexchange(GoodexchangeId id, Exchange exchange, Good good,
			Integer quantity, Integer propose) {
		this.id = id;
		this.exchange = exchange;
		this.good = good;
		this.quantity = quantity;
		this.propose = propose;
	}

	public GoodexchangeId getId() {
		return this.id;
	}

	public void setId(GoodexchangeId id) {
		this.id = id;
	}

	public Exchange getExchange() {
		return this.exchange;
	}

	public void setExchange(Exchange exchange) {
		this.exchange = exchange;
	}
```

```
	public Good getGood() {
		return this.good;
	}

	public void setGood(Good good) {
		this.good = good;
	}

	public Integer getQuantity() {
		return this.quantity;
	}

	public void setQuantity(Integer quantity) {
		this.quantity = quantity;
	}

	public Integer getPropose() {
		return this.propose;
	}

	public void setPropose(Integer propose) {
		this.propose = propose;
	}

}
```

Για το αντικείμενο **Grade** έχουμε δημιουργήσει την κλάση Grade.java

```
import java.util.Date;

public class Grade implements java.io.Serializable {

	private Integer gradeId;
	private Exchange exchange;
	private User user;
	private Integer num;
	private String comment;
	private Date comDate;

	public Grade() {
	}

	public Grade(Exchange exchange, User user, Integer num, String comment,
			Date comDate) {
		this.exchange = exchange;
		this.user = user;
		this.num = num;
```

```
        this.comment = comment;
        this.comDate = comDate;
}

public Integer getGradeId() {
        return this.gradeId;
}

public void setGradeId(Integer gradeId) {
        this.gradeId = gradeId;
}

public Exchange getExchange() {
        return this.exchange;
}

public void setExchange(Exchange exchange) {
        this.exchange = exchange;
}

public User getUser() {
        return this.user;
}

public void setUser(User user) {
        this.user = user;
}

public Integer getNum() {
        return this.num;
}

public void setNum(Integer num) {
        this.num = num;
}

public String getComment() {
        return this.comment;
}

public void setComment(String comment) {
        this.comment = comment;
}

public Date getComDate() {
        return this.comDate;
}
```

```
	public void setComDate(Date comDate) {
		this.comDate = comDate;
	}

}
```

Για το αντικείμενο **Member** δημιουργήθηκε η κλάση Member.java

```
package gr.exchange.mappings;

public class Member implements java.io.Serializable {

	private MemberId id;
	private User userByUserId;
	private User userByUseUserId;
	private boolean admin;

	public Member() {
	}

	public Member(MemberId id, User userByUserId, User userByUseUserId,
			boolean admin) {
		this.id = id;
		this.userByUserId = userByUserId;
		this.userByUseUserId = userByUseUserId;
		this.admin = admin;
	}

	public MemberId getId() {
		return this.id;
	}

	public void setId(MemberId id) {
		this.id = id;
	}

	public User getUserByUserId() {
		return this.userByUserId;
	}

	public void setUserByUserId(User userByUserId) {
		this.userByUserId = userByUserId;
	}

	public User getUserByUseUserId() {
		return this.userByUseUserId;
```

```
	}

	public void setUserByUseUserId(User userByUseUserId) {
		this.userByUseUserId = userByUseUserId;
	}

	public boolean isAdmin() {
		return this.admin;
	}

	public void setAdmin(boolean admin) {
		this.admin = admin;
	}

}
```

Για τα αντικείμενα τύπου **MessageBuffer** έχει δημιουργηθεί η κλάση
MessageBuffer.java

```
public class Messagebuffer implements java.io.Serializable {

	private Integer mesId;
	private String mesFrom;
	private String mesTo;
	private String mesCc;
	private String mesBcc;
	private String mesSubject;
	private String mesText;
	private Integer mesInt;

	public Messagebuffer() {
	}

	public Messagebuffer(String mesFrom, String mesTo, String mesCc, String
mesBcc, String mesSubject, String mesText, Integer mesInt) {
		this.mesFrom = mesFrom;
		this.mesTo = mesTo;
		this.mesCc = mesCc;
		this.mesBcc = mesBcc;
		this.mesSubject = mesSubject;
		this.mesText = mesText;
		this.mesInt = mesInt;
	}

	public Integer getMesId() {
		return this.mesId;
	}
```

```
public void setMesId(Integer mesId) {
        this.mesId = mesId;
}

public String getMesFrom() {
        return this.mesFrom;
}

public void setMesFrom(String mesFrom) {
        this.mesFrom = mesFrom;
}

public String getMesTo() {
        return this.mesTo;
}

public void setMesTo(String mesTo) {
        this.mesTo = mesTo;
}

public String getMesCc() {
        return this.mesCc;
}

public void setMesCc(String mesCc) {
        this.mesCc = mesCc;
}

public String getMesBcc() {
        return this.mesBcc;
}

public void setMesBcc(String mesBcc) {
        this.mesBcc = mesBcc;
}

public String getMesSubject() {
        return this.mesSubject;
}

public void setMesSubject(String mesSubject) {
        this.mesSubject = mesSubject;
}

public String getMesText() {
        return this.mesText;
}
```

```
	public void setMesText(String mesText) {
		this.mesText = mesText;
	}

	public Integer getMesInt() {
		return this.mesInt;
	}

	public void setMesInt(Integer mesInt) {
		this.mesInt = mesInt;
	}

}
```

Και τέλος τα στιγμιότυπα του αντικειμένου **Photo** δημιουργούνται από την κλάση photo.java

