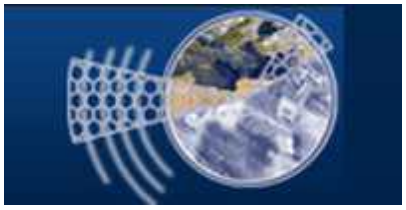
Τεχνολογικό Εκπαιδευτικό Ίδρυμα Κρήτης

Σχολή Τεχνολογικών εφαρμογών

Τμήμα Μηχανικών Πληροφορικής



Πτυχιακή Εργασία

Τίτλος: Σύστημα για παροχή ασφαλιστικών υπηρεσιών

Λιοφάγου Δέσποινα(ΑΜ:2763)

Επιβλέπων Καθηγητής: Παπαδάκης Νικόλαος

Ηράκλειο 2015

Ευχαριστίες

Πρώτα από όλα θα ήθελα να ευχαριστήσω την οικογένεια μου για την πολύπλευρη στήριξη τους όλα αυτά τα χρόνια των σπουδών μου. Επίσης θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή μου κ. Νικόλαο Παπαδάκη για την ανάθεση της συγκεκριμένης πτυχιακής.

Abstract

In this thesis, a system was created for the provision of insurance services for a car insurance company. Initially, the installation of the XAMPP took place and then followed the system development process , starting from its designing process and reaching its implementation .Some summative reports were also made. The main technologies used for the implementation of this thesis are the following: PHP, JavaScript and Ajax. The SQL language was used for the creation and the management of the database.

Περίληψη

Σε αυτή την πτυχιακή εργασία δημιουργήθηκε ένα σύστημα για την παροχή ασφαλιστικών υπηρεσιών, για μια ασφαλιστική εταιρία αυτοκινήτων. Αρχικά, πραγματοποιήθηκε η εγκατάσταση του XAMPP και μετά ακλούθησε η διαδικασία ανάπτυξης του συστήματος, ξεκινώντας από τη διαδικασία σχεδίασης του φτάνοντας μέχρι και την υλοποίηση του. Επίσης έγινε η υλοποίηση κάποιων συγκεντρωτικών αναφορών. Οι κύριες τεχνολογίες που χρησιμοποιήθηκαν για την υλοποίηση της εργασία είναι οι εξής : PHP, JavaScript και Ajax. Για την δημιουργία και την διαχείριση της βάσης δεδομένων χρησιμοποιήθηκε η γλώσσα SQL.

Περιεχόμενα

Ευχαριστίες 2
Abstract 3
Περίληψη 4
Πίνακας εικόνων 7
1 Εισαγωγή 9
1.1 Δομή εργασίας 9
2 Θεωρίες 10
2.1 Τι είναι το διαδίκτυο 10
2.2 World Wide Web 10
2.3 Τι είναι μια ιστοσελίδα 10
2.4 Στατικές και δυναμικές ιστοσελίδες 10
3 Οι βασικές τεχνολογίες, γλώσσες και προγράμματα που χρησιμοποιήθηκαν. 11
3.1 PHP 11
3.2 JavaScript 11
3.3 Ajax 11
3.4 Css 11
3.5 ExtJS 12
3.6 Τι είναι η MySQL 12
3.7 Τι είναι ο Apache server 12
3.8 Τι είναι PhpMyAdmin 12
3.9 XAMPP 12
3.9.1 Εγκατάσταση XAMPP 12
4 Μοντέλο οντοτήτων-συσχετίσεων 17
5 Σχεσιακό Μοντέλο 18
6 Βάση δεδομένων 19
6.1 Δημιουργία των πινάκων τις βάσης 19
6.2 Περιγραφή των πινάκων τις βάσης 21
7 Λειτουργικότητα του συστήματος 23
7.1 Σύντομη περιγραφή του συστήματος 23
7.2 Λειτουργίες που υποστηρίζει το σύστημα 23
7.3 Σελίδα Intex.php 24
7.4 Σελίδα Admin.php 24
7.4.1 Δημιουργία -Τροποποίηση- Διαγραφή τμήματος 24

