ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ –
ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΗΛΕΚΤΡΟΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ ΚΑΙ ΔΙΚΤΥΩΝ

Σχεδίαση και υλοποίηση δικτυακού ηλεκτρονικού καταστήματος (e-shop) με χρήση Java EE, JSF, μηχανισμούς αντιστοίχισης σχεδιακού μοντέλου βάσης (Object Relational Mapping) καθώς και ολοκλήρωση με συστήματα ασφαλών τραπεζικών συναλλαγών proxypay3 (Secure Payment Module).

Διπλωματική εργασία

Σιμήτα Παρασκευή

Επιβλέπων

Χούστης Ηλίας, Καθηγητής

Δεύτερο μέλος επιτροπής

Παπαδάκης Νικόλαος, Συμβασιούχος ΠΔ 407/80

ΟΚΤΩΒΡΙΟΣ 2009

ΒΟΛΟΣ

Πρόλογος

Ο στόχος της παρούσας διπλωματικής εργασίας είναι η σχεδίαση και υλοποίηση ηλεκτρονικού καταστήματος (e-shop). Για την επίτευξη αυτού του στόχου γίνεται μελέτη σχετικών τεχνολογιών προγραμματισμού καθώς επίσης και μία αναφορά στο σύστημα ασφαλών τραπεζικών συναλλαγών proxypay3.

Η ύλη διαρθρώνεται σε δύο βασικές ενότητες :

Η πρώτη ενότητα περιέχει μία περιγραφή των φάσεων ανάπτυξης του λογισμικού. Κατανέμονται σε 3 κεφάλαια ως εξής :

- Στο 1ο κεφάλαιο, γίνεται περιγραφή διεργασίας ανάπτυξης λογισμικού.
- Στο 2ο κεφάλαιο, γίνεται εξαγωγή, ανάλυση, εξακρίβωση και έλεγχος και συμπλήρωση απαιτήσεων και προδιαγραφών.
- Στο 3ο κεφάλαιο, γίνεται η σχεδίαση.

Η δεύτερη ενότητα περιέχει μία παρουσίαση των τεχνολογιών Java EE, JSF, μηχανισμών αντιστοίχισης σχεσιακού μοντέλου βάσης και του συστήματος ασφαλών τραπεζικών συναλλαγών proxypay3.

- Στο 4ο κεφάλαιο, γίνεται περιγραφή των Java EE και JSF.
- Στο 5ο κεφάλαιο, γίνεται περιγραφή των μηχανισμών αντιστοίχισης σχεσιακού μοντέλου βάσης
- Στο 6ο κεφάλαιο, γίνεται περιγραφή Strus.
- Στο 7ο κεφάλαιο, γίνεται περιγραφή του συστήματος ασφαλών τραπεζικών συναλλαγών.

Στο σημείο αυτό θα ήθελα να ευχαριστήσω τον κ. Χούστη Ηλία και τον κ. Παπαδάκη Νικόλαο για την καθοδήγηση, την υπομονή και τη βοήθειά τους σε όλη τη διάρκεια της φοίτησής μου και για την ολοκλήρωση αυτής της διπλωματικής εργασίας.

Οκτώβριος 2009

Σιμήτα Παρασκευή

Περιεχόμενα

Πρόλογος....3

1. Περιγραφή διεργασίας ανάπτυξης λογισμικού....8

1.1 Εισαγωγή 8

1.2 Γενικό αντικείμενο διπλωματικής εργασίας....8

2. Εξαγωγή, ανάλυση, εξακρίβωση και έλεγχος και συμπλήρωση απαιτήσεων και προδιαγραφών....10

2.1 Βασικές λειτουργίες 10

2.2 Κριτήρια σχεδίασης 11

2.3 Απαιτήσεις συμπεριφοράς 12

2.3.1 Διάγραμμα περίπτωσης χρήσης (Use Case Diagram)....12

2.3.2 Διαγράμματα αλληλουχίας (Sequence diagrams)....19

2.3.3 Διάγραμμα κατηγορίας (Class Diagrams)....28

2.3.4 Δομή συστήματος....29

2.3.5 Σχέδιο δεδομένων....30

3. Σχεδίαση....32

3.1 Επιλογή Αρχιτεκτονικής 32

3.1.1 Αρχιτεκτονική client-server....32

3.1.2 Αρχιτεκτονική πελάτη-εξυπηρετητή τριών επιπέδων (3-Tiers)....33

3.1.3 Σύγκριση αρχιτεκτονικών....35

3.2 Αποσύνθεση συστήματος 37

3.3 Σύνολο αποκοπής 37

3.3.1 Δέντρο σφαλμάτων....37

3.3.2 Υπολογισμός συνόλου αποκοπής....38

4. Java EE και JSF....40

5. Μηχανισμοί αντιστοίχισης σχεσιακού μοντέλου βάσης....41

6. Strus....48

7. Σύστημα ασφαλών τραπεζικών συναλλαγών....50

8. Βιβλιογραφία....53

Σχεδίαση και υλοποίηση δικτυακού ηλεκτρονικού καταστήματος (e-shop) με χρήση Java EE, JSF, μηχανισμούς αντιστοίχισης σχεσιακού μοντέλου βάσης (Object Relational Mapping) καθώς και ολοκλήρωση με συστήματα ασφαλών τραπεζικών συναλλαγών proxypay3 (Secure Payment Module).

---

1. Περιγραφή διεργασίας ανάπτυξης λογισμικού

1.1 Εισαγωγή

Γενικά επικρατεί μία νέα τάση επιχειρηματικής δραστηριότητας. Οργανισμοί, επιχειρήσεις, αλλά και άτομα, δραστηριοποιούνται όλο και συχνότερα επιχειρηματικά μέσω του Διαδικτύου, προκειμένου να εκμεταλλευτούν την ταχύτητα και την αυξημένη αποτελεσματικότητα που προσφέρεται. Ο νεολογισμός του e-business ερμηνεύεται από δύο οπτικές γωνίες: α) την ως προς τη χρήση της τεχνολογίας για τον επανασχεδιασμό των επιχειρησιακών διαδικασιών (Internal focus) και β) την ως προς τη σχέση της επιχείρησης με το εξωτερικό της περιβάλλον και ειδικά τους προμηθευτές και τους πελάτες (external focus).

Η Πληροφορική Επανάσταση άλλαξε ριζικά τον τρόπο ζωής και επέφερε αλλαγές στις επιχειρήσεις, οι οποίες αποτελούν σημαντικό όπλο στα χέρια των επιχειρήσεων με τη δημιουργία Ηλεκτρονικών καταστημάτων και αγορών. Η δυνατότητα διαξαγωγής πραγματικών αγορών από το σπίτι ξεκίνησε νε τη χρήση της τηλεόρασης και του τηλεφώνου, αλλά σήμερα έχει επεκταθεί και γίνεται μέσω των Η/Υ και του Διαδικτύου και κυρίως με τη χρήση Ηλεκτρονικών Καταστημάτων. Επιπλέον, οι επιχειρήσεις έχουν τη δυνατότητα να εξατομικεύσουν και να αναπληρώσουν τα αποθέματά τους με πιο αποδοτικό τρόπο και να προωθήσουν αποτελεσματικότερα τα προϊόντα τους,

1.2 Γενικό αντικείμενο διπλωματικής εργασίας

Το παρόν έγγραφο περιγράφει την υλοποίηση ενός ηλεκτρονικού καταστήματος. Κύριος στόχος του είναι η προώθηση της ηλεκτρονικής παραγγελίας. Η αγορά μέσω ενός ηλεκτρονικού καταστήματος έχει το πλεονέκτημα ότι ο πελάτης μπορεί να τροποποιήσει την παραγγελία του ακόμη και αν έχει ήδη επιλέξει τα προϊόντα που θέλει (από τη στιγμή που δεν την έχει αποστείλει) προσφέροντάς του ευελιξία, έχοντας τον απόλυτο έλεγχο στην παραγγελία. Περαιτέρω, είναι συχνό το φαινόμενο να θέλει κανείς να κάνει την αγορά του όταν έχει χρόνο ή διάθεση, οπότε από τον υπολογιστή του μπορεί να διαχειρίζεται πάντα αυτά που θέλει να αγοράσει αλλά και τον τρόπο με τον οποίο θα το κάνει. Τέλος, το μεγαλύτερο όφελος ενός τέτοιου είδους ηλεκτρονικού καταστήματος είναι το γεγονός ότι δεν χρειάζεται να επενδύσει ο ιδιοκτήτης χρήματα σε κάποιο ακίνητο καθώς και σε όλα επιπλέον έξοδα που αυτό επιφέρει. Τελικά, με ένα απλό ηλεκτρονικά κατάστημα μειώνει το κόστος λειτουργίας για την επιχείρησή του, αλλά και διευκολύνει τον πελάτη να βρει αυτό ακριβώς που ζητάει.

Ο σχεδιασμός της σελίδας βασίζεται στην απλή, εύκολη, γρήγορη και εύληπτη ενημέρωση του χρήστη. Επιπλέον, πλαισιώνεται με φωτογραφίες και αρκετά πεδία επιλογής έτσι ώστε να

είναι πιο άμεση, χειροπιαστή και απτή η διαδικασία παραγγελίας για να μην οδηγείται σε λάθη ο χρήστης και να προβλέπεται μία λάθος επιλογή. Γενικά, προωθείται το φιλικό περιβάλλον ο οποίος για να φτάσει στον στόχο του δε χρειάζεται να μπει σε ατέρμονες διαδικασίες περιττών κινήσεων και μεταβάσεων από σελίδα σε σελίδα, μειώνοντας το φόρτο του εξυπηρετητή.

Το πελατολόγιο στο οποίο απευθύνεται το συγκεκριμένο ηλεκτρονικό κατάστημα μπορεί να αφορά άτομα όλων των ηλικιών αλλά και ενδιαφερόντων, τα οποία όμως πρέπει να έχουν μία επαφή με Η/Υ και Internet, χωρίς αυτό να απαιτεί κάποια ιδιαίτερη εξειδίκευση, αφού μπορεί άνετα ο βασικός σκελετός να τροποποιηθεί για την πώληση οποιουδήποτε προϊόντος. Ειδικά, η χρήση εικόνων καθοδηγεί τον επισκέπτη της σελίδας με μοναδικό τρόπο, βάζοντας τον σε πειρασμό να αγοράσει κάτι πριν φύγει από την ιστοσελίδα.

Στόχος είναι η κατασκευή από το μηδέν ενός εύχρηστου, φιλικού προς το χρήστη και λειτουργικού ηλεκτρονικού καταστήματος. Επιμέρους στόχος καθορίστηκε να είναι η εύκολη διαχείρισή του από την πλευρά του διαχειριστή/ιδιοκτήτη (χωρίς να γνωρίζει προγραμματισμό ή άλλου είδους τεχνική λεπτομέρεια) καθώς και η εύκολη παραμετροποίησή του. Έτσι λοιπόν, η εφαρμογή αποτελείται στην ουσία από δύο επιμέρους εφαρμογές που απευθύνονται σε δύο διαφορετικές κατηγορίες χρηστών:

- τους απλούς χρήστες οι οποίοι μπορούν να συνδέονται στην βασική εφαρμογή και κάνουν τις παραγγελίες τους αφού πρώτα εγγραφούν

- τους εξουσιοδοτημένους χρήστες ή διαχειριστές του καταστήματος (administrators) οι οποίοι συνδέονται σε ξεχωριστή εφαρμογή και έχουν πλήρη δικαιώματα διαχείρισης της εφαρμογής του χρήστη όπως προσθήκη, τροποποίηση ή διαγραφή προϊόντων και κατηγοριών, επεξεργασία παραγγελίας πελατών κτλ.

2. Εξαγωγή, ανάλυση, εξακρίβωση και έλεγχος και συμπλήρωση απαιτήσεων και προδιαγραφών

2.1 Βασικές λειτουργίες

Οι βασικές λειτουργίες του ηλεκτρονικού καταστήματος θα είναι οι εξής:

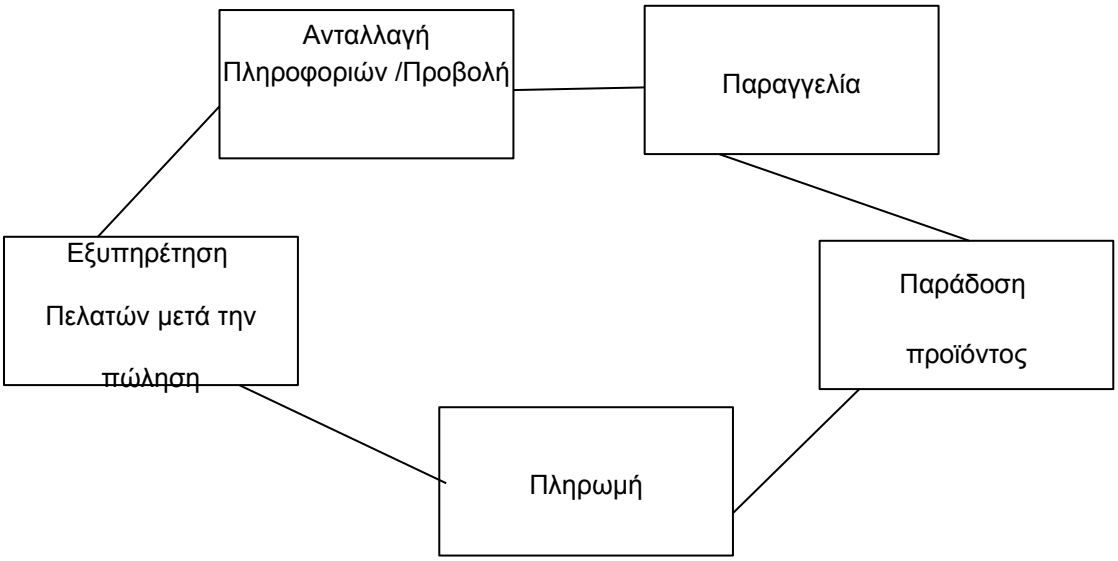
- Ανταλλαγή Πληροφοριών / Προβολή: χρησιμοποιώντας το e-shop με τις ηλεκτρονικές φόρμες, τους ηλεκτρονικούς καταλόγους και το e-mail, μπορεί να γίνει άντληση πληροφοριών από τους υπάρχοντες και πιθανούς πελάτες, ενώ ταυτόχρονα γίνεται προβολή των προϊόντων του καταστήματος. Ενώ, οι πελάτες μαθαίνουν για τα προϊόντα του e-shop, ο κάτοχος αυτού μαθαίνει τις καταναλωτικές συνήθειες και τα μελλοντικά σχέδια των πελατών.

- Παραγγελία: χρησιμοποίηση ηλεκτρονικών φορμών και ηλεκτρονικού ταχυδρομείου για την αποστολή – λήψη παραγγελιών.

- Παράδοση Προϊόντος: προϊόντα όπως λογισμικό, βιβλία, μουσική, φωτογραφίες, σχέδια, ηλεκτρομηχανολογικά είναι δυνατόν να παραδοθούν ηλεκτρονικά.

- Πληρωμή: π.χ. Αριθμός πιστωτικής κάρτας, ηλεκτρονικό χρήμα, αντικαταβολή, κτλ.

- Εξυπηρέτηση πελατών μετά την πώληση: λειτουργία σελίδας παραπόνων πελάτη, σελίδα εξυπηρέτησης με συχνά διατυπωμένες ερωτήσεις, συντήρηση προϊόντων, ημερομηνίες κυκλοφορίας νέας έκδοσης, κτλ.