```
public class Photo implements java.io.Serializable {

	private Integer photoId;
	private Good good;
	private String photoPath;
	private String photoName;

	public Photo() {
	}

	public Photo(Good good, String photoPath, String photoName) {
		this.good = good;
		this.photoPath = photoPath;
		this.photoName = photoName;
	}

	public Integer getPhotoId() {
		return this.photoId;
	}

	public void setPhotoId(Integer photoId) {
		this.photoId = photoId;
	}

	public Good getGood() {
		return this.good;
	}

	public void setGood(Good good) {
```

```
            this.good = good;
        }

        public String getPhotoPath() {
            return this.photoPath;
        }

        public void setPhotoPath(String photoPath) {
            this.photoPath = photoPath;
        }

        public String getPhotoName() {
            return this.photoName;
        }

        public void setPhotoName(String photoName) {
            this.photoName = photoName;
        }
}
```

Παράρτημα Γ

Σύνδεση αντικειμένων της εφαρμογής με τη βάση δεδομένων

Για κάθε πίνακα της σχεσιακής βάσης δεδομένων παράγεται ένα xml αρχείο για τη σύνδεση αυτού με τον αντίστοιχο αντικείμενο της εφαρμογής.

Για τον πίνακα **User** έχουμε το παρακάτω xml αρχείο

```
<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD
3.0//EN"
"http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">
<hibernate-mapping>
  <class name="gr.exchange.mappings.User" table="user" catalog="diplwma">
    <id name="userId" type="java.lang.Integer">
      <column name="user_id" />
      <generator class="identity" />
    </id>
    <property name="userName" type="string">
      <column name="user_name" length="100" />
    </property>
    <property name="userSurname" type="string">
      <column name="user_surname" length="100" />
    </property>
    <property name="userDescription" type="string">
      <column name="user_description" length="500" />
    </property>
    <property name="userUsername" type="string">
      <column name="user_username" length="100" />
    </property>
    <property name="userPassword" type="string">
      <column name="user_password" length="100" />
    </property>
    <property name="userAddress" type="string">
      <column name="user_address" length="100" />
    </property>
    <property name="userCity" type="string">
      <column name="user_city" length="100" />
    </property>
    <property name="userCountry" type="string">
      <column name="user_country" length="100" />
    </property>
    <property name="userZip" type="java.lang.Integer">
      <column name="user_zip" />
    </property>
```

```
<property name="userCommunity" type="java.lang.Integer">
  <column name="user_community" />
</property>
<property name="userGrade" type="java.lang.Integer">
  <column name="user_grade" />
</property>
<property name="userPhotoPath" type="string">
  <column name="user_photo_path" />
</property>
<property name="userDateCreated" type="timestamp">
  <column name="user_date_created" length="0" />
</property>
<property name="userDateModified" type="timestamp">
  <column name="user_date_modified" length="0" />
</property>
<property name="userPhotoName" type="string">
  <column name="user_photo_name" length="200" />
</property>
<set name="goods" inverse="false" table="usergood" lazy="false">
  <key>
    <column name="user_id" not-null="true" />
  </key>
  <many-to-many entity-name="gr.exchange.mappings.Good">
    <column name="good_id" not-null="true" />
  </many-to-many>
</set>
<set name="grades" inverse="true">
  <key>
    <column name="user_id" />
  </key>
  <one-to-many class="gr.exchange.mappings.Grade" />
</set>
<set name="exchanges" inverse="false" table="usersinexchange" lazy="false">
  <key>
    <column name="user_id" not-null="true" />
  </key>
  <many-to-many entity-name="gr.exchange.mappings.Exchange">
    <column name="exc_id" not-null="true" />
  </many-to-many>
</set>
<set name="membersForUserId" inverse="false" lazy="false">
  <key>
    <column name="user_id" not-null="true" />
  </key>
  <one-to-many class="gr.exchange.mappings.Member" />
</set>
<set name="membersForUseUserId" inverse="false" lazy="false">
```

```
        <key>
          <column name="use_user_id" not-null="true" />
        </key>
        <one-to-many class="gr.exchange.mappings.Member" />
      </set>
    </class>
</hibernate-mapping>
```

Για τον πίνακα **Good** έχουμε το παρακάτω xml αρχείο:

```
<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD
3.0//EN"
"http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">
<hibernate-mapping>
   <class name="gr.exchange.mappings.Good" table="good" catalog="diplwma">
      <id name="goodId" type="java.lang.Integer">
         <column name="good_id" />
         <generator class="identity" />
      </id>
      <property name="goodDescription" type="string">
         <column name="good_description" length="500" />
      </property>
      <property name="goodQuantity" type="java.lang.Integer">
         <column name="good_quantity" />
      </property>
      <property name="goodAsk" type="java.lang.Boolean">
         <column name="good_ask" />
      </property>
      <property name="goodCategory" type="java.lang.Boolean">
         <column name="good_category" />
      </property>
      <property name="goodDateCreate" type="timestamp">
         <column name="good_date_create" length="0" />
      </property>
      <property name="goodDateExp" type="timestamp">
         <column name="good_date_exp" length="0" />
      </property>
      <set name="goodsForGoodId" inverse="true" table="parent">
         <key>
            <column name="goo_good_id" not-null="true" />
         </key>
         <many-to-many entity-name="gr.exchange.mappings.Good">
            <column name="good_id" not-null="true" />
         </many-to-many>
      </set>
```

```
    <set name="goodexchanges" inverse="false" lazy="false">
      <key>
        <column name="good_id" not-null="true" />
      </key>
      <one-to-many class="gr.exchange.mappings.Goodexchange" />
    </set>
    <set name="users" inverse="true" table="usergood">
      <key>
        <column name="good_id" not-null="true" />
      </key>
      <many-to-many entity-name="gr.exchange.mappings.User">
        <column name="user_id" not-null="true" />
      </many-to-many>
    </set>
    <set name="goodsForGooGoodId" inverse="false" table="parent" lazy="false">
      <key>
        <column name="good_id" not-null="true" />
      </key>
      <many-to-many entity-name="gr.exchange.mappings.Good">
        <column name="goo_good_id" not-null="true" />
      </many-to-many>
    </set>
    <set name="photos" inverse="true">
      <key>
        <column name="good_id" />
      </key>
      <one-to-many class="gr.exchange.mappings.Photo" />
    </set>
  </class>
</hibernate-mapping>
```

Για τον πίνακα **Exchange** έχουμε το παρακάτω xml αρχείο:

```
<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD
3.0//EN"
"http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">
<hibernate-mapping>
  <class name="gr.exchange.mappings.Exchange" table="exchange"
catalog="diplwma">
    <id name="excId" type="int">
      <column name="exc_id" />
      <generator class="assigned" />
    </id>
    <property name="excType" type="string">
      <column name="exc_type" length="100" />
```

```
    </property>
    <property name="excActive" type="java.lang.Integer">
      <column name="exc_active" />
    </property>
    <property name="excDateCreate" type="timestamp">
      <column name="exc_date_create" length="0" />
    </property>
    <set name="grades" inverse="true">
      <key>
        <column name="exc_id" />
      </key>
      <one-to-many class="gr.exchange.mappings.Grade" />
    </set>
    <set name="goodexchanges" inverse="false" lazy="false">
      <key>
        <column name="exc_id" not-null="true" />
      </key>
      <one-to-many class="gr.exchange.mappings.Goodexchange" />
    </set>
    <set name="users" inverse="true" table="usersinexchange">
      <key>
        <column name="exc_id" not-null="true" />
      </key>
      <many-to-many entity-name="gr.exchange.mappings.User">
        <column name="user_id" not-null="true" />
      </many-to-many>
    </set>
  </class>
</hibernate-mapping>
```

Για τον πίνακα **Goodexchange** έχουμε:

```
<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD
3.0//EN"
"http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">
<hibernate-mapping>
  <class name="gr.exchange.mappings.Goodexchange" table="goodexchange"
catalog="diplwma">
    <composite-id name="id" class="gr.exchange.mappings.GoodexchangeId">
      <key-property name="goodId" type="int">
        <column name="good_id" />
      </key-property>
      <key-property name="excId" type="int">
        <column name="exc_id" />
      </key-property>
    </composite-id>
```

```
    <many-to-one name="exchange" class="gr.exchange.mappings.Exchange"
update="false" insert="false" fetch="select">
      <column name="exc_id" not-null="true" />
    </many-to-one>
    <many-to-one name="good" class="gr.exchange.mappings.Good" update="false"
insert="false" fetch="select">
      <column name="good_id" not-null="true" />
    </many-to-one>
    <property name="quantity" type="java.lang.Integer">
      <column name="quantity" />
    </property>
    <property name="propose" type="java.lang.Integer">
      <column name="propose" />
    </property>
  </class>
</hibernate-mapping>
```

Για τον πίνακα **Grade**:

```
<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD
3.0//EN"
"http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">
<hibernate-mapping>
  <class name="gr.exchange.mappings.Grade" table="grade" catalog="diplwma">
    <id name="gradeId" type="java.lang.Integer">
      <column name="grade_id" />
      <generator class="identity" />
    </id>
    <many-to-one name="exchange" class="gr.exchange.mappings.Exchange"
fetch="select">
      <column name="exc_id" />
    </many-to-one>
    <many-to-one name="user" class="gr.exchange.mappings.User" fetch="select">
      <column name="user_id" />
    </many-to-one>
    <property name="num" type="java.lang.Integer">
      <column name="num" />
    </property>
    <property name="comment" type="string">
      <column name="comment" length="1000" />
    </property>
    <property name="comDate" type="timestamp">
      <column name="com_date" length="0" />
    </property>
  </class>
</hibernate-mapping>
```

Για τον πίνακα **Member**:

```
<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD
3.0//EN"
"http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">
<hibernate-mapping>
  <class name="gr.exchange.mappings.Member" table="member" catalog="diplwma">
    <composite-id name="id" class="gr.exchange.mappings.MemberId">
      <key-property name="useUserId" type="int">
        <column name="use_user_id" />
      </key-property>
      <key-property name="userId" type="int">
        <column name="user_id" />
      </key-property>
    </composite-id>
    <many-to-one name="userByUserId" class="gr.exchange.mappings.User"
update="false" insert="false" fetch="select">
      <column name="user_id" not-null="true" />
    </many-to-one>
    <many-to-one name="userByUseUserId" class="gr.exchange.mappings.User"
update="false" insert="false" fetch="select">
      <column name="use_user_id" not-null="true" />
    </many-to-one>
    <property name="admin" type="boolean">
      <column name="admin" not-null="true" />
    </property>
  </class>
</hibernate-mapping>
```

Για τον πίνακα **MessageBuffer**:

```
<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD
3.0//EN"
"http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">
<!-- Generated 28 ??? 2009 12:12:35 ?? by Hibernate Tools 3.2.4.CR1 -->
<hibernate-mapping>
  <class name="gr.exchange.mappings.Messagebuffer" table="messagebuffer"
catalog="diplwma">
    <id name="mesId" type="java.lang.Integer">
      <column name="mes_id" />
      <generator class="identity" />
    </id>
    <property name="mesFrom" type="string">
      <column name="mes_from" length="200" />
    </property>
    <property name="mesTo" type="string">
```

```
      <column name="mes_to" length="200" />
    </property>
    <property name="mesCc" type="string">
      <column name="mes_cc" length="200" />
    </property>
    <property name="mesBcc" type="string">
      <column name="mes_bcc" length="200" />
    </property>
    <property name="mesSubject" type="string">
      <column name="mes_subject" length="100" />
    </property>
    <property name="mesText" type="string">
      <column name="mes_text" length="2000" />
    </property>
    <property name="mesInt" type="java.lang.Integer">
      <column name="mes_int" />
    </property>
  </class>
</hibernate-mapping>
```

Και τέλος για τον πίνακα **photo** έχουμε το παρακάτω αρχείο:

```
<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD
3.0//EN"
"http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">
<hibernate-mapping>
  <class name="gr.exchange.mappings.Photo" table="photo" catalog="diplwma">
    <id name="photoId" type="java.lang.Integer">
      <column name="photo_id" />
      <generator class="identity" />
    </id>
    <many-to-one name="good" class="gr.exchange.mappings.Good" fetch="select">
      <column name="good_id" />
    </many-to-one>
    <property name="photoPath" type="string">
      <column name="photo_path" />
    </property>
    <property name="photoName" type="string">
      <column name="photo_name" length="200" />
    </property>
  </class>
</hibernate-mapping>
```

9 ΒΙΒΛΙΟΓΡΑΦΙΑ

1. Λεοντίεφ. Α. , *Πολιτική Οικονομία.* , Κοινωνικές εκδόσεις Αθήνα 1978
2. http://www.soc.uoc.gr/appapa/courses/monpolnotes.pdf
3. http://www.focusmag.gr/articles/view-article.rx?oid=27224
4. http://www.freecycle.org
5. http://www.peliti.gr/diktia_sinalagis.htm
6. https://www.hibernate.org/
7. http://www.eclipse.org/
8. Sepandar D. Kamvar, Mario T. Schlosser and Hector Garcia-Molina. The Eigen-Trust Algorithm For Reputation Management in P2P Networks
9. http://gargoyle.arcadia.edu/mathcs/zhengpei/StudentPapers/p2preputation.doc
10. http://yury.name/reputation/01reputation-hand.pdf
11. http://www.cis.upenn.edu/~westand/docs/p2psim_chapter_draft.pdf
12. http://el.wikipedia.org/wiki