7.4.2 Δημιουργία-Τροποποίηση-Διαγραφή Υπαλλήλου 27
7.4.3 Δημιουργία-Τροποποίηση-Διαγραφή Κατηγορίας οχήματος 31
7.5 Σελίδα Home.php 35
7.5.1 Δημιουργία-Τροποποίηση-Διαγραφή Πελατών(εταιρικών και μη) 35
7.5.2 Δημιουργία-Τροποποίηση-Διαγραφή Οχημάτων(εταιρικών και μη) 38
7.5.3 Δημιουργία-Τροποποίηση-Διαγραφή Συμβολαίων 41
7.5.4 Δημιουργία-Τροποποίηση-Διαγραφή Ατυχημάτων 44
7.5.5 Συγκεντρωτική αναφορά για τα έσοδα και έξοδα κάθε μήνα ανά πελάτη(Analytic Balance) 48
7.5.6 Συγκεντρωτική αναφορά για τα έσοδα και έξοδα κάθε μήνα ανά κατηγορία οχήματος, ανά φύλο και ανά ηλικία(Groupped Balance). 49
7.5.7 Συγκεντρωτική αναφορά τα ατυχήματα αν φύλλο πελατών και αν ηλικία(Groupped Accidents) 51
7.5.8 Συγκεντρωτική αναφορά για τα έσοδα, έξοδα και συμβόλαια που κάνει κάθε υπάλληλος/τμήμα σε ένα συγκεκριμένο χρονικό διάστημα(Groupped Balance by Employee).. 52
Βιβλιογραφία 54
ΠΑΡΑΡΤΗΜΑ 55

Πίνακας εικόνων

Εικόνα 1: Πατάω Next 13
Εικόνα 2: Επιλογή Components 13
Εικόνα 3: Επιλογή προορισμού εγκατάστασης 14
Εικόνα 4: Έτοιμο για εγκατάσταση 14
Εικόνα 5:Γίνεται η εγκατάσταση 15
Εικόνα 6:Τέλος εγκατάστασης 15
Εικόνα 7:Ενεργοποίηση Apache και MySQL 16
Εικόνα 8:Επιβεβαίωση επιτυχούς εγκατάστασης XAMPP 16
Εικόνα 9: ER-model 17
Εικόνα 10: Σχεσιακό Μοντέλο 18
Εικόνα 11:Σελίδα Admin.php 24
Εικόνα 12:Logout από Admin.php 24
Εικόνα 13:Αρχική οθόνη τμημάτων 25
Εικόνα 14:Φόρμα εισαγωγής τμήματος 25
Εικόνα 15:Εισαγωγή στοιχείων για την δημιουργία τμήματος 25
Εικόνα 16:Εμφάνιση νέου τμήματος 26
Εικόνα 17:Τροποποίηση τμήματος 26
Εικόνα 18: Διαγραφή τμήματος 27
Εικόνα 19:Αρχική οθόνη υπαλλήλων 27
Εικόνα 20:Φόρμα δημιουργίας υπαλλήλου 28
Εικόνα 21:Εισαγωγή στοιχείων για την δημιουργία υπαλλήλου 28
Εικόνα 22:Ανεπιτυχής τροποποίηση 29
Εικόνα 23:Τροποποίηση Υπαλλήλου 29
Εικόνα 24:Ανεπιτυχής διαγραφή υπαλλήλου 30
Εικόνα 25:Διαγραφή Υπ6 30
Εικόνα 26:Αρχική οθόνη κατηγοριών οχημάτων 31
Εικόνα 27:Φόρμα δημιουργίας κατηγορίας οχημάτων 32
Εικόνα 28:Εισαγωγή στοιχείων στην φόρμα κατηγορίας οχημάτων 32
Εικόνα 29:Τροποποίηση κατηγορίας οχήματος 33
Εικόνα 30:Διαγραφή επιλεγμένης κατηγορίας οχήματος 34
Εικόνα 31:Κατηγορίες που διαχειρίζεται η σελίδα 35
Εικόνα 32:Συγκεντρωτικές αναφορές 35
Εικόνα 33:Έξοδος απο την Home.php 35
Εικόνα 34:Αρχική οθόνη πελατών 36
Εικόνα 35:Φόρμα δημιουργίας πελάτη 36
Εικόνα 36:Εισαγωγή στοιχείων στην φόρμα δημιουργίας πελάτη 37
Εικόνα 37:Τροποποίηση πελάτη 37
Εικόνα 38:Ανεπιτυχής διαγραφή πελάτη 38
Εικόνα 39:Αρχική οθόνη οχημάτων 38
Εικόνα 40:Φόρμα δημιουργίας οχήματος 39
Εικόνα 41:Εισαγωγή στοιχείων στην φόρμα δημιουργίας οχήματος 39
Εικόνα 42:Τροποποίηση οχήματος 40
Εικόνα 43:Ανεπιτυχής διαγραφή οχήματος 41
Εικόνα 44:Αρχική οθόνη Συμβολαίων 41