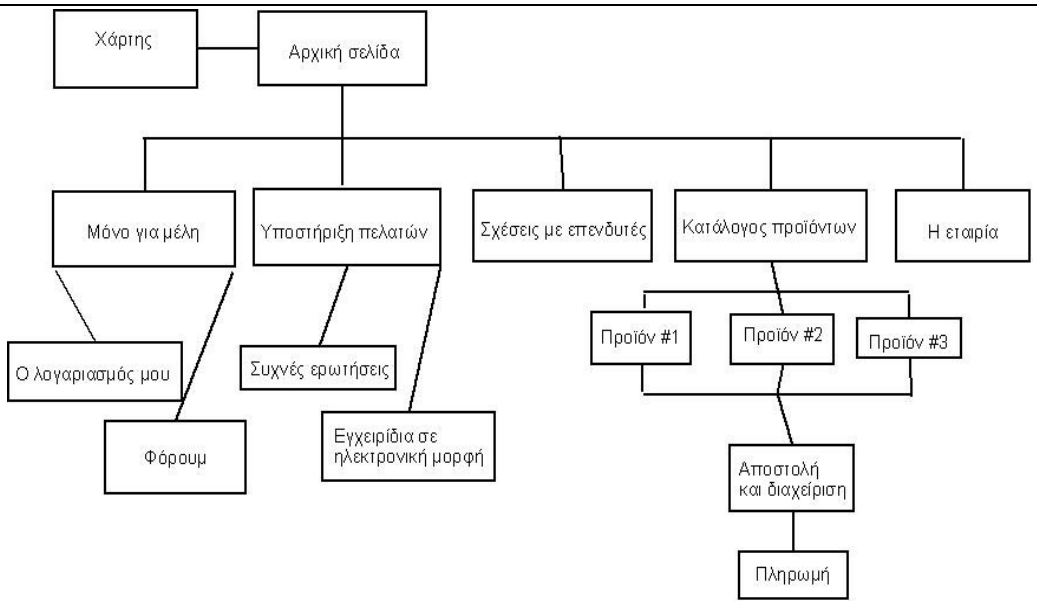


Εικόνα: Κύκλος βασικών λειτουργιών

2.2 Κριτήρια σχεδίασης

Τα βασικά κριτήρια σχεδίασης σχετίζονται με την ικανοποίηση των προσδοκιών του πελάτη και πιο συγκεκριμένα με τον προσδιορισμό των αναγκών, των προσδοκιών και των προβλημάτων του πελάτη. Μία λίστα σημαντικών κριτηρίων βάσει των οποίων έγινε ο σχεδιασμός είναι τα εξής:

- Περιήγηση: Είναι εύκολο να βρουν οι επισκέπτες το δρόμο μέσα στις ιστοσελίδες;
- Συνέπεια: Είναι τα στοιχεία σχεδίασης, ειδιακά τα στοιχεία εμφάνισης και αίσθησης, συνεπή σε όλες τις σελίδες;
- Χρόνος απόκρισης: Πόσος χρόνος απαιτείται για να εμφανιστεί μια ιστοσελίδα;
- Εμφάνιση: Είναι αισθητικά ελκυστική; Η εμφάνιση και η αίσθηση εκφράζουν την εικόνα που επιδιώκεται να μεταδοθεί;
- Διασφάλιση ποιότητας: Εργάζονται σωστά τα συστήματα υπολογισμού, οι συνδέσεις περιήγησης, οι διεργασίες εγγραφής επισκεπτών, τα εργαλεία αναζήτησης και τα υπόλοιπα εργαλεία;
- Διαθεσιμότητα: Διορθώνονται γρήγορα νεκρές συνδέσεις, είναι διαθέσιμο το ηλεκτρονικό κατάστημα 24 ώρες την ημέρα, 7 ημέρες την εβδομάδα;
- Διαδραστικότητα: Ενθαρρύνει τον επισκέπτη να παίξει ενεργό ρόλο στην εκμάθηση πληροφοριών για τα προϊόντα ή τις υπηρεσίες της επιχείρησης;
- Περιεχόμενο: Πόσα πολυμέσα χρησιμοποιούνται; Πόσο επίκαιρο και σχετικό είναι το περιεχόμενο; Είναι ευανάγνωστο;
- Χρησιμοποίηση: Πόσο εύκολη είναι η χρήση; Πόσο εύκολο είναι να γίνουν σφάλματα και να διορθωθούν;
- Ασφάλεια: Προστατεύονται οι πληροφορίες των πελατών; Αισθάνονται ασφαλής;
- Κλιμάκωση: Παρέχεται η δυνατότητα για εύκολη αναβάθμιση, τροποποίηση στο μέλλον; Η εξέλιξη και η αυξημένη χρήση προστατεύουν την αρχική επένδυση για την κατασκευή του ηλεκτρονικού καταστήματος;



Εικόνα: Ιεραρχική δομή

Σύμφωνα με το παραπάνω διάγραμμα το e-shop περιέχει μία αρχική σελίδα, που καλωσορίζει τον επισκέπτη και τον εισάγει στο κλίμα του δικτυακού τόπου, σελίδες βοήθειας, που βοηθούν τον επισκέπτη να χρησιμοποιήσει τις ιστοσελίδες, εταιρικές σελίδες, που πληροφορούν τον επισκέπτη για την ηλεκτρονική επιχείρηση, σελίδες συναλλαγών, που καθοδηγούν τον πελάτη μέσα στη διαδικασία αγορών και σελίδες περιεχομένων, που παρέχουν πληροφορίες για προϊόντα και υπηρεσίες σε όλες τις φάσεις της διαδιασίας αγορών, από αναζήτηση πληροφοριών μέχρι εξυπηρέτηση μετά την αγορά και αποτίμηση αγοράς.

2.3 Απαιτήσεις συμπεριφοράς

2.3.1 Διάγραμμα περίπτωσης χρήσης (Use Case Diagram)

Παρακάτω παρατίθονται οι περιπτώσεις χρήσης οι οποίες προκύπτουν από μία διαδικασία κατά την οποία σκοπός μας είναι να εντοπίσουμε τους συμμετέχοντες στο σύστημα, τις διεργασίες του συστήματος καθώς και το χρόνο κατά τον οποίο αυτές εκτελούνται. Συγκεκριμένα “απαντώντας” στις παρακάτω ερωτήσεις μπορούμε ευκολότερα και ασφαλέστερα να πετύχουμε το στόχο μας.

- Ποιοι χρήστες ή ομάδες χρησιμοποιούν το σύστημα για να εκτελέσουν μια εργασία.
- Ποιοι χρήστες ή ομάδες χρειάζονται, έτσι ώστε το σύστημα να εκτελέσει τις λειτουργίες του.

- Ποια εξωτερικά συστήματα χρησιμοποιούν το σύστημα για να εκτελέσουν μια εργασία.
- Ποια εξωτερικά συστήματα, χρήστες ή ομάδες, στέλνουν πληροφορίες στο σύστημα.
- Ποια εξωτερικά συστήματα, χρήστες ή ομάδες, παραλαμβάνουν πληροφορίες από το σύστημα.

Έτσι προκύπτουν οι περιπτώσεις χρήσης, μερικές από τις οποίες είναι οι εξής:

UC1 - Turn on server

Προϋποθέσεις: UC2

Τύπος χρήστη: Διαχειριστής συστήματος

Περιγραφή ροής: 1. Ο διαχειριστής του συστήματος ανάβει το server που περιέχει την βάση δεδομένων του προγράμματος.

2. Οι άλλοι υπολογιστές μπορούν πλέον να επικοινωνήσουν με το server και οι χρήστες να χρησιμοποιήσουν το πρόγραμμα.

UC2 - Turn off server

Προϋποθέσεις: UC1

Τύπος χρήστη: Διαχειριστής συστήματος

Περιγραφή ροής: 1. Ο διαχειριστής του συστήματος turns off το server που περιέχει την βάση δεδομένων του προγράμματος.

2. Οι άλλοι υπολογιστές που χρησιμοποιούν το πρόγραμμα δεν μπορούν πλέον να κάνουν login ή οποιαδήποτε πράξη που αφορά την κεντρική βάση δεδομένων.

UC3 - Login στο σύστημα

Προϋποθέσεις: UC1

Τύπος χρήστη: Διαχειριστής συστήματος, πελάτης.

Περιγραφή ροής: 1. Ο χρήστης εισάγει το username και το password του και πατάει το κουμπί login.

2. Το σύστημα ψάχνει στην βάση δεδομένων για το username

3. Αν βρεθεί, ελέγχει αν το password είναι σωστό.

4. Αν ταιριάζουν, το σύστημα εμφανίζει κατάλληλο αρχικό μενού ανάλογα με τα δικαιώματα του χρήστη που έχει κάνει login

Εναλλακτική ροή: 1. Ο χρήστης εισάγει το username και το password του και πατάει το κουμπί login.

2. Το σύστημα ψάχνει στην βάση δεδομένων για το username

3. Αν δεν βρεθεί, το σύστημα εμφανίζει κατάλληλο μήνυμα στον χρήστη.

4. Επιστροφή στο αρχικό μενού του login

Εναλλακτική ροή: 1. Ο χρήστης εισάγει το username και το password του και πατάει το κουμπί login.

2. Το σύστημα ψάχνει στην βάση δεδομένων για το username

3. Αν βρεθεί, ελέγχει αν το password είναι σωστό.

4. Αν δεν ταιριάζουν, το σύστημα εμφανίζει κατάλληλο μήνυμα στον χρήστη.

5. Επιστροφή στο αρχικό μενού του login

UC4 - Logout από το σύστημα

Προϋποθέσεις: UC3

Τύπος χρήστη: Διαχειριστής συστήματος, πελάτης.

Περιγραφή ροής: 1. Το σύστημα ζητάει confirmation.

2. Ο χρήστης πατάει logout.

3. Επιστροφή στο αρχικό μενού του login.

Εναλλακτική ροή: 1. Το σύστημα ζητάει confirmation.

2. Ο χρήστης πατάει cancel.

3. Επιστροφή στο αρχικό μενού χρήστη.

UC5 - Αναβάθμιση λογισμικού

Προϋποθέσεις: UC3

Τύπος χρήστη: Διαχειριστής συστήματος

Περιγραφή ροής:

1. Ο διαχειριστής του συστήματος αρχίζει την εγκατάσταση της αναβάθμισης.

2. Το σύστημα δημιουργεί ένα backup της τρέχουσας κατάστασης του λογισμικού.

3. Η εγκατάσταση της αναβάθμισης ολοκληρώνεται.

4. Reboot του server.

5. Επιστροφή στο αρχικό μενού του login.

Εναλλακτική ροή:

1. Ο διαχειριστής του συστήματος πατάει cancel

2. Επιστροφή στο αρχικό μενού διαχείρισης.

UC6 - Κατάργηση λογισμικού

Προϋποθέσεις: UC3

Τύπος χρήστη: Διαχειριστής συστήματος

Περιγραφή ροής:

1. Ο διαχειριστής του συστήματος πατάει το κουμπί recover from backup

2. Το σύστημα απεγκαθιστά την τρέχουσα έκδοση του λογισμικού.

3. Το σύστημα επανεγκαθιστά μία παλιά έκδοση του λογισμικού από το τελευταίο backup.

4. Reboot του server.

5. Επιστροφή στο αρχικό μενού του login.

Εναλλακτική ροή:

1. Ο διαχειριστής του συστήματος πατάει το κουμπί delete all software

2. Το σύστημα ψάχνει για backup του λογισμικού και αν βρεθεί, το διαγράφει.

3. Το σύστημα απεγκαθιστά την τρέχουσα έκδοση του λογισμικού

4. Reboot του server.

Εναλλακτική ροή:

1. Ο διαχειριστής του συστήματος πατάει cancel

2. Επιστροφή στο αρχικό μενού διαχείρισης.

UC7 - Δημιουργία login/password στην βάση δεδομένων

Προϋποθέσεις: UC3

Τύπος χρήστη: Διαχειριστής συστήματος

Περιγραφή ροής:
1. Ο διαχειριστής του συστήματος εισάγει νέο username/password και τα δικαιώματα που θα έχει ο συγκεκριμένος χρήστης
2. Η πληροφορία αυτή κρυπτογραφείται
3. Η κρυπτογραμμένη πληροφορία αποθηκεύεται στην βάση δεδομένων
4. Επιστροφή στο αρχικό μενού διαχείρισης

Εναλλακτική ροή:
1. Ο διαχειριστής του συστήματος πατάει cancel
2. Επιστροφή στο αρχικό μενού διαχείρισης.

UC8 - Αλλαγή login/password στην βάση δεδομένων

Προϋποθέσεις: UC3

Τύπος χρήστη: Διαχειριστής συστήματος

Περιγραφή ροής:
1. Ο διαχειριστής του συστήματος κάνει αναζήτηση στην βάση δεδομένων για κάποιο login
2. Το σύστημα ψάχνει στην βάση δεδομένων γι' αυτό το login
3. Αν βρεθεί, το σύστημα εμφανίζει μία σελίδα στην οποία ο διαχειριστής του συστήματος μπορεί να αλλάξει είτε το login, είτε το password, είτε και τα δύο.
4. Ο διαχειριστής πατάει το κουμπί save για να αποθηκευτούν οι αλλαγές.
5. Επιστροφή στο αρχικό μενού διαχείρισης.

Εναλλακτική ροή:
1. Ο διαχειριστής του συστήματος κάνει αναζήτηση στην βάση δεδομένων για κάποιο login
2. Το σύστημα ψάχνει στην βάση δεδομένων γι' αυτό το login
3. Αν δεν βρεθεί, το σύστημα ειδοποιεί τον διαχειριστή με κατάλληλο μήνυμα.
4. Επιστροφή στο αρχικό μενού διαχείρισης.

Εναλλακτική ροή: 1. Ο διαχειριστής του συστήματος πατάει cancel

2. Επιστροφή στο αρχικό μενού διαχείρισης.

UC9 - Δημιουργία νέας εγγραφής πελάτη

Προϋποθέσεις: UC3

Τύπος χρήστη: Πελάτης

Περιγραφή ροής: 1. Ο πελάτης συμπληρώνει τα κενά με τα στοιχεία και πατάει το κουμπί save new entry.

2. Το σύστημα δημιουργεί νέα εγγραφή τύπου πελάτη στην βάση δεδομένων και αποθηκεύει τα στοιχεία που έδωσε ο χρήστης

3. Επιστροφή στο αρχικό μενού.

Εναλλακτική ροή: 1. Ο πελάτης πατάει το κουμπί cancel.

2. Επιστροφή στο αρχικό μενού.

UC10 - Δημιουργία καλαθιού

Προϋποθέσεις: UC3

Τύπος χρήστη: Πελάτης

Περιγραφή ροής: 1. Ο πελάτης διαλέγει προϊόντα ή εκτελεί αναζήτηση.

2. Το σύστημα ψάχνει στην βάση δεδομένων και επιστρέφει τα αποτελέσματα.

3. Ο πελάτης επιλέγει κάποια από αυτά και πατάει το κουμπί next

4. Το σύστημα εμφανίζει μία σελίδα στην οποία ο χρήστης πρέπει να συμπληρώσει τα στοιχεία του.

5. Το σύστημα αποθηκεύει τα δεδομένα στην βάση δεδομένων

6. Επιστροφή στο αρχικό μενού.

Εναλλακτική ροή: 1. Ο πελάτης διαλέγει προϊόντα ή εκτελεί αναζήτηση.

2. Το σύστημα ψάχνει στην βάση δεδομένων και επιστρέφει τα αποτελέσματα.

3. Ο πελάτης επιλέγει κάποια από αυτά και πατάει το κουμπί next

4. Το σύστημα εμφανίζει μία σελίδα στην οποία ο χρήστης πρέπει να συμπληρώσει τα στοιχεία του.

5. Το σύστημα αποθηκεύει τα δεδομένα στην βάση δεδομένων.

6. Ο πελάτης πατάει cancel.

7. Το σύστημα αποθηκεύει τα δεδομένα στην βάση δεδομένων.

8. Επιστροφή στο αρχικό μενού.

Εναλλακτική ροή: 1. Ο πελάτης πατάει το κουμπί cancel σε οποιοδήποτε βήμα πριν το 8ο των παραπάνω ροών.

2. Επιστροφή στο αρχικό μενού.

UC11 - Αλλαγή στοιχείων πελάτη

Προϋποθέσεις: UC3

Τύπος χρήστη: Πελάτης

Περιγραφή ροής: 1. Ο πελάτης κάνει Login και πατάει στο προφίλ του.

2. Το σύστημα ψάχνει στην βάση δεδομένων και επιστρέφει μία λίστα χαρακτηριστικών που αφορούν τον συγκεκριμένο πελάτη.

3. Ο πελάτης επιλέγει το κουμπί modify data.

4. Το σύστημα ψάχνει στην βάση δεδομένων για τα στοιχεία και τα εμφανίζει με δυνατότητα αλλαγών.

5. Ο πελάτης κάνει κάποιες άλλες πιθανές αλλαγές και πατάει το κουμπί save.

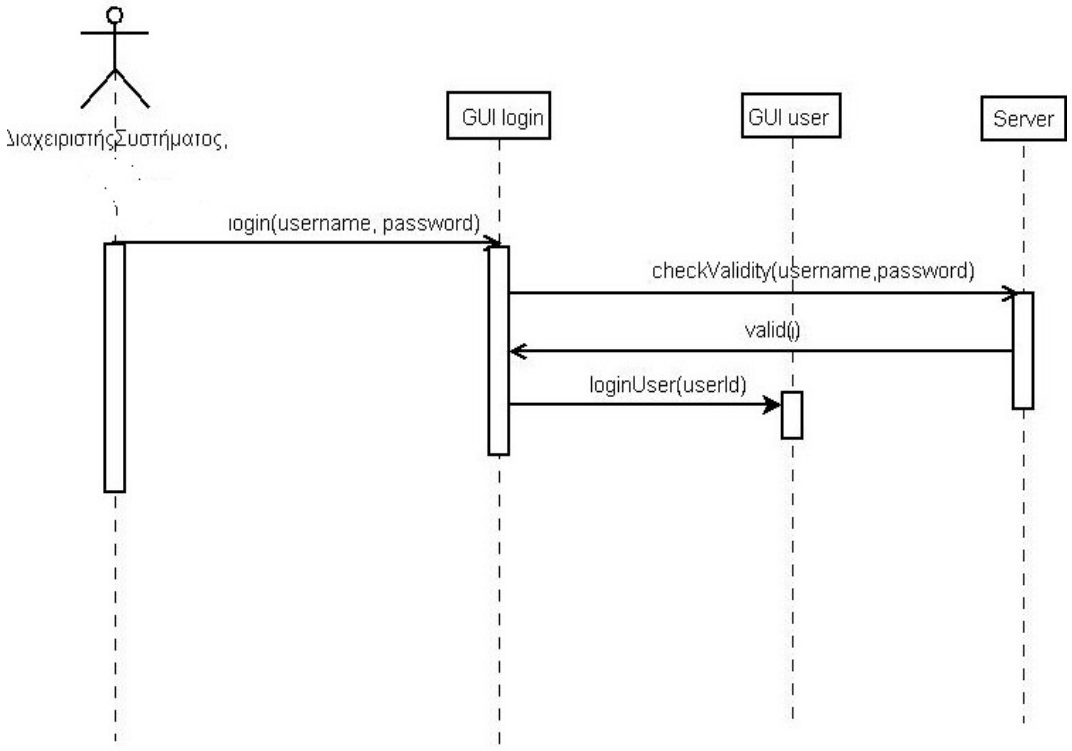
8. Το σύστημα αποθηκεύει τις αλλαγές στην βάση δεδομένων

9. Επιστροφή στο αρχικό μενού.

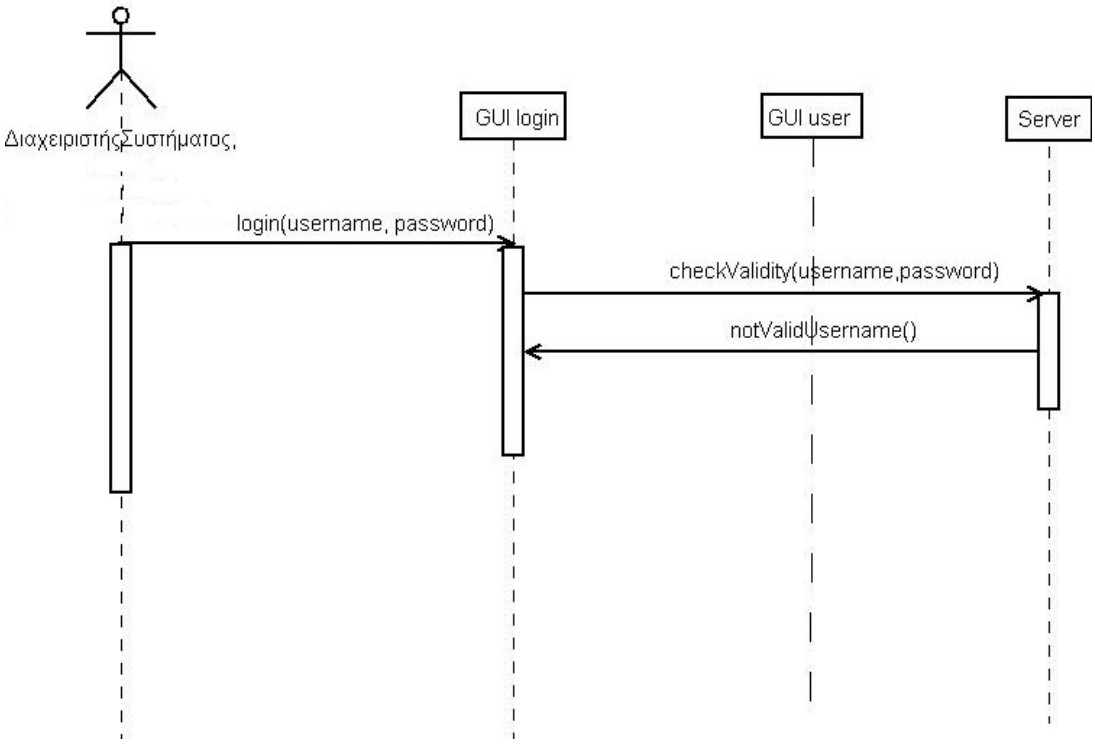
Εναλλακτική ροή: 1. Ο πελάτης πατάει το κουμπί cancel.

2. Επιστροφή στο αρχικό μενού.

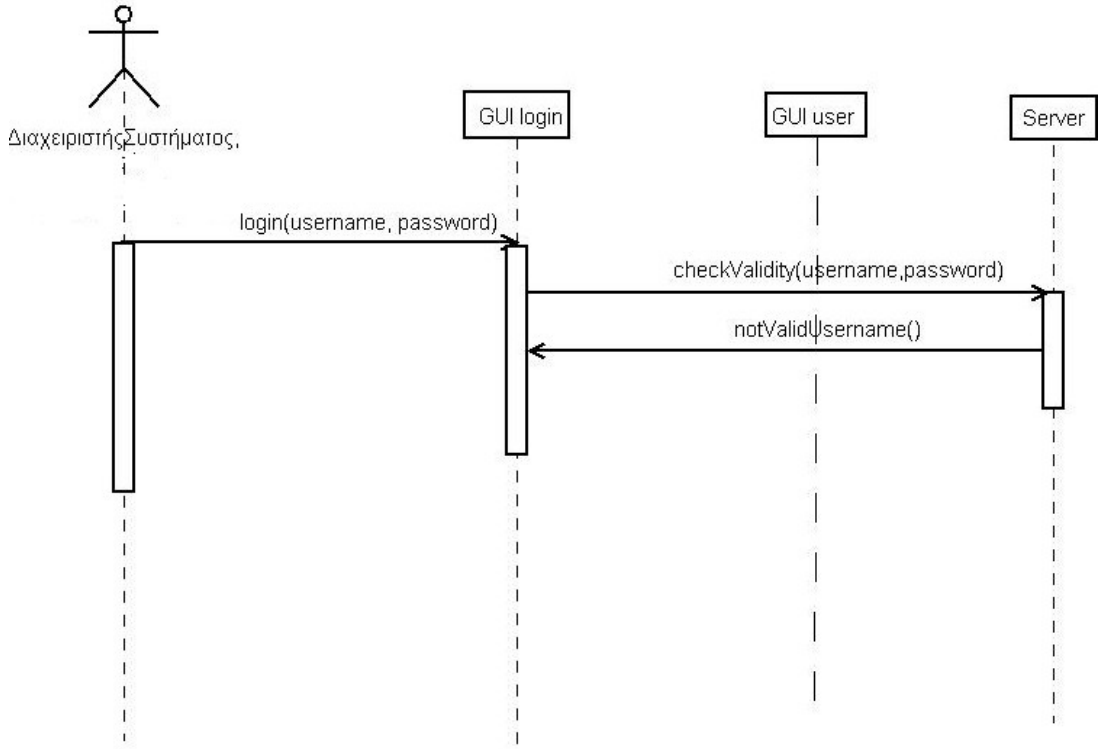
2.3.2 Διαγράμματα αλληλουχίας (Sequence diagrams)



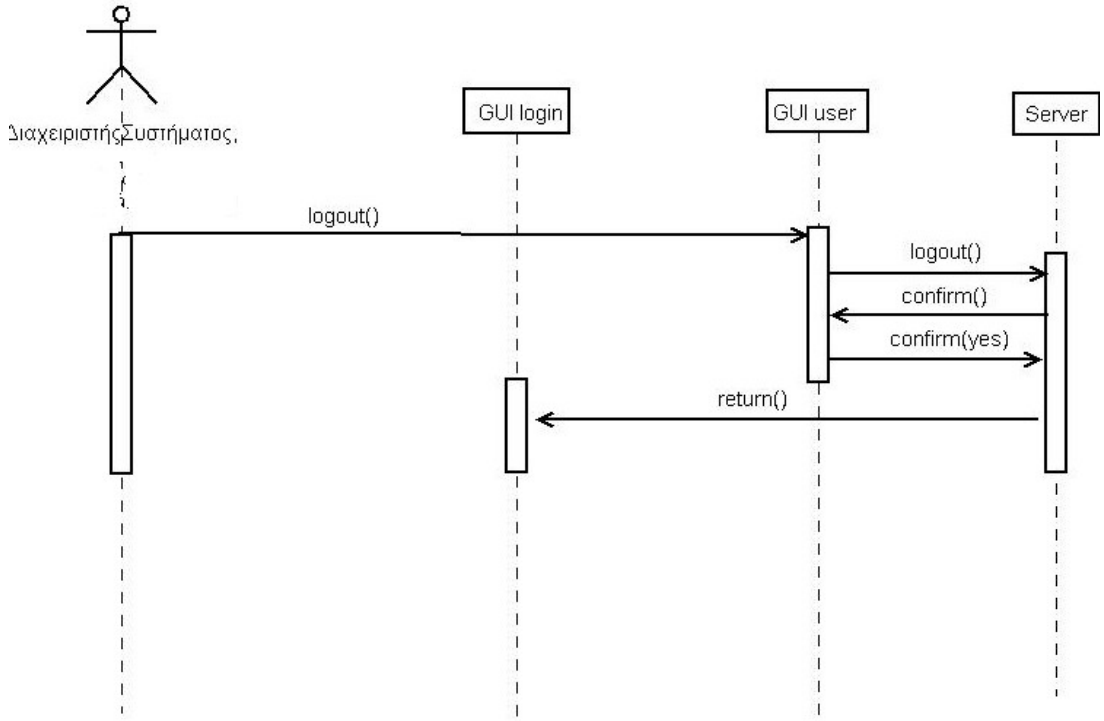
Εικόνα: Login στο σύστημα



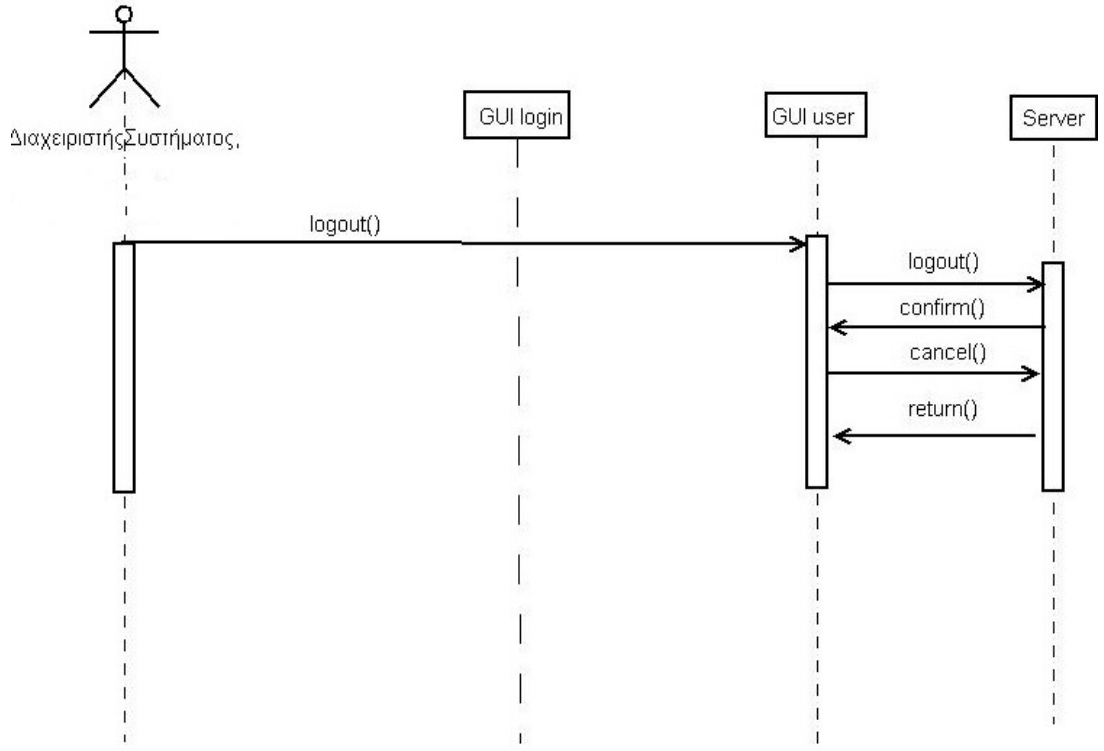
Εικόνα: Login στο σύστημα (αποτυχία του server)



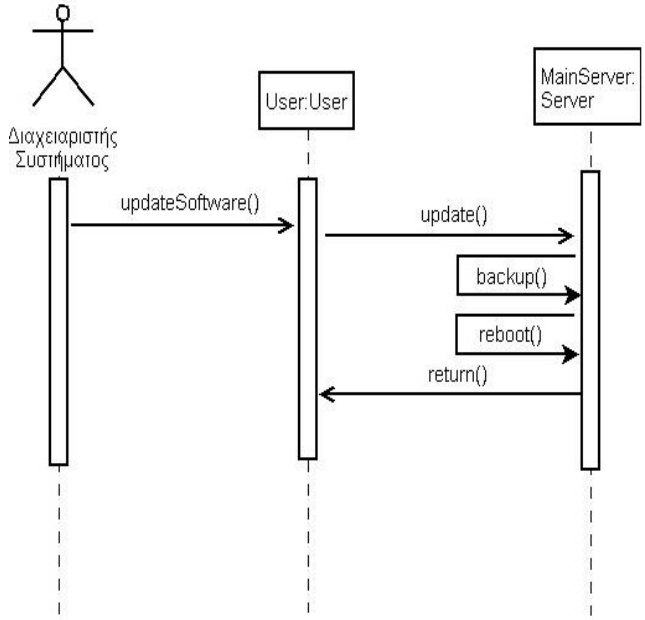
Εικόνα: Login στο σύστημα (αποτυχία του password)



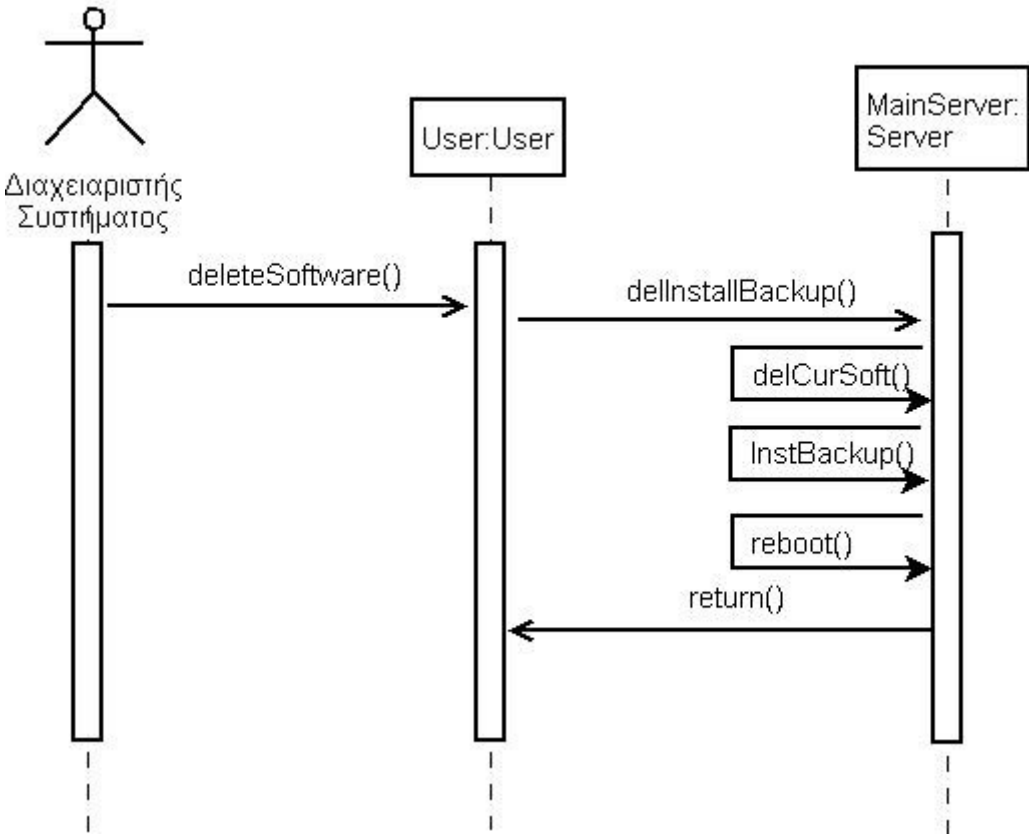
Εικόνα: Logout από το σύστημα (επιτυχία)



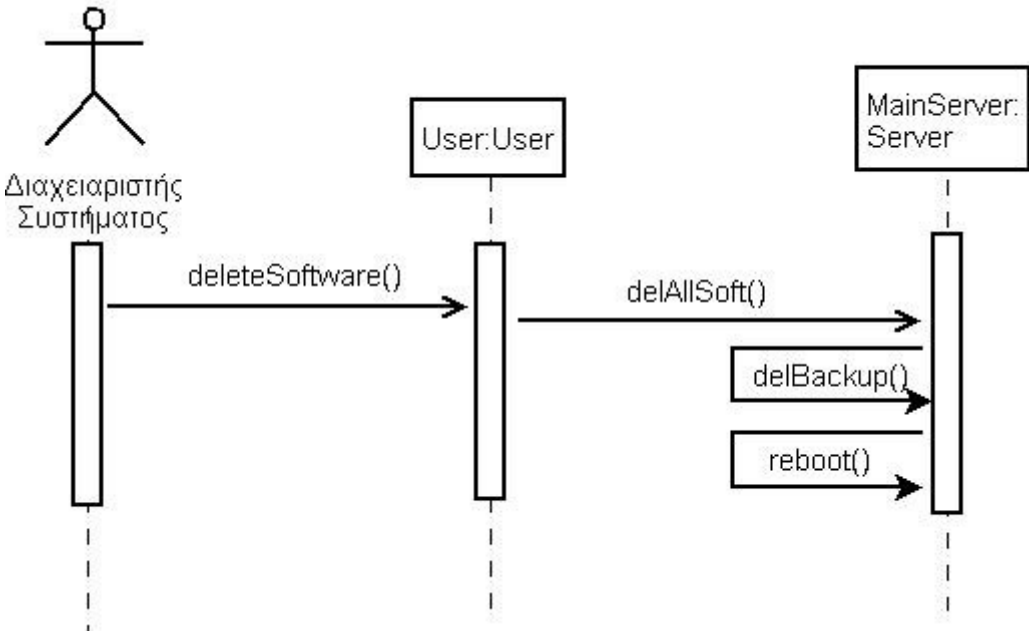
Εικόνα: Logout από το σύστημα (ακύρωση από τον χρήστη)