Εικόνα 45:Φόρμα δημιουργίας συμβολαίου 42
Εικόνα 46:Εισαγωγή στοιχείων στην φόρμα ασφαλιστηρίου 42
Εικόνα 47:Τροποποίηση συμβολαίου 43
Εικόνα 48:Διαγραφή συμβολαίου 44
Εικόνα 49:Αρχική οθόνη ατυχημάτων 45
Εικόνα 50:Φόρμα δημιουργίας ατυχήματος 45
Εικόνα 51:Εισαγωγή στοιχείων στην φόρμα δημιουργίας ατυχήματος 46
Εικόνα 52:Τροποποίηση ατυχήματος 47
Εικόνα 53:Διαγραφή ατυχήματος 47
Εικόνα 54:Analytic Balance 48
Εικόνα 55:Έσοδα - έξοδα ανά πελάτη 11/2014 49
Εικόνα 56:Groupped Balance 49
Εικόνα 57:Έσοδα - έξοδα ανά κατηγορία οχήματος ανα φύλο και ανα ηλικία 1/2015 50
Εικόνα 58:Groupped Accidents 51
Εικόνα 59:Αριθμός ατυχημάτων ανά φύλο και ανά ηλικία πελατών 51
Εικόνα 60:Groupped Balance by Employee 52
Εικόνα 61:Έσοδα- έξοδα- συμβόλαια υπαλλήλου ανά τμήμα 52

1 Εισαγωγή

1.1 Δομή εργασίας

Η δομή της εργασίας περιλαμβάνει τα εξής:

- Κεφάλαιο 2:Ανάπτυξη θεωρητικών κομματιών.
- Κεφάλαιο 3:Περιγραφή των τεχνολογιών που χρησιμοποιήθηκαν στην πτυχιακή και η εγκατάσταση του XAMPP .
- Κεφάλαιο 4:Δημιουργήθηκε το μοντέλο οντοτήτων-συσχετίσεων.
- Κεφάλαιο 5: Δημιουργία σχεσιακού μοντέλου.
- Κεφάλαιο 6:Αυτό το κεφάλαιο περιλαμβάνει τον κώδικα για την δημιουργία των πινάκων της βάσης, καθώς και την περιγραφή αυτών.
- Κεφάλαιο 7:Στο κεφάλαιο αυτό υπάρχει μια σύντομή περιγραφή του συστήματος, ακολουθεί η περιγραφή των λειτουργιών που υποστηρίζει το σύστημα, γενικά και μετά ανά σελίδα, και στην συνέχεια ένα εγχειρίδιο χρήσης για κάθε σελίδα του συστήματος.
- Βιβλιογραφία.
- Παράρτημα: Περιέχει τα στοιχεία σύνδεσης με την MySQL, τις Stored procedures που υλοποιούν τις αναφορές και τα sql queries για όλα τα ερωτήματα που έχουν υλοποιηθεί.