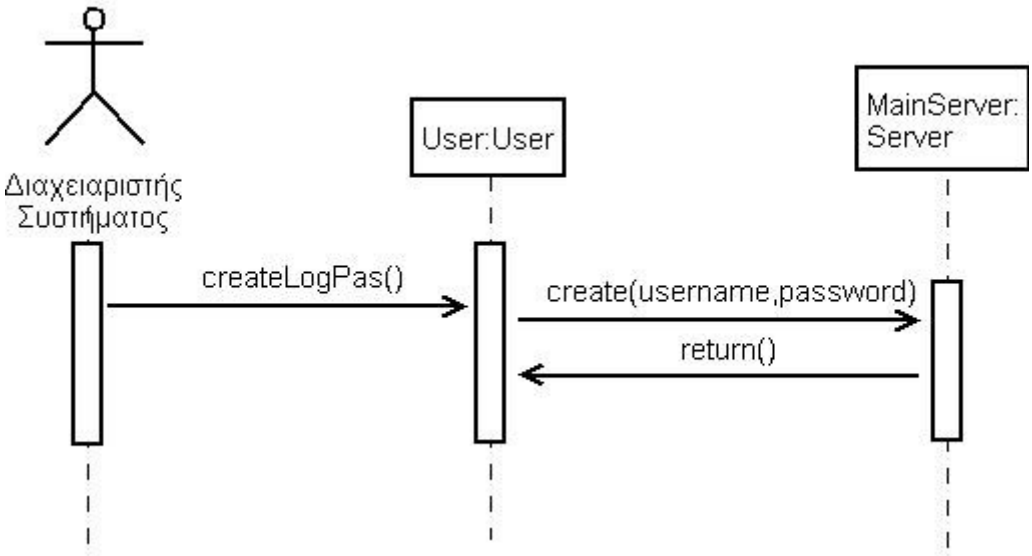
Εικόνα: Αναβάθμιση λογισμικού



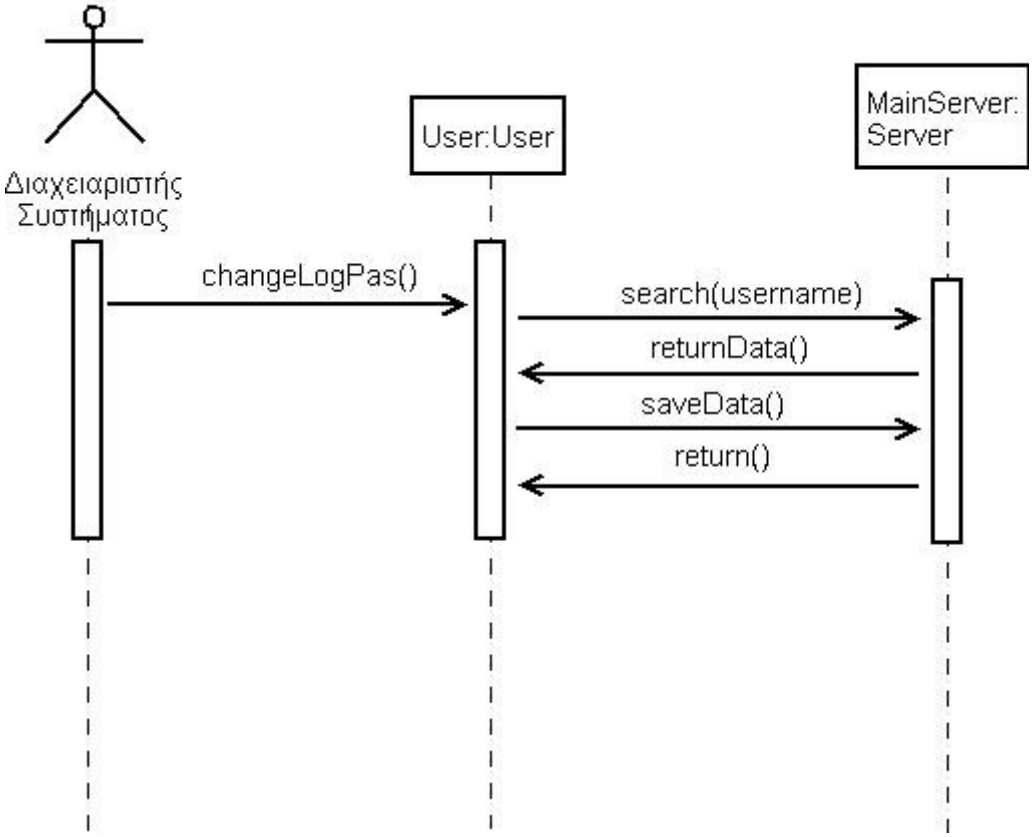
Εικόνα: Κατάργηση λογισμικού (με εγκατάσταση backup)



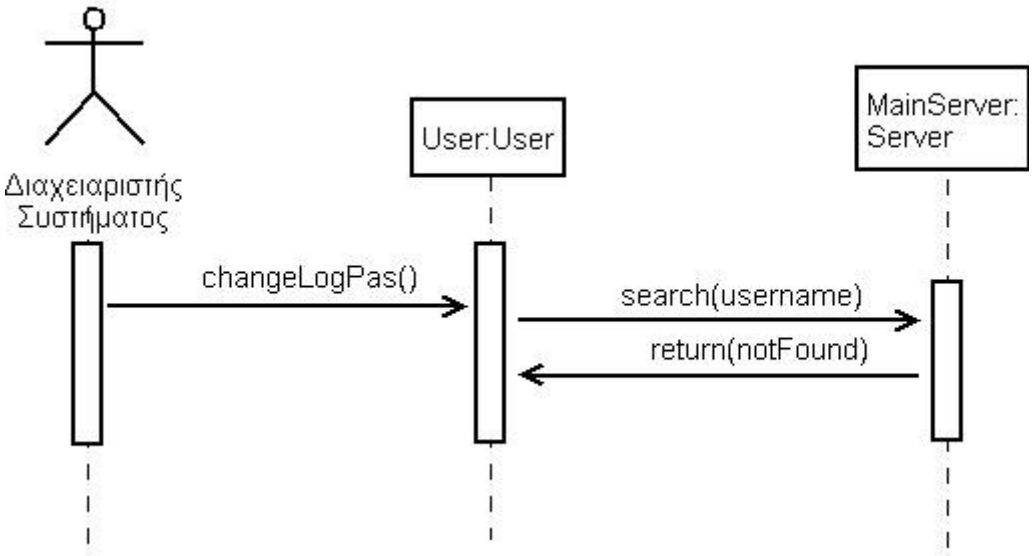
Εικόνα: Κατάργηση λογισμικού (χωρίς εγκατάσταση backup)



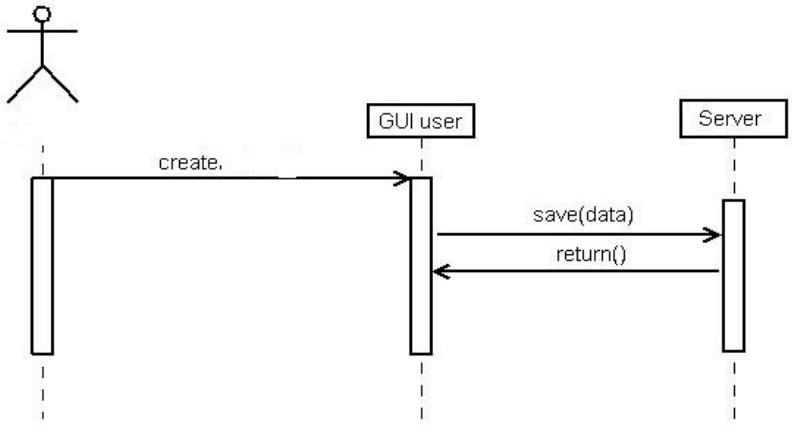
Εικόνα: Δημιουργία login/password



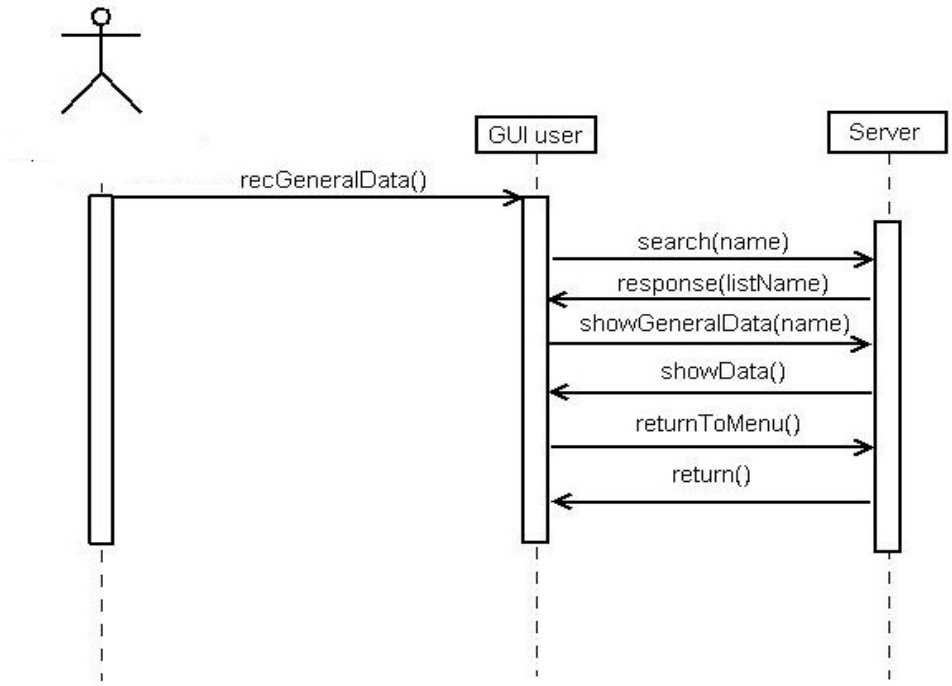
Εικόνα: Αλλαγή login/password (επιτυχία)



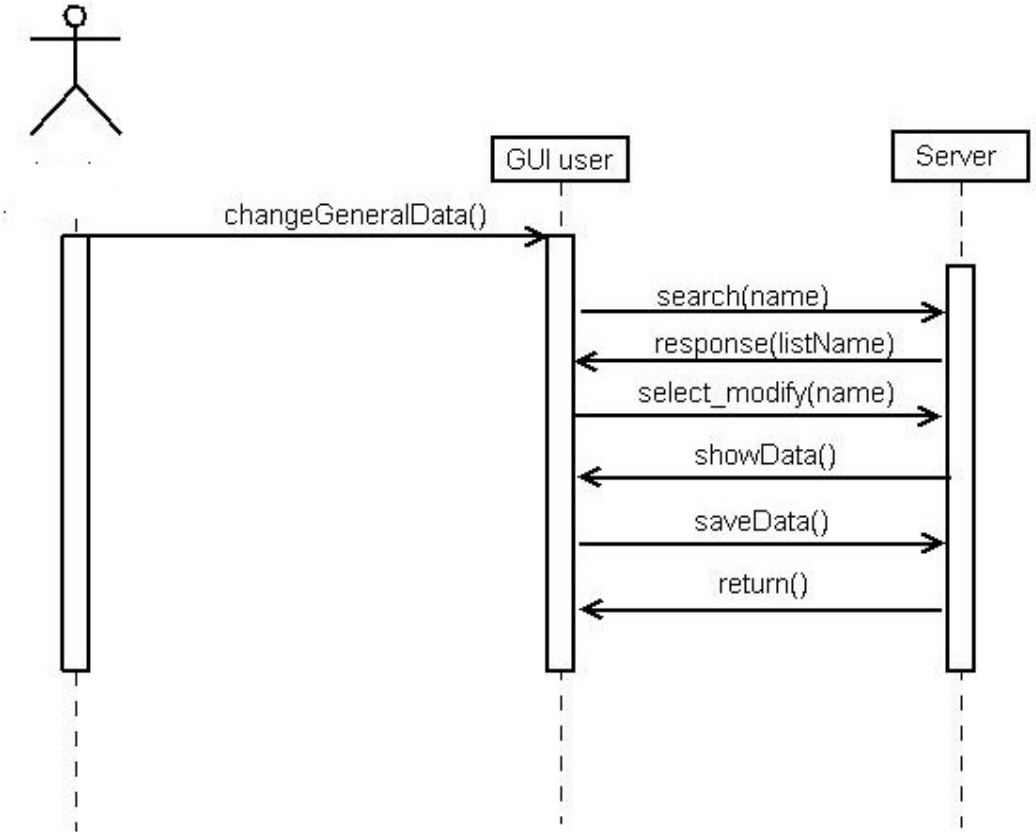
Εικόνα: Αλλαγή login/password (αποτυχία: username not found)



Εικόνα: Δημιουργία νέας εγγραφής

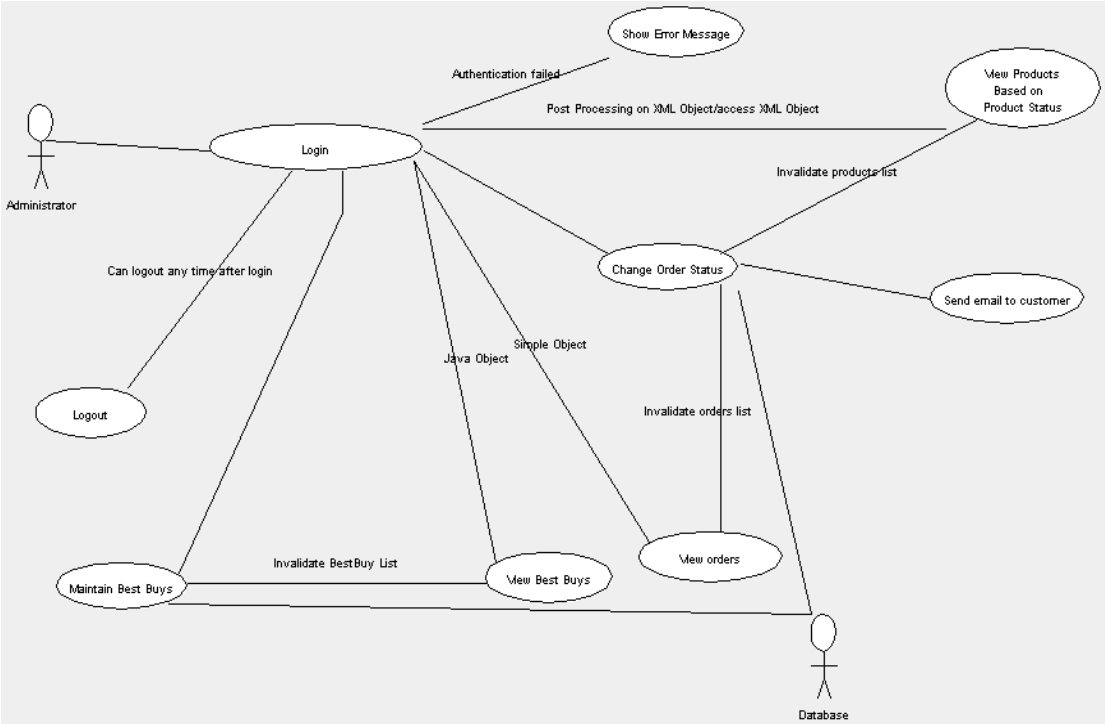


Εικόνα: Αναγνώριση στοιχείων πελάτη

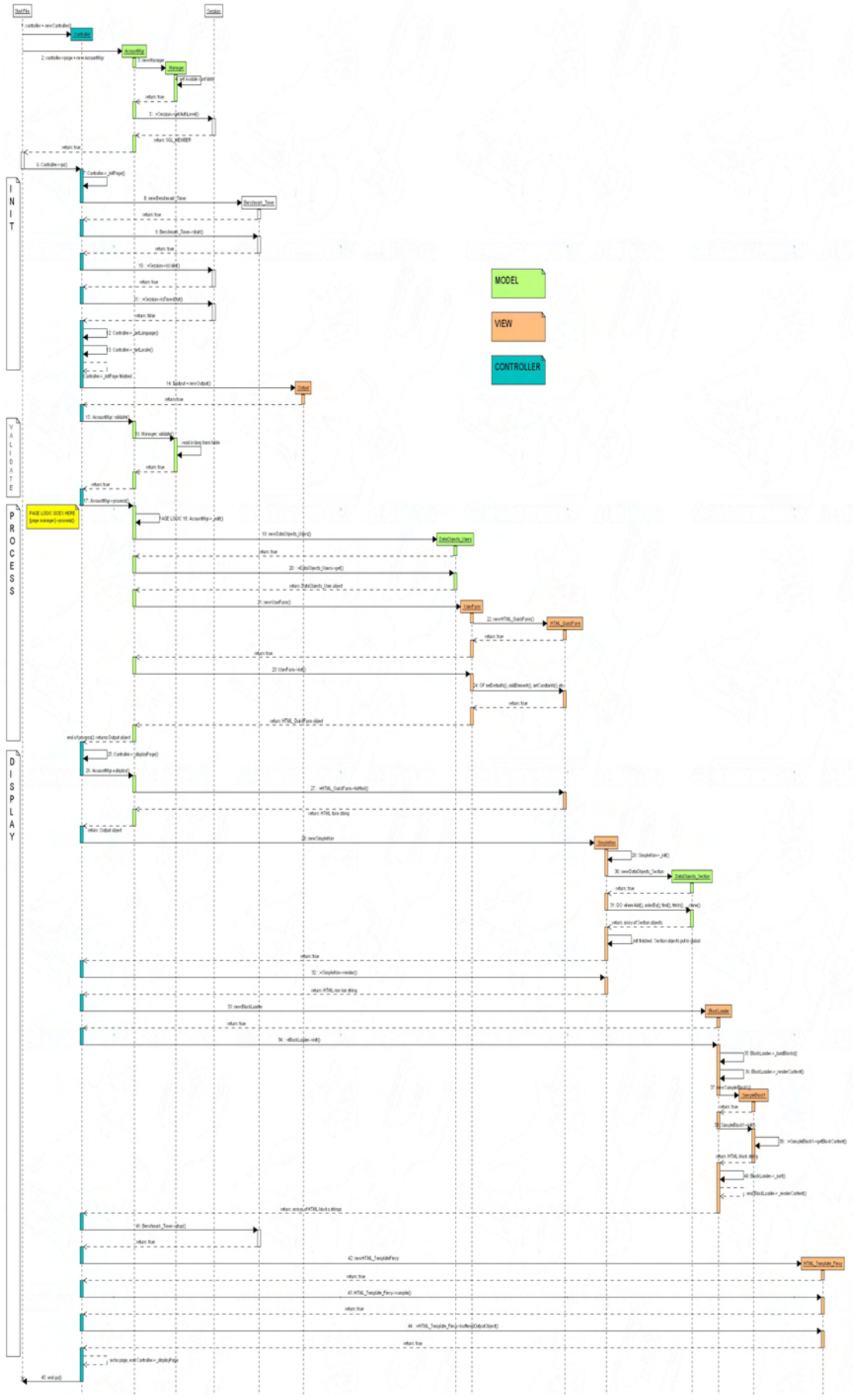


Εικόνα: Αλλαγή στοιχείων πελάτη

Γενικό διάγραμμα περιπτώσεων:

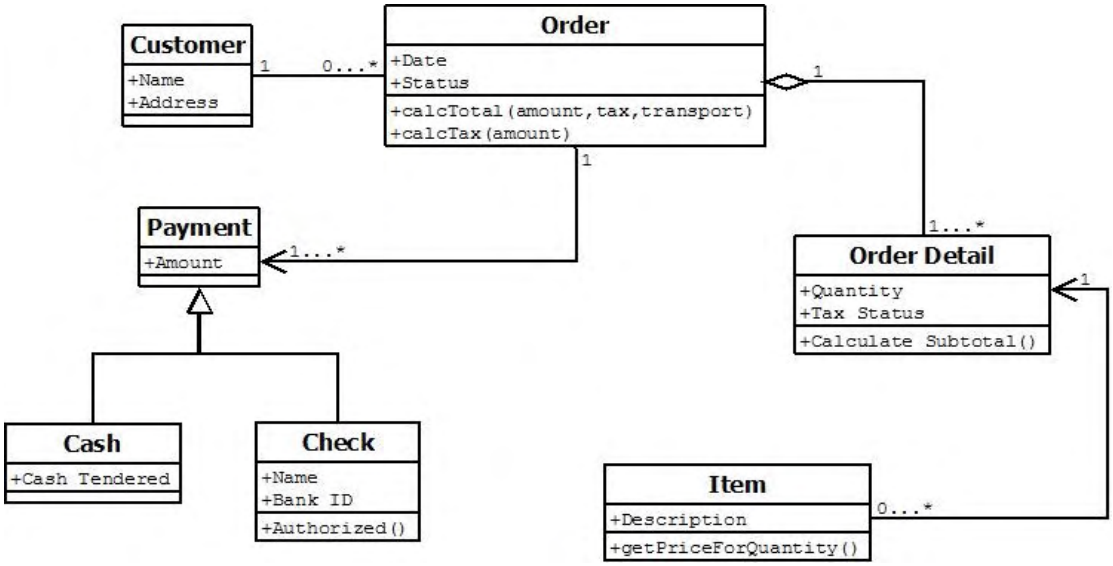


Γενικό διάγραμμα αλληλουχίας:

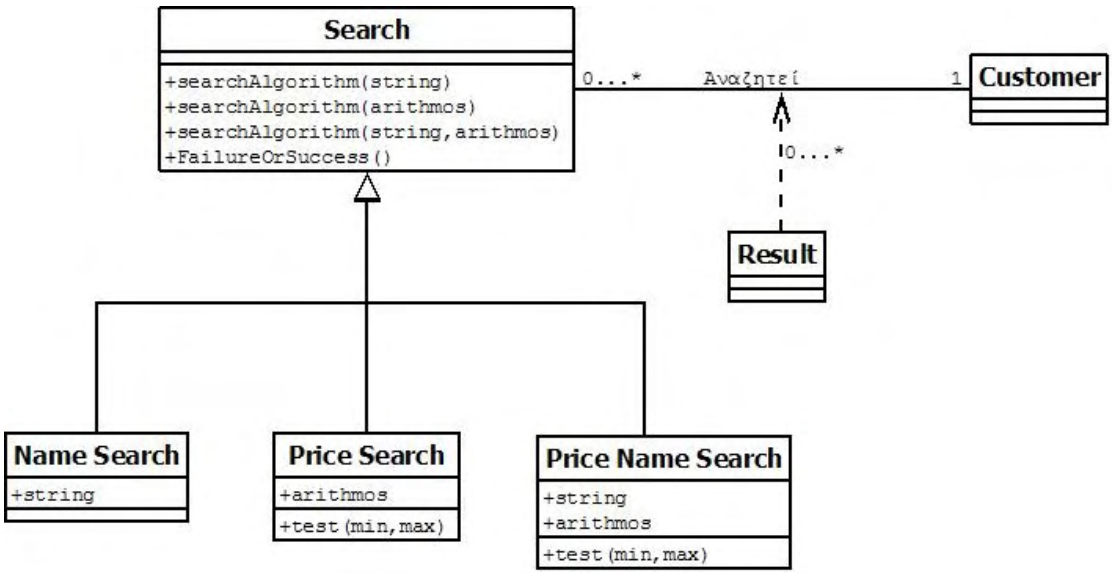


2.3.3 Διάγραμμα κατηγορίας (Class Diagrams)

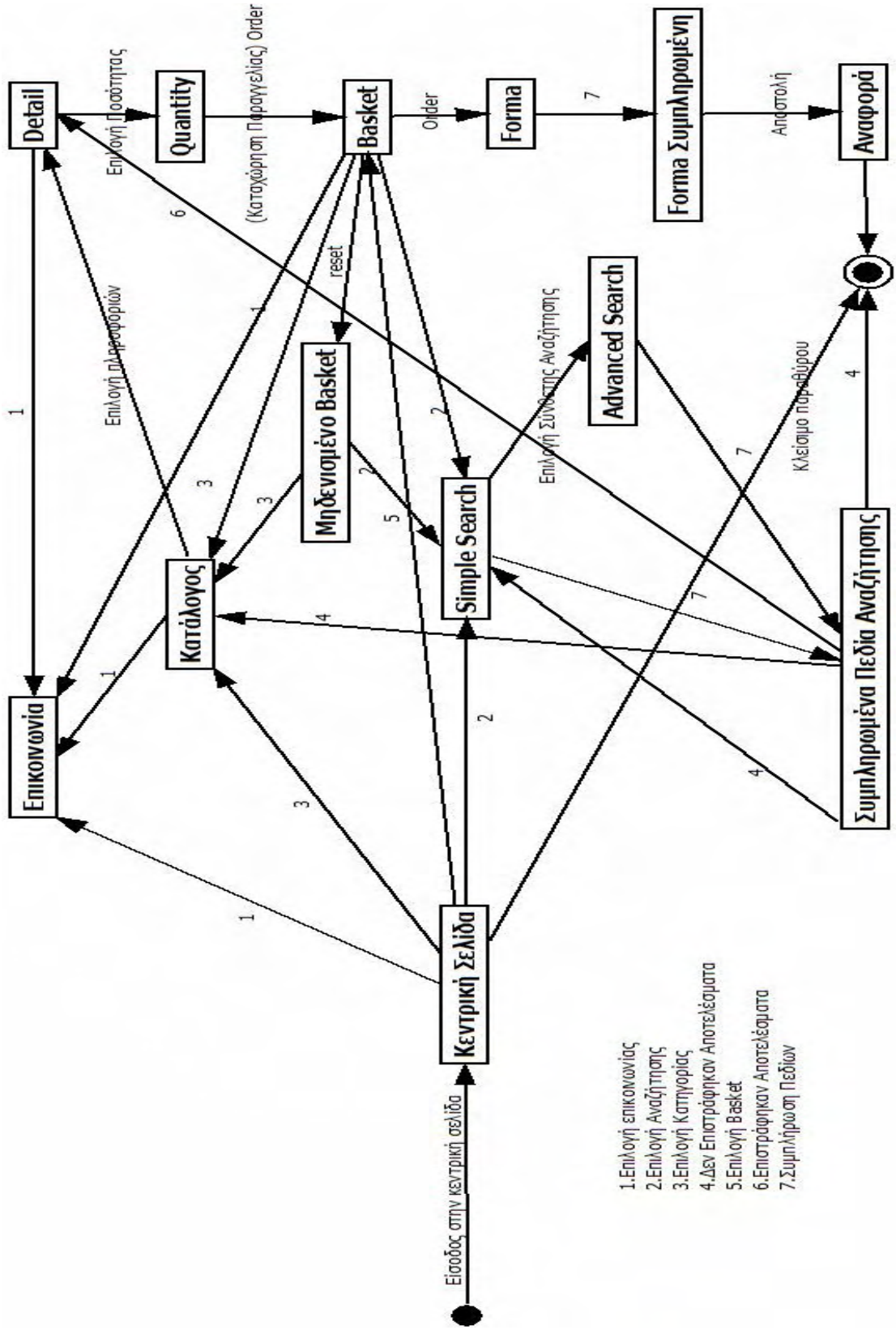
Λεπτομερής περιγραφή της παραγγελίας (order):



Λεπτομερής περιγραφή της αναζήτησης (search):



2.3.4 Δομή συστήματος



2.3.5 Σχέδιο δεδομένων

Τα δεδομένα των δίσκων που χρησιμοποιούνται σε ένα αρχείο βάσης δεδομένων με το όνομα “catalog.txt ”.

Υπάρχουν 9 διαφορετικοί τύποι εγγραφών στη βάση δεδομένων.

Περιγραφή: Η εγγραφή περιλαμβάνει πληροφορίες για το κάθε προϊόν. Για κάθε προϊόν έχουμε αποθήκευση σε ξεχωριστή εγγραφή.

Δομή:

Όνομα Πεδίου	**Τύπος**	**Μήκος**	**Περιγραφή**
Id	integer	4	Αύξων αριθμός
Category type	String χαρακτήρων	12	Κατηγορία προϊόντος
Available	integer	5	Διαθεσιμότητα προϊόντος
Label	String χαρακτήρων	64	Εκδότρια Εταιρία
NumProd	integer	3	Πλήθος products που περιέχονται, δυνατότητα επιλογής από 1 έως 6
ReleaseDate	String αριθμών με παύλες με τη μορφή yyyy-mm-dd	10	Ημερομηνία πρώτης έκδοσης
Description	String χαρακτήρων	100	Γενική περιγραφή
Price	integer	5	Τιμή τεμαχίου
Image	JPEG, JPG	500Kb max	Εικόνα του προϊόντος

Τα δεδομένα των διαχειριστών που χρησιμοποιούνται σε ένα αρχείο βάσης δεδομένων με το όνομα “admin.txt ”.

Υπάρχουν 2 διαφορετικοί τύποι εγγραφών στη βάση δεδομένων.

Περιγραφή: Η εγγραφή περιλαμβάνει πληροφορίες για τους διαχειριστές. Για κάθε διαχειριστή δημιουργείται ξεχωριστή εγγραφή.

Δομή:

Όνομα Πεδίου	**Τύπος**	**Μήκος**	**Περιγραφή**
Username	String χαρακτήρων	4	Όνομα χρήστη
Password	String χαρακτήρων	4	Κωδικός σύνδεσης

Κατάσταση αποθεμάτων του ηλεκτρονικού καταστήματος

Περιγραφή: Η δομή περιέχει πληροφορίες για την κατάσταση του ηλεκτρονικού καταστήματος (όριο τεμαχίων ενός προϊόντος προς παραγγελία ανάλογα με τη διαθεσιμότητα)

Δομή:

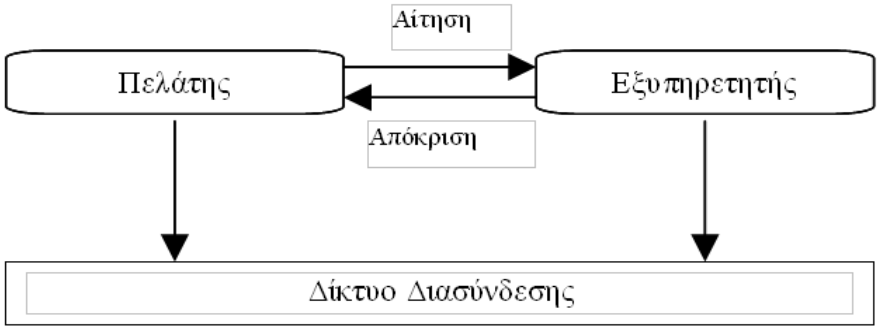
Όνομα Πεδίου	**Τύπος**	**Μήκος**	**Περιγραφή**
Available	integer	5	Διαθεσιμότητα προϊόντος

3. Σχεδίαση

3.1 Επιλογή Αρχιτεκτονικής

3.1.1 Αρχιτεκτονική client-server

Εφ'όσον οι μηχανές που συμμετέχουν στο σύστημα δεν έχουν κοινή μνήμη, οποιαδήποτε επικοινωνία μεταξύ τους βασίζεται σε ανταλλαγή μηνυμάτων μέσω του δικτύου. Για να είναι αποτελεσματική η επικοινωνία, ο αποστολέας και ο παραλήπτης πρέπει να συμφωνούν στη σημασία των πληροφοριών που ανταλλάσσονται μεταξύ τους και αυτό γίνεται σε διάφορα επίπεδα. Γι' αυτό το λόγο, η αρχιτεκτονική client-server είναι υποψήφια για το εν λόγω σύστημα.



Εικόνα: Μοντέλο πελάτη-εξυπηρετητή

Στο μοντέλο αυτό, το σύστημα δομείται ως ένα σύνολο διεργασιών (server) οι οποίες προσφέρουν υπηρεσίες στις διεργασίες του χρήστη (client). Οι διεργασίες και οι υπηρεσίες εκτελούνται σε διαφορετικές μηχανές πάνω από το ίδιο ή διαφορετικά λειτουργικά συστήματα, αρκεί να υπάρχει συμφωνία μεταξύ τους στην αναπαράσταση και την ερμηνεία των μηνυμάτων που ανταλλάσσονται. Οι υπηρεσίες που προσφέρονται στις διεργασίες του χρήστη χρησιμοποιούν ένα απλό πρωτόκολλο αίτησης/απάντησης (request/reply protocol) και σε αυτές περιλαμβάνονται και υπηρεσίες βάσεως δεδομένων.

Μία τέτοια αρχιτεκτονική είναι ιδιαιτέρως αξιόπιστη όταν εφαρμόζεται σε ένα ενσύρματο τοπικό δίκτυο καθώς η επικοινωνία πελάτη-εξυπηρετητή υλοποιείται χωρίς εγκατάσταση συνδέσεων. Το πλεονέκτημα είναι η απλότητα της επικοινωνίας : ο πελάτης στέλνει μία αίτηση, και παίρνει πίσω μία απάντηση, χωρίς να χρειάζεται να εγκατασταθεί ούτε να

τερματιστεί μία σύνδεση. Το μήνυμα απάντησης μπορεί να χρησιμοποιηθεί ως υπονοούμενο μήνυμα παραδοχής (acknowledgement message) της αίτησης.

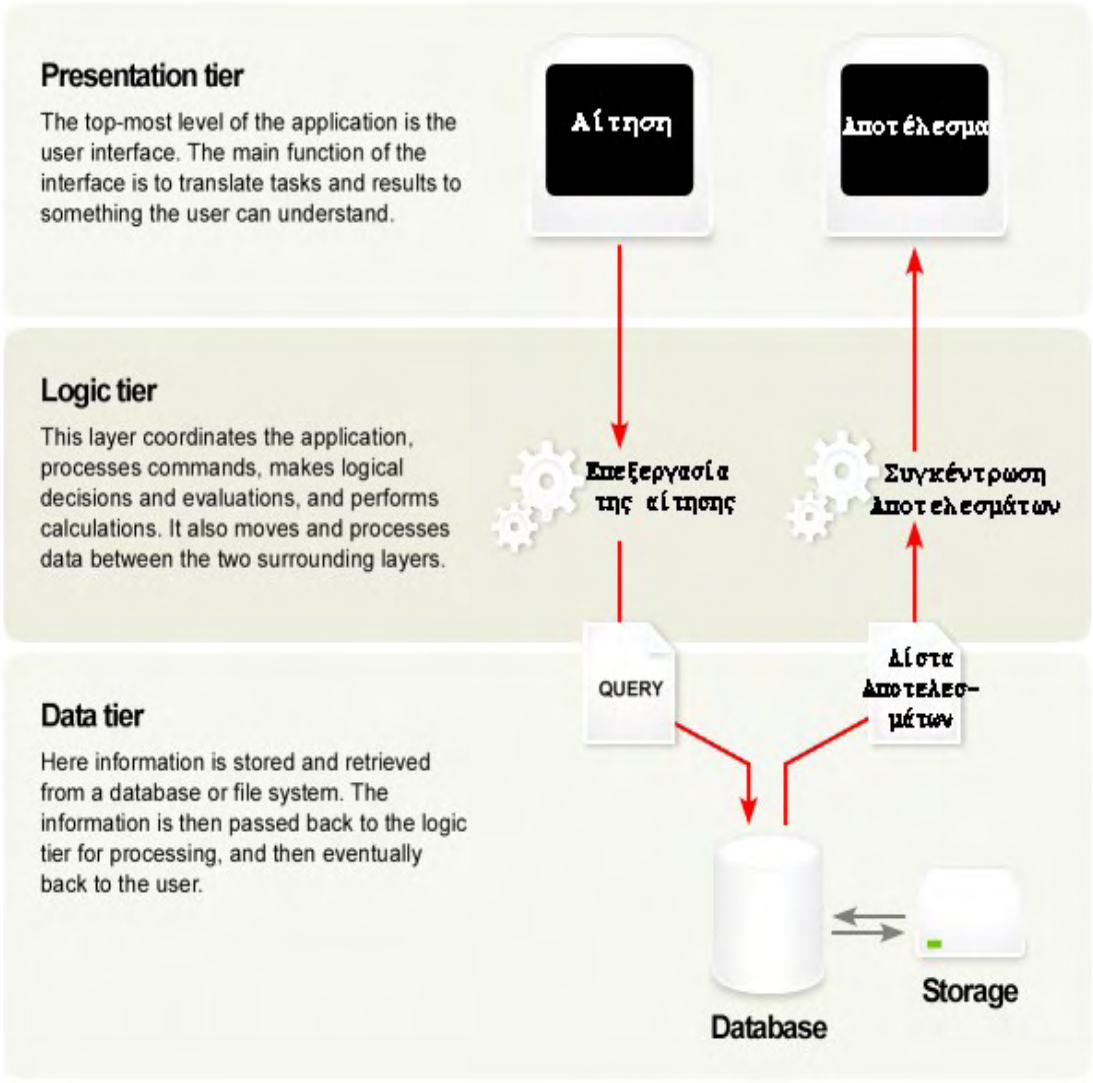
Η απλότητα του μοντέλου αυτού σημαίνει μικρή επιβάρυνση στο σύστημα, αφού απαιτούνται λιγότερα επίπεδα πρωτοκόλλων. Στην συγκεκριμένη περίπτωση, θα χρησιμοποιηθούν 4 επίπεδα πρωτοκόλλων διότι όλες οι μηχανές είναι ίδιου τύπου.

Ωστόσο ένα τέτοιο σύστημα, όταν εφαρμόζεται σε ασύρματο τοπικό δίκτυο, δεν είναι τόσο αξιόπιστο καθώς κάποια μηνύματα χάνονται ή παραμορφώνονται. Για να αυξηθεί η αξιοπιστία του πρωτοκόλλου, ο πελάτης μπορεί να αναμεταδίδει την αίτησή του, αν δεν λάβει την αντίστοιχη απάντηση μέσα σε κάποιο χρονικό διάστημα. Το πρόβλημα είναι ότι τότε δεν μπορούμε να διακρίνουμε κατά πόσο η έλλειψη απάντησης οφείλεται σε απώλεια της αίτησης ή της απόκρισης. Αν χάθηκε η αίτηση, η εκ νέου μετάδοσή της δεν δημιουργεί κανένα πρόβλημα. Αν, όμως χάθηκε η απόκριση, ο εξυπηρετητής θα έχει ήδη εκτελέσει κάποιες ενέργειες για να εξυπηρετήσει το αίτημα. Η αναμετάδοση της αίτησης μπορεί έτσι να οδηγήσει σε διπλή εκτέλεση των ενεργειών. Το κατά πόσο είναι αυτό προβληματικό εξαρτάται από την αίτηση. Αν η αίτηση αφορά την ανάγνωση στοιχείων, δεν έχουμε πρόβλημα. Αν, όμως, αφορά στην προσθήκη στοιχείων στο τέλος ενός αρχείου, η επανάληψη της αίτησης οδηγεί σε ανεπιθύμητη αλλαγή του αρχείου. Τελικά η μόνη εναλλακτική λύση για το προαναφερθέν πρόβλημα είναι η αύξηση της πολυπλοκότητας του πρωτοκόλλου πελάτη-εξυπηρετητή.

3.1.2 Αρχιτεκτονική πελάτη-εξυπηρετητή τριών επιπέδων (3-Tiers)

Σε αυτή την αρχιτεκτονική, οι πελάτες και οι εξυπηρετητές δεν είναι ξεκάθαρα ξεχωρισμένοι. Αυτό αιτιολογείται με το γεγονός ότι ένας εξυπηρετητής βάσεων δεδομένων μπορεί να είναι ταυτόχρονα και πελάτης ενός εξυπηρετητή αρχείων στον οποίο αποθηκεύονται οι πίνακες της βάσης δεδομένων, λειτουργώντας ως μεσάζων ανάμεσα στους πελάτες της βάσης και τους εξυπηρετητές αρχείων. Ο εξυπηρετητής βάσεων δεδομένων παρέχει την επεξεργασία των πρωτογενών δεδομένων, προκειμένου να ικανοποιούνται τα αιτήματα των πελατών. Στην πράξη η οργάνωση πελατών και εξυπηρετητών σε τρία επίπεδα είναι η πλέον συνηθισμένη σε συστήματα διαχείρισης βάσεων δεδομένων.

Το πρώτο επίπεδο παρέχει την διεπαφή με το χρήστη (user interface), το δεύτερο την επεξεργασία των δεδομένων (data processing) και το τρίτο την αποθήκευση των δεδομένων (data storage).



Εικόνα: Μοντέλο πελάτη-εξυπηρετητή τριών επιπέδων

Το επίπεδο διεπαφής έχει καθαρά ρόλο πελάτη, και περιέχει τις εφαρμογές που επιτρέπουν στους χρήστες να αλληλεπιδρούν με την βάση δεδομένων. Για το σύστημά μας, θα χρειαστούμε υποστήριξη μίας γραφικής διεπαφής με μενού και πλαίσια διαλόγου. Η εφαρμογή διαχειρίζεται την οθόνη και τις συσκευές εισόδου του χρήστη (πληκτρολόγιο και ποντίκι) και επικοινωνεί με το επόμενο επίπεδο μόνο για να στείλει τις επερωτήσεις του χρήστη και για να παρουσιάσει στην οθόνη τα στοιχεία που επιστρέφονται από τη βάση.

Το επίπεδο επεξεργασίας παρέχει μία γέφυρα ανάμεσα στη διεπαφή με το χρήση και την αποθήκευση δεδομένων, η οποία μπορεί να παρέχεται από εξυπηρετητές αρχείων ή εξυπηρετητές βάσεων δεδομένων. Τα καθήκοντα του επιπέδου επεξεργασίας εξαρτώνται

από την υλοποιούμενη εφαρμογή, αλλά ο στόχος του είναι να αντλεί στοιχεία από πληροφοριακά συστήματα γενικής χρήσεως και να τα επεξεργάζεται, για να ικανοποιεί τις απαιτήσεις της εφαρμογής.

Τέλος το επίπεδο αποθήκευσης έχει καθαρά ρόλο εξυπηρετητή, αφού περιέχει τις εφαρμογές που διαχειρίζονται τα δεδομένα στα οποία τελικά αποκτά πρόσβαση ο χρήστης. Το επίπεδο αυτό είναι ανεξάρτητο από το επίπεδο επεξεργασίας, με την έννοια ότι τα δεδομένα που αποθηκεύονται είναι διαθέσιμα ακόμη και όταν η εφαρμογή επεξεργασίας δε χρησιμοποιείται. Έτσι το ίδιο επίπεδο αποθήκευσης μπορεί να χρησιμοποιηθεί μα πολλά διαφορετικά επίπεδα επεξεργασίας, πράγμα που συμβαίνει πολύ συχνά στην πράξη. Το επίπεδο αποθήκευσης μπορεί να είναι είτε ένα σύστημα αρχείων είτε μία ολοκληρωμένη βάση δεδομένων.

3.1.3 Σύγκριση αρχιτεκτονικών

Η σύγκριση θα πραγματοποιηθεί βάσει των παρακάτω κριτηρίων:

- Ελεγξιμότητα / προσομοίωση
- Συντηρησιμότητα
- Τμηματικότητα
- Ασφάλεια
- Ωριμότητα της τεχνικής
- Κλημακοσιμότητα

Ελεγξιμότητα / προσομοίωση : Ο κώδικας μπορεί να ελεγχθεί καλύτερα στη δεύτερη αρχιτεκτονική εφ'όσον υπάρχει ξεχωριστός server και server βάσης δεδομένων και έτσι ελέγχεται καλύτερα ο επιμερισμένος κώδικας παρά ο ενιαίος.

Συντηρησιμότητα : Στην $2^{η}$ αρχιτεκτονική δεν υπάρχει τόσο μεγάλη σύζευξη και άρα το σύστημα συντηρείται πιο εύκολα.

Τμηματικότητα : Στην $2^{η}$ αρχιτεκτονική το σύστημα μπορεί να αποσυντεθεί σε μικρότερα τμήματα τα οποία γίνονται ευκολότερα κατανοητά και έτσι οι αλλαγές σε μικρότερα τμήματα του κώδικα δεν προκαλούν μεγάλες αλλαγές και σε άλλα τμήματά του.

Ασφάλεια : Σε περίπτωση που ο κώδικας δεν συμπεριφέρεται σωστά κάτω από απρόσμενες συνθήκες, μπορούν να ελεγχθούν καλύτερα οι συνέπειες της συμπεριφοράς αυτής στην 3-tiers αρχιτεκτονική γιατί το σύστημα είναι διαιρεμένο σε μικρότερα τμήματα.

Ωριμότητα της τεχνικής : Η 2η τεχνική είναι πιο διαδεδομένη καθώς δείχνει κατανόηση στη δυσκολία που υπάρχει κατά την υλοποίηση ενός ενιαίου συστήματος σε αντίθεση με την ευελιξία που παρουσιάζεται σε ένα τμηματικοποιημένο σύστημα.

Κλημακοσιμότητα : Στην αρχιτεκτονική 3-tiers, το σύστημα μπορεί να πάρει οποιαδήποτε μορφή ως προς την έκταση που θα αποφασίσει ο τεχνολόγος ή ο πελάτης αφού το μόνο που χρειάζεται είναι να προστεθεί κάποιος επιπλέον server.

ια την κατασκευή πίνακα σύγκρισης χρησιμοποιούμε κλίμακα από 1-5.

Κριτήριο	Προτεραιότητα	Client-Server	3-tiers
Ελεγξιμότητα / προσομοίωση	5	2	4
Συντηρησιμότητα	3	1	3
Τμηματικότητα	5	2	4
Ασφάλεια	5	3	4
Ωριμότητα της τεχνικής	3	3	4
Κλημακοσιμότητα	4	3	4

Συνοψίζοντας :

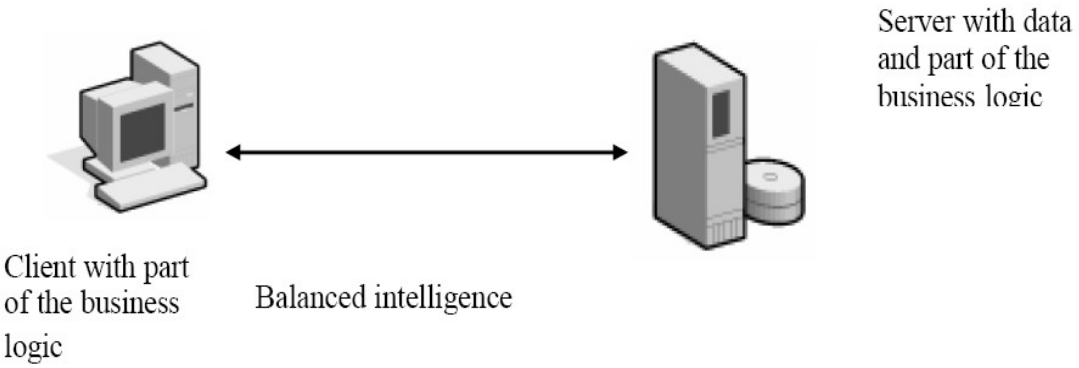
Client-Server : 5*2+3*1+5*2+5*3+3*3+4*3 = 59

3-tiers : 5*4+3*3+5*4+5*4+3*4+4*4 = 97

Επομένως, επιλέγεται και μαθηματικά πλέον η 3-Tiers αρχιτεκτονική.

3.2 Αποσύνθεση συστήματος

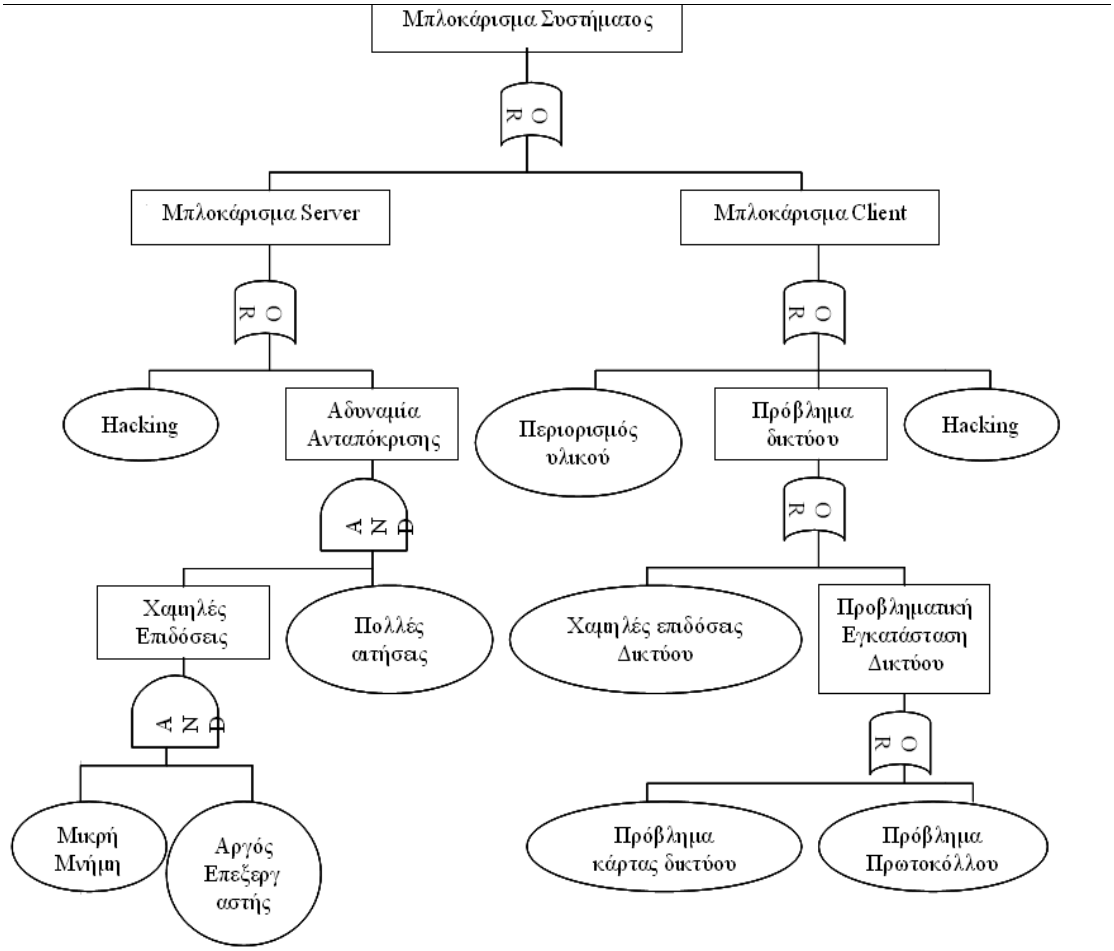
Σύμφωνα με την απαιτούμενη δομή και τα δεδομένα του συστήματος θα χρησιμοποιήσουμε τη δομή balanced intelligence. Σύμφωνα με αυτή, οι λειτουργίες που υποστηρίζονται μοιράζονται μεταξύ server και client έτσι ώστε να μειωθεί ο φόρτος εργασίας και για τους δύο.

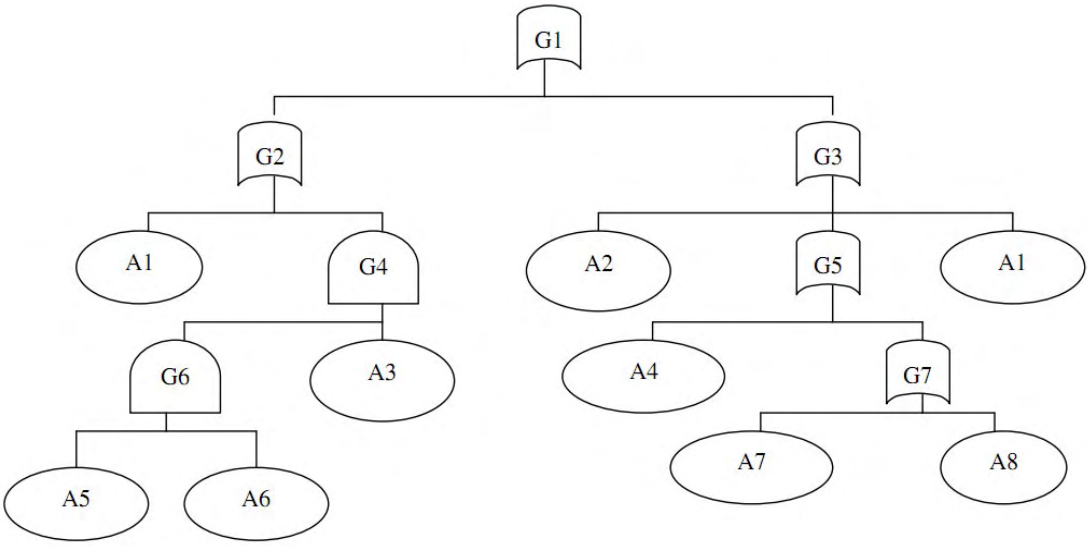


Αποσύνθεση που προσανατολίζεται στα γεγονότα: θα γίνει με βάση το διάγραμμα περιπτώσεων χρήσης που παρουσιάστηκε στον ορισμό των απιατήσεων.

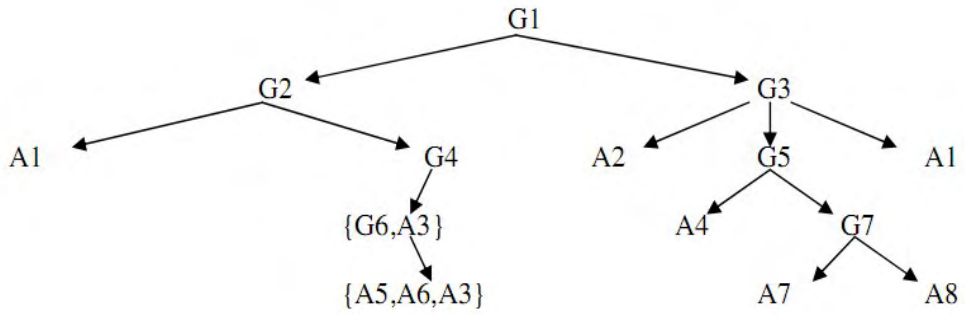
3.3 Σύνολο αποκοπής

3.3.1 Δέντρο σφαλμάτων





3.3.2 Υπολογισμός συνόλου αποκοπής



Με βάση το παραπάνω δέντρο, βρίσκουμε το σύνολο αποκοπής :

{A1},{A3,A5,A6},{A2},{A4},{A7},{A8}

4. Java EE και JSF

Ο κορμός του ηλεκτρονικού καταστήματος, δηλαδή ο σκελετός του προγράμματος είναι γραμμένος σε Java EE. Η Java Platform, Enterprise Edition ή αλλιώς Java EE είναι μια ευρέως διαδεδομένη πλατφόρμα για προγραμματισμό Server που στηρίζεται πάνω στην γλώσσα προγραμματισμού Java. Χρησιμοποίησα αυτή την τεχνολογία για το γεγονός ότι προσθέτει βιβλιοθήκες οι οποίες παρέχουν λειτουργικότητα ώστε να αναπτυχθεί διαδικτυακό, πολυμερές λογισμικό σε Java, βασισμένο σε αυτοτελή μέρη που τρέχουν σε application servers.

Επιπλέον χρησιμοποίησα JavaServer Faces. JavaServer Faces (JSF) είναι ένα Java-based Web application framework με σκοπό την απλούστευση της ανάπτυξης των διεπαφών χρήστη για εφαρμογές Java EE. Σε αντίθεση με το MVC (Model-View-Controller) web frameworks, το οποίο είναι καθοδηγούμενο από την αίτηση, το JSF χρησιμοποιεί μια component-based προσέγγιση.