2 Θεωρίες

2.1 Τι είναι το διαδίκτυο

Το διαδίκτυο είναι ένα παγκόσμιο δίκτυο υπολογιστών που συνδέονται και επικοινωνούν μεταξύ τους με τη χρήση κυρίως των πρωτοκόλλων TCP και IP, βέβαια αυτά δεν είναι τα μοναδικά. Οι διασυνδεδεμένοι υπολογιστές οι οποίοι είναι σε ένα κοινό δίκτυο επικοινωνίας κάνουν ανταλλαγή μηνυμάτων, χρησιμοποιώντας διάφορα πρωτόκολλα τα οποία υλοποιούνται σε επίπεδο υλικού και λογισμικού. Το κοινό αυτό δίκτυο λέγεται διαδίκτυο. Στην πιο εξειδικευμένη μορφή με τον όρο διαδίκτυο περιγράφεται το παγκόσμιο πλέγμα διασυνδεδεμένων υπολογιστών περιλαμβανομένων και των υπηρεσιών και πληροφοριών που παρέχει στους χρήστες.

2.2 World Wide Web

Το World Wide Web (Παγκόσμιος Ιστός) είναι ένα κατανεμημένο πληροφοριακό σύστημα οργάνωσης και πρόσβασης πληροφοριών που υλοποιεί τις βασικές αρχές οργάνωσης του υπερκειμένου. Ως πληροφοριακό σύστημα παρέχει συγκεκριμένο μοντέλο δεδομένων το οποίο βασίζεται σε κόμβους και υπερσυνδέσμους. Δημιουργήθηκε στα τέλει της δεκαετίας του 1980. Όταν γράφτηκε ήταν το μοναδικό πρόγραμμα για την περιήγηση στον παγκόσμιο ιστό. Ο πηγαίος κώδικας έγινε κοινό κτήμα το 1993. Παραμένει σαν ιστορικό τεχνούργημα σε έναν υπολογιστή NextCube στο μουσείο του CERN.

2.3 Τι είναι μια ιστοσελίδα

Ιστοσελίδα είναι ένα είδος εγγράφου του παγκόσμιου ιστού (WWW) που περιέχει πληροφορίες με την μορφή κειμένου, υπερκειμένου, εικόνας, βίντεο και ήχου. Πολλές ιστοσελίδες μαζί αποτελούν έναν ιστότοπο. Οι σελίδες ενός ιστοτόπου εμφανίζονται κάτω από το ίδιο όνομα χώρου. Οι ιστοσελίδες συνδέονται η μια με την άλλη και μπορεί ο χρήστης να πάει από τη μία στην άλλη κάνοντας «κλικ», επιλέγοντας δηλαδή συνδέσμους που υπάρχουν στο κείμενο ή στις φωτογραφίες της ιστοσελίδας. Οι σύνδεσμοι που μας οδηγούν σε άλλες σελίδες εμφανίζονται συνήθως υπογραμμισμένοι και με μπλε χρώμα για να είναι γρήγορα ξεκάθαρο στον επισκέπτη ότι πρόκειται για σύνδεσμο προς άλλη ιστοσελίδα, αλλά αυτό δεν είναι πάντα απαραίτητο.

2.4 Στατικές και δυναμικές ιστοσελίδες

Στατική ιστοσελίδα λέμε μια ιστοσελίδα της οποίας το περιεχόμενο μεταφέρεται στον χρήστη ακριβώς στην μορφή που είναι αποθηκευμένο στον εξυπηρετητή ιστοσελίδων. Τα περιεχόμενα μιας στατικής ιστοσελίδας εμφανίζονται με την ίδια μορφή σε όλους τους χρήστες ,με την μορφή που είναι αποθηκευμένα στο σύστημα αρχείων του εξυπηρετητή ιστοσελίδων. Οι στατικές ιστοσελίδες είναι αποθηκευμένες συνήθως σε μορφή HTML και μεταφέρονται χρησιμοποιώντας το πρωτόκολλο HTTP.

Δυναμική ιστοσελίδα λέμε μια ιστοσελίδα η οποία δημιουργείται δυναμικά την στιγμή της πρόσβασης σε αυτή ή την στιγμή που ο χρήστης αλληλεπιδρά με τον εξυπηρετητή ιστοσελίδων. Οι δυναμικές ιστοσελίδες θεωρούνται δομικό στοιχείο της νέας γενιάς του παγκόσμιου ιστού όπου η πληροφορία διαμοιράζεται σε πολλαπλές ιστοσελίδες .Η δυναμική ιστοσελίδα μπορεί να δημιουργείται δυναμικά από ένα σενάριο εντολών, το οποίο εκτελείται τοπικά στο πελάτη ή στον εξυπηρετητή ή και στον πελάτη και στον εξυπηρετητή.

3 Οι βασικές τεχνολογίες, γλώσσες και προγράμματα που χρησιμοποιήθηκαν.

3.1 PHP

Η php είναι μια γλώσσα προγραμματισμού για τη δημιουργία web σελίδων με δυναμικό περιεχόμενο. Μια σελίδα PHP περνά από επεξεργασία από ένα συμβατό διακομιστή του Παγκόσμιου Ιστού (π.χ. Apache), ώστε να παραχθεί σε πραγματικό χρόνο το τελικό περιεχόμενο, που θα σταλεί στο πρόγραμμα περιήγησης των επισκεπτών σε μορφή κώδικα HTML. Αυτή η γλώσσα εκτελείται στην πλευρά του server (server-side scripting). Αυτό έχει ως πλεονέκτημα τη δημιουργία δυναμικών ιστοσελίδων οι οποίες αντλούν τα δεδομένα τους από κάποια βάση δεδομένων αλλά παρέχει και τη δυνατότητα διαχείρισης αυτής της βάσης με σκοπό τη διαχείριση περιεχομένου της δυναμικής ιστοσελίδας.

3.2 JavaScript

Η JavaScript (JS) είναι μια διερμηνευμένη γλώσσα προγραμματισμού για ηλεκτρονικούς υπολογιστές. Αρχικά ήταν μέρος της υλοποίησης των φυλλομετρητών ιστού ώστε τα σενάρια από την πλευρά του πελάτη (client-side scripts) να μπορούν να επικοινωνούν με τον χρήστη, να ανταλλάσσουν δεδομένα ασύγχρονα και να μεταβάλουν δυναμικά το περιεχόμενο του εγγράφου που εμφανίζεται.

Η JavaScript είναι μια γλώσσα σεναρίων που βασίζεται στα πρωτότυπα (prototype-based), είναι δυναμική, με ασθενείς τύπους και έχει συναρτήσεις ως αντικείμενα πρώτης τάξης. Η σύνταξή της είναι επηρεασμένη από τη C. Η JavaScript αντιγράφει πολλά ονόματα και συμβάσεις ονοματοδοσίας από τη java, αλλά αυτές οι δύο γλώσσες δε σχετίζονται και έχουν πολύ διαφορετική σημασιολογία. Οι βασικές αρχές σχεδιασμού της JavaScript προέρχονται από τις γλώσσες προγραμματισμού Self και Scheme.

3.3 Ajax

Ajax (programming) είναι μια ομάδα αλληλένδετων τεχνικών ανάπτυξης Web που χρησιμοποιούνται στην πλευρά του πελάτη για την δημιουργία ασύγχρονων εφαρμογών Web. Με Ajax, διαδικτυακές εφαρμογές μπορούν να σταλούν δεδομένα και να ανακτηθούν από έναν διακομιστή ασύγχρονα (στο παρασκήνιο). Η Ajax δεν είναι μια ενιαία τεχνολογία, αλλά μια ομάδα τεχνολογιών HTML και CSS μπορεί να χρησιμοποιηθεί σε συνδυασμό με τη σήμανση και τις πληροφορίες στυλ. Το DOM είναι προσβάσιμο με τη JavaScript για να εμφανιστεί δυναμικά - και επιτρέπουν στο χρήστη να αλληλεπιδρά με - τις πληροφορίες που παρουσιάζονται .JavaScript και το αντικείμενο XMLHttpRequest παρέχει μια μέθοδο για την ανταλλαγή δεδομένων ασύγχρονα μεταξύ browser και server για να αποφύγει την πλήρη επαναφόρτωση της σελίδας.