Η κατάσταση του User-Interface, διεπαφή χρήστη, components σώζεται, όταν ζητήσει ο πελάτης μια νέα σελίδα και αποκαθίσταται όταν η επιστρέψει η απάντηση.

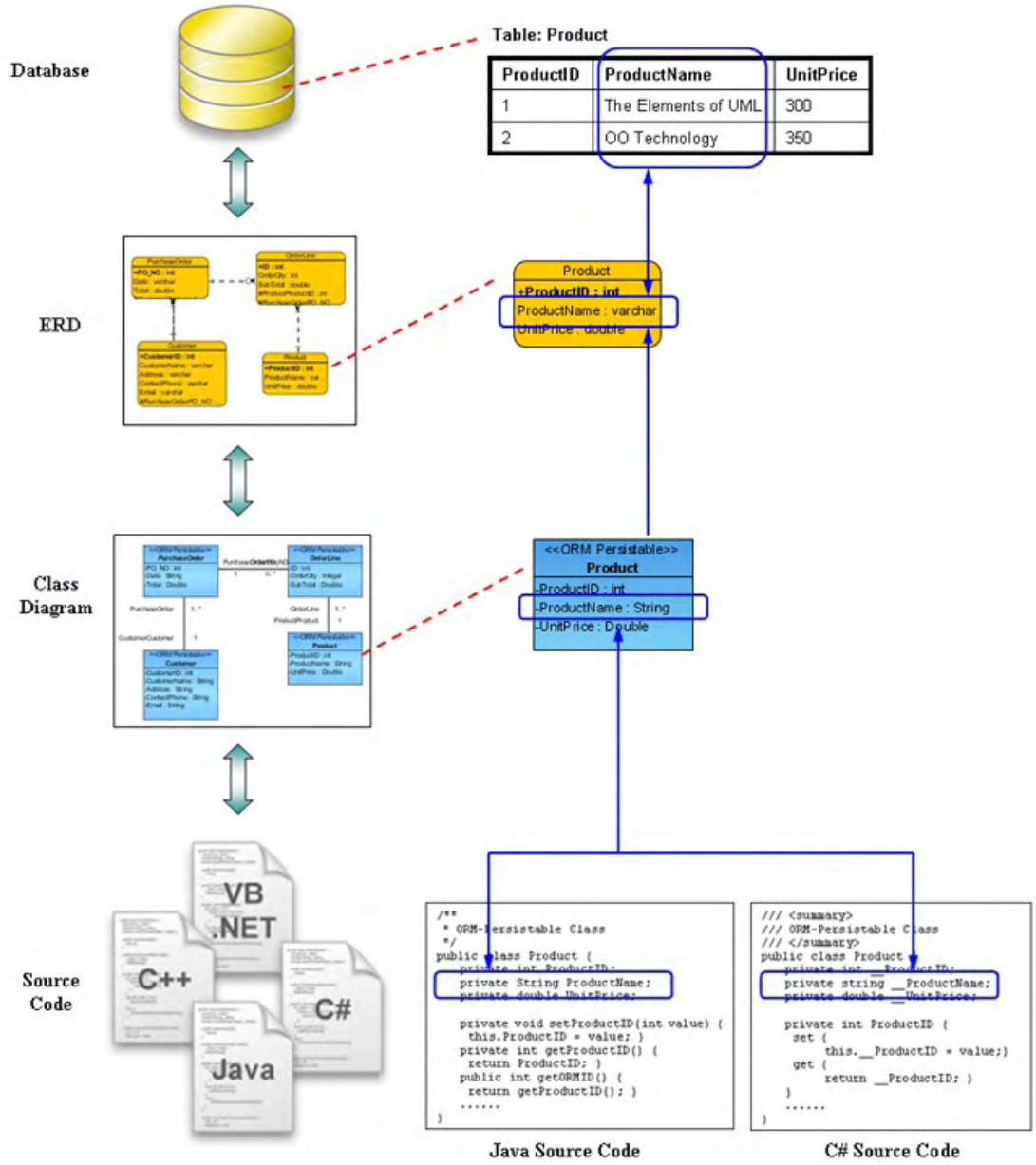
Η JSF χρησιμοποιεί JavaServer Pages (JSP) για την τεχνολογία απεικόνισης του, αλλά μπορεί να φιλοξενήσει και άλλες τεχνολογίες (όπως το XUL και Facelets).

JSF περιέχει:

- A set of APIs for representing user interface (UI) components and managing their state, handling events and input validation, converting values, defining page navigation, and supporting internationalization and accessibility
- A default set of UI components
- Two JavaServer Pages (JSP) custom tag libraries for expressing a JavaServer Faces interface within a JSP page.
- A server-side event model
- State management
- Managed Beans (JavaBeans created with dependency injection)
- Unified Expression Language for both JSP 2.0 and JSF 1.2

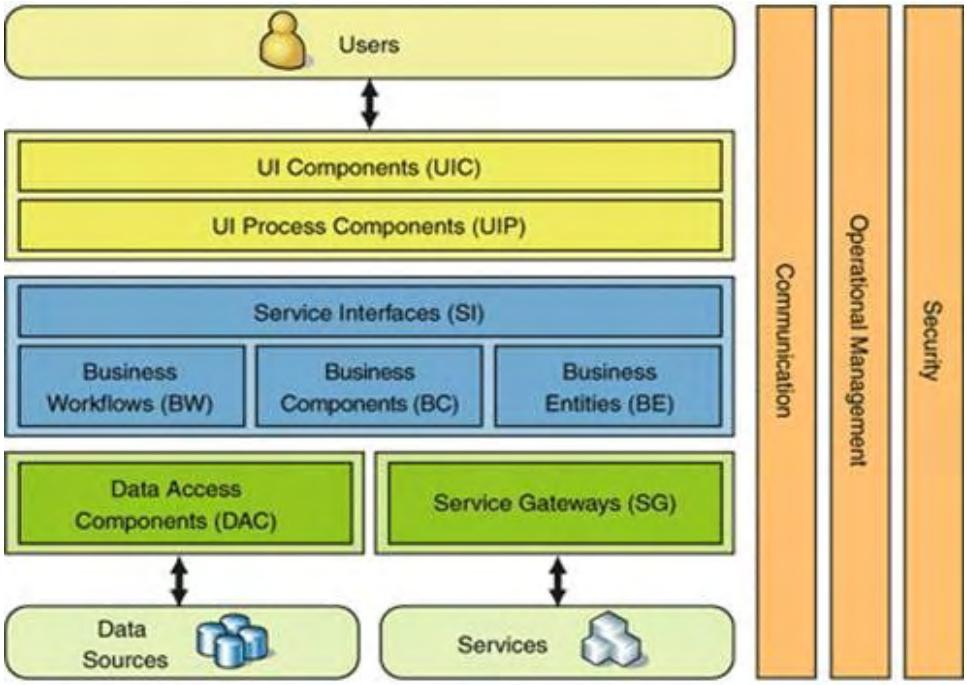
5. Μηχανισμοί αντιστοίχισης σχεσιακού μοντέλου βάσης

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).

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 τα επιστρέφει στον προγραμματιστή ως αντικείμενα της εφαρμογής. Είναι δηλαδή ένα ενδιάμεσο επίπεδο μεταξύ εφαρμογής και βάσης δεδομένων.



Επιλέγοντας αυτή την τεχνολογία υπάρχει μεγάλη 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. Με τη χρήση του γράφονται σημαντικά λιγότερες γραμμές κώδικα και ο κώδικας είναι πιο κατανοητός και καλογραμμένος.

- Αυτό κάνει την συντήρηση της εφαρμογής ευκολότερη.
- Υπάρχει ανεξαρτησία από τη βάση δεδομένων, με τη συμβατότητα του 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.

Για παράδειγμά υποθέτουμε πως έχουμε ένα μοντέλο στο οποίο ένας άνθρωπος (Person) μπορεί να διαθέτει κανένα ή περισσότερα οχήματα (Vehicle). Το αντίστοιχο διάγραμμα που δείχνει τα αντικείμενα του μοντέλου μας είναι το παρακάτω:

Η κατάσταση αυτή μπορεί να μεταφραστεί σε μια σχεσιακή βάση δεδομένων χρησιμοποιώντας δύο πίνακες (PERSON και VEHICLE) με τα αντίστοιχα πεδία όπως φαίνεται παρακάτω:

Προσέξτε ότι το πεδίο PERSON_ID του πίνακα VEHICLE είναι εξωτερικό κλειδί (foreign key) και συνδέει το PERSON με τα VEHICLEs τα οποία κατέχει. Στην εφαρμογή μας πρέπει να κατασκευάσουμε δύο αντικείμενα τα Person και Vehicle:

Αρχείο Person.java

```
package example;
class Person{
    private long id;
    private String name;
    private boolean sex;
    private int age;
    private List vehicles;
    public Person(){}
    public String getName(){
        return this.name;
    }
    public void setName(String name){
        this.name=name;
    }
    //το ίδιο και για τα άλλα τέσσερα χαρακτηριστικά
```

```
    …
}
```

Αρχείο Vehicle.java

```
package example;
class Vechicle{
    private long id;
    private String registration; //πινακίδα
    private int engine; //κυβισμός
    private String make; //μάρκα
    public Vehicle(){}
    public String getRegistration(){
        return this.registration;
    }
    public void setRegistration(String registration){
        this.registration=registration;
    }
    //το ίδιο και για τα άλλα τρία χαρακτηριστικά
     …
}
```

Χρησιμοποιώντας Hibernate πρέπει να κατασκευάσουμε τα XML αρχεία που περιγράφουν την αντιστοιχία μεταξύ των αντικειμένων και των πινάκων της βάσης δεδομένων:

Αρχείο engine/Person.hbm.xml

```
<xml version = "1.0"?>
 <!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD//EN"
"http://hibernate.sourceforge.net/hibernate-mapping-2.0.dtd">
    <class name="example.Person" table="PERSON">
        <id name="id" column="PERSON_ID" type="long">
            <generator class="native">
        </id>
        <property name="name" column="P_NAME" type="string"/>
        <property name="sex" column="P_SEX" type="boolean"/>
        <property name="age" column="P_AGE" type="integer"/>
        <set name="vehicles" inverse="true" cascade="save-update">
```

```
            <key column="PERSON_ID" />
            <one-to-many class="example.Vehicle"/>
        </set>
    </class>
</hibernate-mapping>
```

Αρχείο engine/Vehicle.hbm.xml

```
<xml version = "1.0"?>
<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping
DTD//EN"
"http://hibernate.sourceforge.net/hibernate-mapping-2.0.dtd">
    <class name="example.Vehicle" table="VEHICLE">
        <id name="id" column="VEHICLE_ID" type="long">
            <generator class="native">
        </id>
              <property name="registration" column="V_REGISTRATION"
type="string"/>
        <property name="engine" column="V_ENGINE" type="integer"/>
        <property name="make" column="V_MAKE" type="string"/>
                    <many-to one name="person" column="PERSON_ID"
class="example.Person" not-null="true"/>
    </class>
</hibernate-mapping>
```

Επίσης πρέπει να ορίσουμε ένα αρχείο με τα χαρακτηριστικά της βάσης δεδομένων που χρησιμοποιούμε. Για παράδειγμα, αν χρησιμοποιούμε PostgreSQL και η βάση μας λέγεται myBD τότε το αρχείο μας περιέχει τα παρακάτω:

Αρχείο hibernate.properties

```
hibernate.connection.driver_class= org.postgresql.Driver
hibernate.connection.url=jdbc:postgresql://localhost/myDB
hibernate.connection.username = user
hibernate.connection.password = pass
hibernate.dialect = net.sf.hibernate.dialect.PostgreSQLDialect
```

Γράφοντας τα παραπάνω έχουμε κάνει τα εξής:

- Έχουμε αντιστοιχίσει τα αντικείμενα Person και Vehicle με τους πίνακες PERSON και VEHICLE της βάσης δεδομένων (με το class των XML αρχείων).

- Έχουμε ορίσει τα πρωτεύοντα κλειδιά (με το id των XML αρχείων).
- Έχουμε αντιστοιχίσει τα χαρακτηριστικά των αντικειμένων (name, sex, age, registration, engine, make) με τις αντίστοιχες στήλες των σχεσιακών πινάκων (P_NAME, P_SEX, P_AGE, V_REGISTRATION, V_ENGINE, V_MAKE) (με το property των XML αρχείων).
- Έχουμε ορίσει τη συσχέτιση (association) μεταξύ των δύο πινάκων (με τα many-to-one, one-to-many και set των XML αρχείων).
- Έχουμε ορίσει τη βάση δεδομένων που θα χρησιμοποιήσει το Hibernate (το αρχείο αυτό τοποθετείται στο classpath της εφαρμογής μας και το Hibernate το βρίσκει αυτόματα).

Έχοντας ορίσει όλα αυτά μπορούμε να αρχίσουμε το Hibernate και να αναφερθούμε στα αντικείμενά μας όπως περιγράφεται στη συνέχεια:

```
//αρχίζουμε το Hibernate
Configuration cfg=new Configuration();
cfg.addResource("engine/Person.hbm.xml");
cfg.addResource("engine/Vehicle.hbm.xml");
SessionFactory session=new cfg.buildSessionFactory();

//δουλεύουμε με τα αντικείμενα
//αποθήκευση αντικειμένων
Session session=getSessionFactory().openSession();
Transaction tx=session.beginTransaction();
Vehicle v=new Vehicle();
v.setRegistration("XXX-1234");
v.setEngine(2000);
v.setMake("Nissan");
Person p=new Person();
p.setName("Nick");
p.setSex(true); //άνδρας
p.setAge(29);
p.getVehicles().add(v);
session.save(p);
tx.commit();
session.close();

//αναζήτηση αντικειμένων
Session newSession=getSessionFactory().openSession();
Transaction newTx=newSession.beginTransaction();
Person p2 = (Person) newSession.load(Person.class, new Long(1));
```

```
List vehicles=p2.getVehicles();
System.out.println("Number of Vehicles: "+vehicles.size());
newTx.commit();
newSession.close();
```

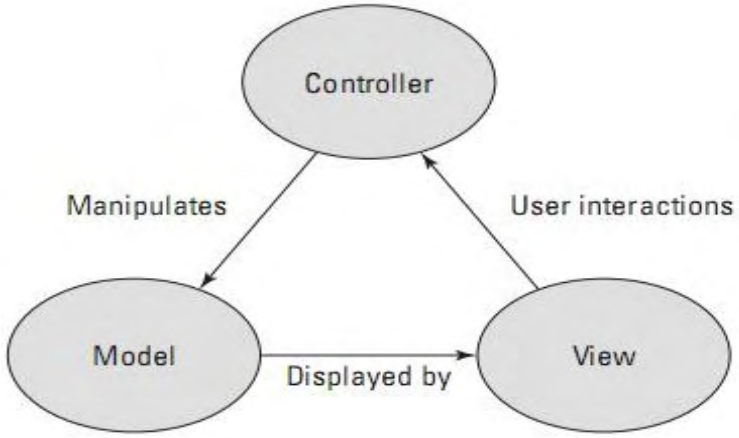
Στο παράδειγμα αυτό αποθηκεύσαμε στη βάση δεδομένων, στους αντίστοιχους πίνακες, τα αντικείμενά μας p και v και αναζητήσαμε το person και τα vehicles που έχει.

Παρατηρήστε ότι σε όλη την έκταση του παραδείγματος δε χρησιμοποιήσαμε καθόλου SQL κώδικα, πράγμα που επιτρέπει το διαχωρισμό του business logic της εφαρμογής απο τη βάση δεδομένων. Ο προγραμματιστής μπορεί έτσι να αφοσιωθεί στην κωδικοποίηση της εφαρμογής χωρίς να συνδέει στενά τον κώδικα της εφαρμογής με τον κώδικα που χρειάζεται για την τροποποίηση των δεδομένων στη βάση.

Επίσης, οι αλλαγές στη βάση δεδομένων είναι ευκολότερο να αντιμετωπιστούν στα XML αρχεία των αντιστοιχίσεων παρά οπουδήποτε μέσα στον κώδικα. Ακόμα, χρειάζονται πολύ λιγότερες γραμμές κώδικα για τις βασικές τροποποιήσεις των δεδομένων (CRUD – Create Read Update Delete) στη βάση δεδομένων. Αυτά, σε μεγάλης έκτασης εφαρμογές, αποδεικνύονται πολύτιμα τόσο όσον αφορά το χρόνο ανάπτυξης της εφαρμογής όσο και στην αποσφαλμάτωση του κώδικα κατά την ανάπτυξη της εφαρμογής.

6. Strus

Struts είναι ένα MVC (Model-View-Controller) framework το οποίο παρέχει ένα page-at-a-time MVC και δεν έχει ένα μοντέλο component όπως είναι αυτό του JSF. Το MVC είναι ένα αρχιτεκτονικό πρότυπο για τη διάρθρωση της ανάπτυξης λογισμικού σε τρεις μονάδες: Μοντέλο Δεδομένων, Παρουσίαση και Πρόγραμμα ελέγχου. Struts επιτυγχάνει πολύ την ανάπτυξη των δικτυακών εφαρμογών αφού επεξεργάζεται αιτήσεις HTTP σε μια τυποποιημένη διαδικασία, χρησιμοποιώντας τυποποιημένες τεχνολογίες όπως Java Servlets. Αυτό σημαίνει ότι πολλές λειτουργίες οι οποίες είναι σχετικές με τις εφαρμογές περιέχονται ήδη και είναι έτοιμοι προς εφαρμογή.