3.4 Css

Η CSS είναι μια γλώσσα υπολογιστή. Ανήκει στην κατηγορία των γλωσσών φύλλων στυλ που χρησιμοποιείται για τον έλεγχο της εμφάνισης ενός εγγράφου που έχει γραφτεί με μια γλώσσα σήμανσης. Χρησιμοποιείται δηλαδή για τον έλεγχο της εμφάνισης μιας ιστοσελίδας και γενικότερα ενός ιστότοπου. Η CSS είναι μια γλώσσα υπολογιστή προορισμένη να αναπτύσσει στυλιστικά μια ιστοσελίδα δηλαδή να διαμορφώνει περισσότερα χαρακτηριστικά, χρώματα, στοίχιση και δίνει περισσότερες δυνατότητες σε σχέση με την html. Για μια όμορφη και καλοσχεδιασμένη ιστοσελίδα η χρήση της CSS κρίνεται ως απαραίτητη.

3.5 ExtJS

Το ExtJS είναι ένα JavaScript Framework που εκτελείτε στην πλευρά του client και υποστηρίζει την τεχνολογία Ajax. Δημιουργήθηκε για την ανάπτυξη web εφαρμογών. Για να μπορέσει ένας browser να εκτελέσει ένα σενάριο γραμμένο σε ExtJS ,αρκεί να διαθέτει έναν ερμηνευτή JavaScript.

3.6 Τι είναι η MySQL

Η MySQL είναι ένα πολύ δυνατό και γρήγορο, σύστημα διαχείρισης βάσεων δεδομένων που μετρά περισσότερες από 11 εκατομμύρια εγκαταστάσεις. Το πρόγραμμα τρέχει έναν εξυπηρετητή παρέχοντας πρόσβαση σε πολλούς χρήστες σε ένα σύνολο βάσεων δεδομένων. Μια βάση δεδομένων επιτρέπει να αποθηκεύει ,να αναζητάει, να ταξινομεί και να ανακαλεί τα δεδομένα που επιθυμεί ένας χρήστης πολύ αποτελεσματικά. Χρησιμοποιεί την SQL (Structured Query Language) την τυπική γλώσσα ερωτημάτων για βάσεις δεδομένων.

3.7 Τι είναι ο Apache server

Ο Apache HTTP γνωστός απλά και ως Apache είναι ένας εξυπηρετητής του παγκόσμιου ιστού (web). Όταν κάποιος χρήστης επισκέπτεται έναν ιστότοπο το πρόγραμμα πλοήγησης (browser) επικοινωνεί μέσω του πρωτοκόλλου HTTP με έναν διακομιστή (server), ο οποίος παράγει τις ιστοσελίδες και τις αποστέλλει στο πρόγραμμα πλοήγησης. Ο Apache είναι ένας από τους πιο γνωστούς εξυπηρετητές ιστού, γιατί είναι πρόγραμμα ελεύθερου λογισμικού και λειτουργεί σε διάφορες πλατφόρμες όπως τα Windows, το Linux το Unix και το Mac OS X. Κυκλοφόρησε υπό την άδεια λογισμικού Apache και είναι λογισμικό ανοιχτού κώδικα. Συντηρείται από μια κοινότητα ανοικτού κώδικα με επιτήρηση από το Ίδρυμα Λογισμικού Apache.

Ο Apache χρησιμοποιείται μόνο σε παγκόσμια αλλά και σε τοπικά δίκτυα ως διακομιστής συνεργαζόμενος με συστήματα διαχείρισης Βάσης Δεδομένων π.χ. Oracle, MySQL. Η πρώτη έκδοση του προγράμματος δημιουργήθηκε από τον Robert McCool, το 1993, ως ένα project του National Center for Supercomputing Applications (NCSA) με το όνομα HTTPd. Όπου θεωρείται ότι έπαιξε σημαντικό ρόλο στην αρχική επέκταση του παγκόσμιου ιστού.