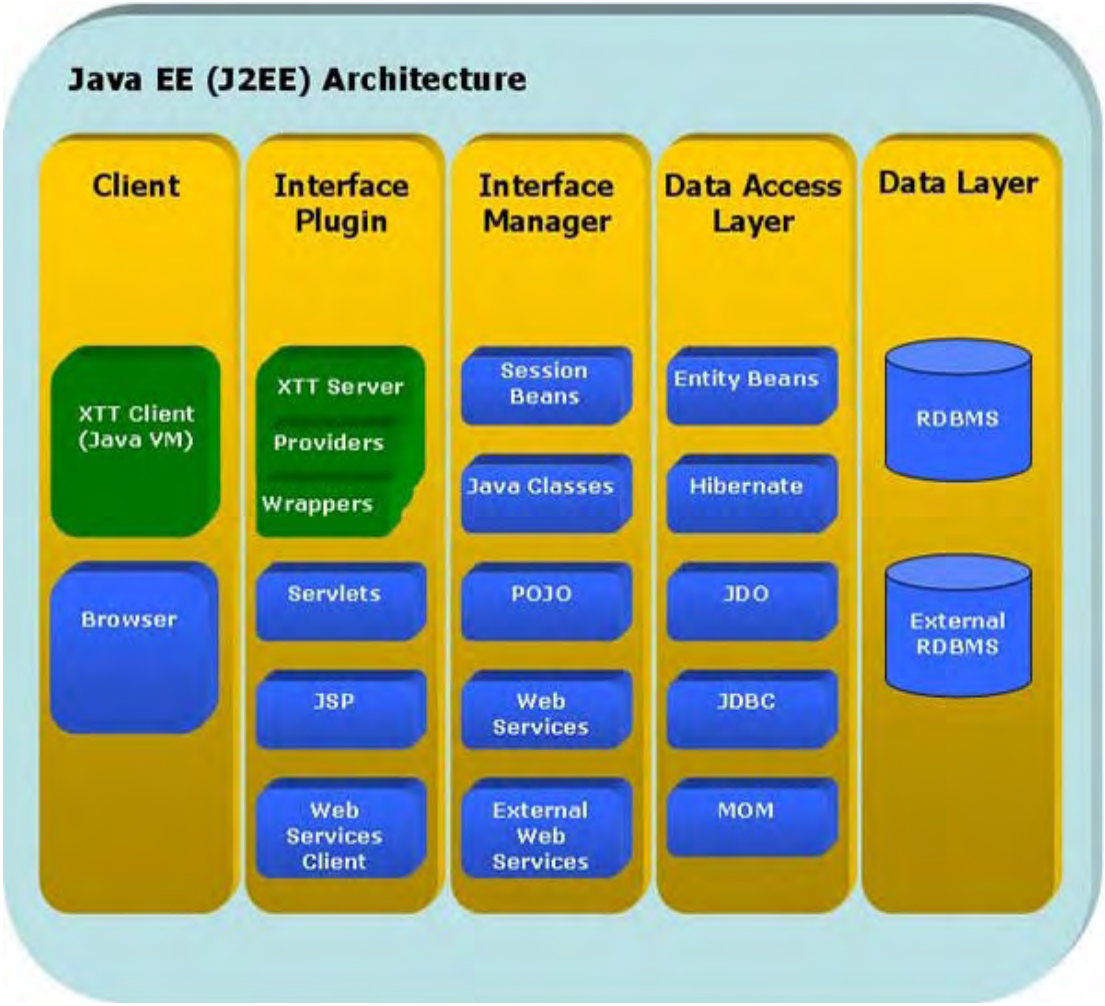


Εικόνα: MVC-Pattern

Apache Struts είναι ένα open-source web application framework για την ανάπτυξη δικτυακών εφαρμογών σε Java EE. Σε standard Java EE, ο πελάτης θα υποβάλλει πληροφορίες στο διακομιστή μέσω μιας φόρμας ιστού. Οι πληροφορίες αυτές στη συνέχεια είτε παραδίδεται σε ένα Java Servlet που επεξεργάζεται, αλληλεπιδρά με μια βάση δεδομένων και παράγει μία μορφή HTML formatted response, ή να δοθεί σε JavaServer Pages (JSP) έγγραφο που σχετίζει HTML και Java κώδικα για να επιτευχθεί το ίδιο αποτέλεσμα . Και οι δύο προσεγγίσεις είναι ανεπαρκείς για μεγάλα έργα, διότι η λογική εφαρμογή αναμειγνύεται με την παρουσίαση και κάνουν δύσκολη τη συντήρηση. Ο στόχος της Struts είναι να διαχωριστούν καθαρά το μοντέλο δεδομένων(λογική της εφαρμογής που αλληλεπιδρά με μια βάση δεδομένων), από την παρουσίαση (σελίδες HTML που βλέπει ο χρήστης) και το πρόγραμμα ελέγχου (π.χ. που περνάει πληροφορίες μεταξύ παρουσίαση και το μοντέλο δεδομένων).

Οι αιτήσεις από τον πελάτη αποστέλλεται στον ελεγκτή με τη μορφή "Ενέργειες" που ορίζονται στο αρχείο ρυθμίσεων, εάν βέβαια ο υπεύθυνος παραλαβής της σχετικής αίτησης

ζητά την αντίστοιχη κατηγορία δράσης που αλληλεπιδρά με την ειδική εφαρμογή model code. Το model code επιστρέφει ένα "ActionForward", ένα αλφαριθμητικό που λέει πως το πρόγραμμα ελέγχου να στείλει την σελίδα προς τον πελάτη. Οι πληροφορίες διαβιβάζονται μεταξύ του μοντέλου δεδομένων και την παρουσίαση με τη μορφή ειδικών JavaBeans. Μία ισχυρή βιβλιοθήκη του επιτρέπει να διαβάζει και να γράφει το περιεχόμενο των Beans από το στρώμα παρουσίασης (presentation layer)χωρίς την ανάγκη για ενσωματωμένο κώδικα Java.



7. Σύστημα ασφαλών τραπεζικών συναλλαγών

Το Proxypay3 SSL Gateway επεξεργάζεται την πληρωμή χρησιμοποιώντας το ειδικό πρωτόκολλο που χρειάζονται για μια συγκεκριμένη μέθοδο πληρωμής. Για τη διεκπεραίωση της συναλλαγής, η διασύνδεση με το host έχει εφαρμοστεί στο κομμάτι του proxypay3.

Η Element έχει χτίσει το proxypay3 για να χειριστεί online συναλλαγές, είναι ένα πλήρες σύστημα, που επιτρέπει την επαλήθευση των στοιχείων των συναλλαγών, το χειρισμό των συναλλαγών και να περάσει όλες τις λεπτομέρειες μετά την επεξεργασία. Υλοποιεί το POS και Gateway λειτουργίες για την παγκόσμια ασφάλεια πληρωμής στο Διαδίκτυο. Proxypay3 έχει αναπτυχθεί με τους εξής στόχους:

Εύκολο να εφαρμοστεί, χρησιμοποιώντας το δικό του σύστημα αγορών του (πρότυπο πρωτόκολλο http(s) – και οχι API). Αρθρωτό σύστημα από την πλευρά του Gateway που επιτρέπει την ταχεία ολοκλήρωση των πρόσθετων στοιχείων / μεθόδων πληρωμής.

Σύνδεση με του Merchant το ERP σύστημα χρησιμοποιώντας μια επιβεβαίωση POST σχήμα. Αποδεδειγμένη ασφάλεια κατά τη χρήση του σχήματος επικύρωσης. Προσαρμοζόμενες δυνατότητες για το Gateway (Back Office / Payment modules) και Έμπορο (Payment templates).

Η Αρχιτεκτονική, ένα έμπορο / κατάστημα Builder ενσωματώνει την online πληρωμή μέσω POST στο proxypay3 SSL Gateway μέσω ορισμένων βασικών παραμέτρων (όπως το ποσό, νόμισμα, αναφορά στους εμπόρους για τον σκοπό, ένα MerchantID, καθώς και άλλες μεταβλητές που μπορούν να χρησιμοποιηθούν, όπως απαιτείται).

Μόλις το proxypay3 έχει λάβει την αίτηση, ελέγχει τις σχετικές παραμέτρους και στέλνει ένα πρότυπο για την πληρωμή στο browser του Πελάτη.

Εάν απαιτείται, αυτό το πρότυπο πληρωμής μπορεί να τροποποιηθεί από τον Merchant έτσι ώστε να λάβει μια εμφάνιση του περιβάλλοντος του καταστήματος του.

Ανάλογα με την επιλεγμένη μέθοδο πληρωμής (αριθμός πιστωτικής κάρτας κλπ.), ο πελάτης πρέπει να εισάγει τα στοιχεία πληρωμής του.

Μόλις το proxypay3 έχει λάβει τα στοιχεία για τις πληρωμές των πελατών, ένα προαιρετικό έλεγχο επικύρωση εκτελείται στην ιστοσελίδα του Εμπόρου, πριν από την επεξεργασία των συναλλαγών.

Ο προαιρετικός έλεγχος επικύρωσης εκτελείται στέλνοντας τα λαμβανόμενα στοιχεία αποστολής (ποσό, νόμισμα και Merchantreference) πίσω σε μια URL του Merchant, η οποία επικυρώνει τότε τα δεδομένα.

Σε αυτό το script ένας Merchant / Integrator μπορεί να επαληθεύσει τις λαμβανόμενους παραμέτρους με τα στοιχεία, που βρέθηκαν στη βάση δεδομένων παραγγελιάς.

Εάν τα δεδομένα δεν ταιριάζουν με τις τιμές που περιλαμβάνονται στη βάση δεδομένων του Merchant, τότε ο πελάτης ανακατευθύνεται σε μια σελίδα του ιστοτόπου που χειρίζεται την απόρριψη πληρωμών.

Αν ταιριάζουν τα δεδομένα, τότε το script επιστρέφει, OK. στο proxypay3 και η διαδικασία της πληρωμής επεξεργάζεται.

Εάν η πληρωμή είναι επιτυχής, ανακατευθύνεται σε μια σελίδα στην πλευρά του πελάτη η οποία διαχειρίζεται την επιτυχή διεκπεραίωση των πληρωμών (δηλαδή δείχνει πληροφορίες για την παραγγελία και σε ποια φάση βρίσκεται).

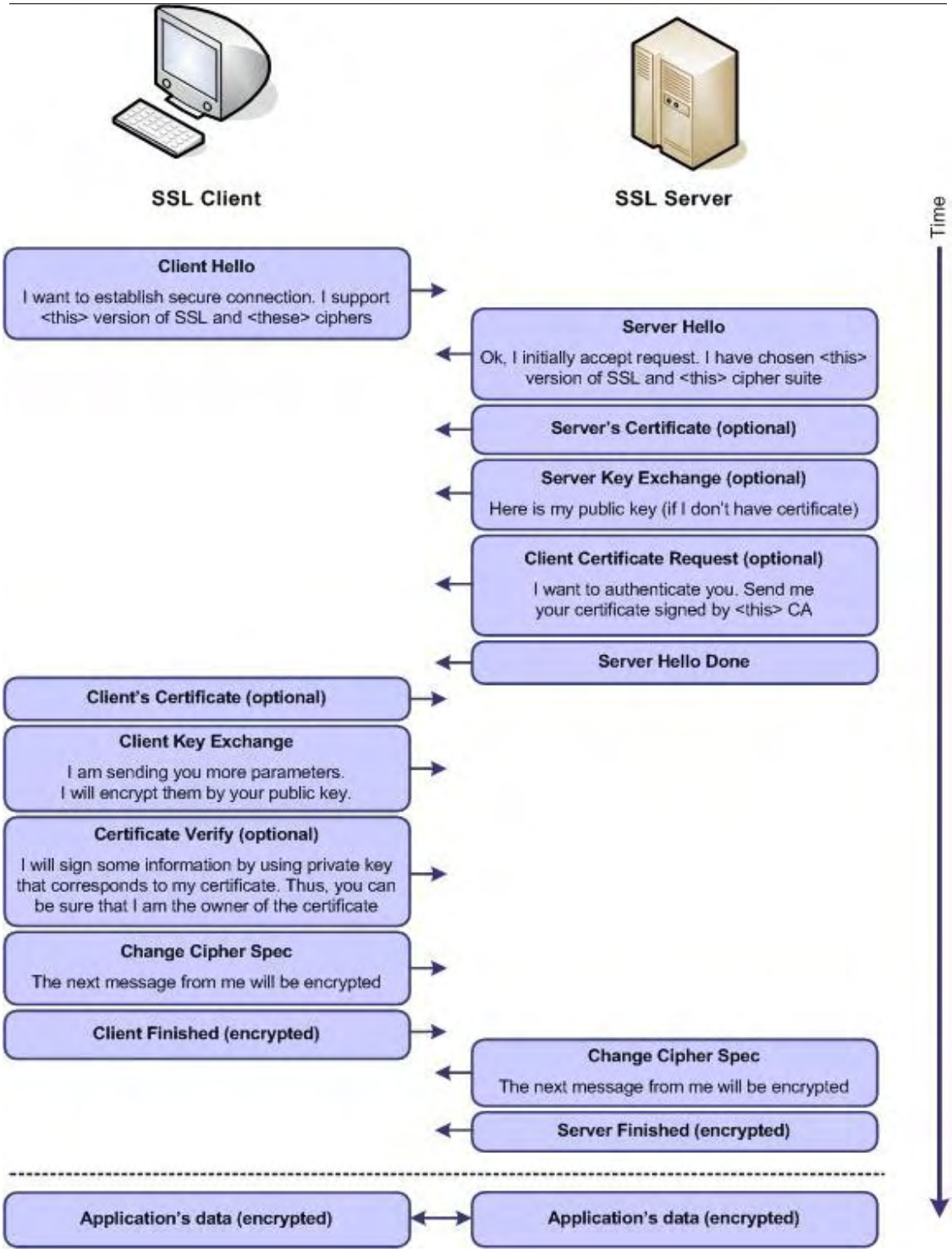
Ο Merchant ειδοποιείται με διάφορος τρόπος από την υπηρεσία ειδοποίησης (deamon). Αυτό το module στέλνει τα στοιχεία για επιτυχημένη πληρωμή σε μια διεύθυνση URL του Merchant, η οποία χειρίζεται τις επιτυχημένες πληρωμές, και ανάλογα της περιπτώσεις εκτελεί ορισμένες εξατομικευμένες δράσεις, όπως πχ. σύνδεση με το σύστημα ERP, επικαιροποίηση της βάσης δεδομένων, στέλνει ένα email στον πελάτη. Επιπλέον, ο Merchant λαμβάνει ένα μήνυμα ηλεκτρονικού ταχυδρομείου ως επιβεβαίωση της παραγγελίας. Ο Έμπορος είναι σε θέση να επεξεργαστεί / δει τις πληρωμές χρησιμοποιώντας το proxypay3 Back Office.

Αυτό το Back Office υλοποιείται χρησιμοποιώντας μια γλώσσα δέσμης ενεργειών που επιτρέπει την εκτέλεση των δηλώσεων βάσης δεδομένων σε συνδυασμό με το πρότυπο εντολές HTML, αλλά με την εφαρμογή ασφάλειας.

Με αυτόν τον τρόπο, τα δεδομένα δεν μπορούν να δειχθούν εάν το απαιτούμενο επίπεδο δεν έχει οριστεί (το επίπεδο πρόσβασης έχει κατά τη σύνδεση με το σύστημα Back Office)

Το σύστημα ελέγχου ταυτότητας Back Office μπορει να εφαρμοστεί σαν απαίτηση στο proxypay3, αλλά το πρότυπο για πρόσβαση είναι το σύστημα ελέγχου ταυτότητας χρήστη / κωδικό.

Στην εικόνα βλέπουμε μια εισαγωγή στην διαδικασία Handshake του SSL (Secure Sockets Layer) .


SSL Client
SSL Server
Time
Client Hello
I want to establish secure connection. I support <this> version of SSL and <these> ciphers
Server Hello
Ok, I initially accept request. I have chosen <this> version of SSL and <this> cipher suite
Server's Certificate (optional)
Server Key Exchange (optional)
Here is my public key (if I don't have certificate)
Client Certificate Request (optional)
I want to authenticate you. Send me your certificate signed by <this> CA
Server Hello Done
Client's Certificate (optional)
Client Key Exchange
I am sending you more parameters. I will encrypt them by your public key.
Certificate Verify (optional)
I will sign some information by using private key that corresponds to my certificate. Thus, you can be sure that I am the owner of the certificate
Change Cipher Spec
The next message from me will be encrypted
Client Finished (encrypted)
Change Cipher Spec
The next message from me will be encrypted
Server Finished (encrypted)
Application's data (encrypted)
Application's data (encrypted)

8. Βιβλιογραφία

- Chuck Cavaness, Jakarta Struts 2002
- Bodoff, Stephanie (2004). The J2EE Tutorial
- Jonas Jacobi, John R. Fallows: Pro JSF and Ajax
- http://www.softscout.com
- Deepak Alur, John Crupi, Dan Malks, Core J2EE Patterns Best Practices And Design Strategies
- http://www.jakarta.apache.org/taglibs/index.html
- http://java.sun.com
- John Ferguson Smart, JSF Jumpstart - A tutorial introduction to building web sites using JSF
- David Geary, Cay Horstmann, Core JavaServer Faces, Second Edition
- Hans Bergsten, JavaServer Faces, O'Reilly & Associates
- James Turner, Craig McClanahan, Kunal Mittal, JavaServer Faces Kick Start
- http://www.developintelligence.com/learn/tutorials.php#JSF
- Perrone, Paul J Chaganti, Krishna (2003). J2EE Developer's Handbook
- Crane, D., et al. Ajax in Action. Greenwich, Manning Publications Co., 2006.
- Thau, D., The Book of Javascript, 2nd
- Edition, San Francisco, No Starch Press, 2007.
- Sklar, D., Trachtenberg, A., PHP Cookbook, 2nd Edition, Sebastopol, O'Reilly, 2006.
- Schmitt, C., PHP Cookbook, Sebastopol, O'Reilly, 2004.
- Pfaffenberger, B., et al. HTML, XHTML, and CSS Bible, 3rdEdition, Indianapolis, Wiley, 2004.
- http://www.php.net
- http://script.aculo.us
- http://mir.aculo.us

- http://w3schools.com
- http://www.w3.org