3.8 Τι είναι PhpMyAdmin

PhpMyAdmin είναι ένα ελεύθερο ανοικτού κώδικα εργαλείο γραμμένο σε PHP που προορίζεται να χειριστεί τη διοίκηση της MySQL με τη χρήση ενός web browser. Μπορεί να εκτελέσει διάφορες εργασίες, όπως η δημιουργία, τροποποίηση ή διαγραφή των δεδομένων, πίνακες, πεδία και γραμμές. Εκτέλεση ερωτημάτων SQL ή τη διαχείριση των χρηστών και των δικαιωμάτων.

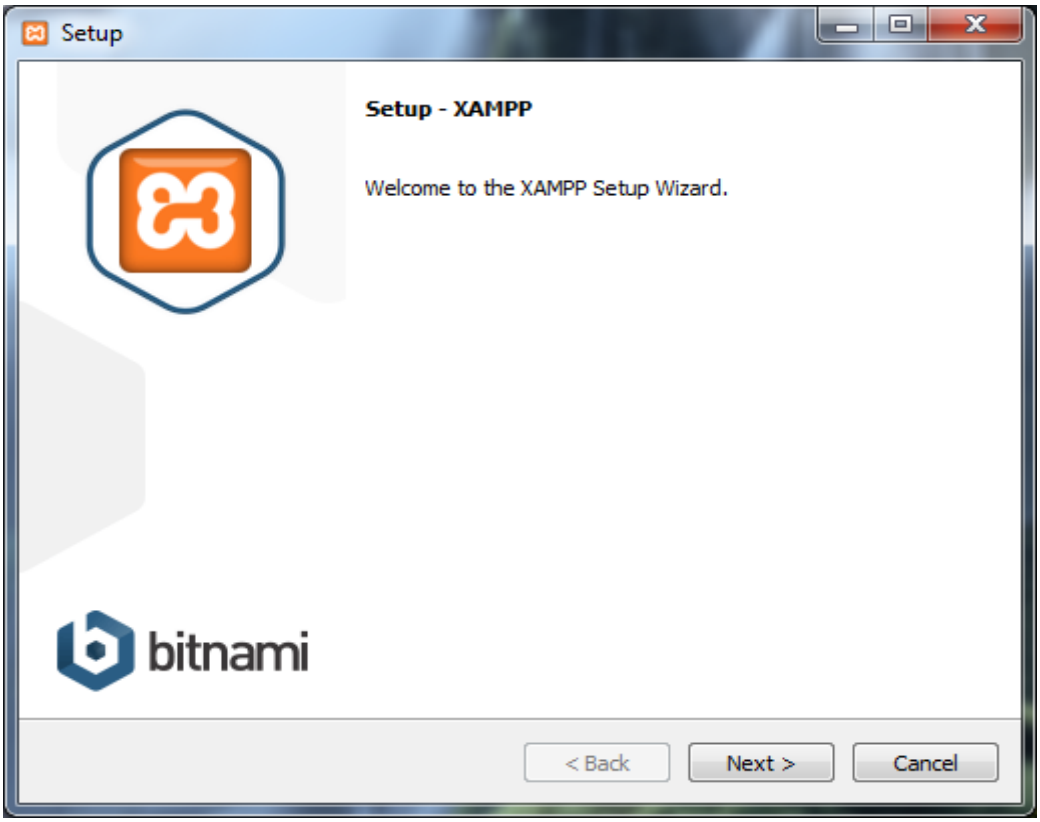
3.9 XAMPP

Το XAMPP είναι ένα πακέτο προγραμμάτων ελεύθερου λογισμικού, λογισμικού ανοικτού κώδικα και ανεξαρτήτου πλατφόρμας, το οποίο περιέχει τον εξυπηρετητή ιστοσελίδων http Apache, την βάση δεδομένων MySQL και ένα διερμηνέα για κώδικα γραμμένο σε γλώσσες προγραμματισμού PHP και Perl.

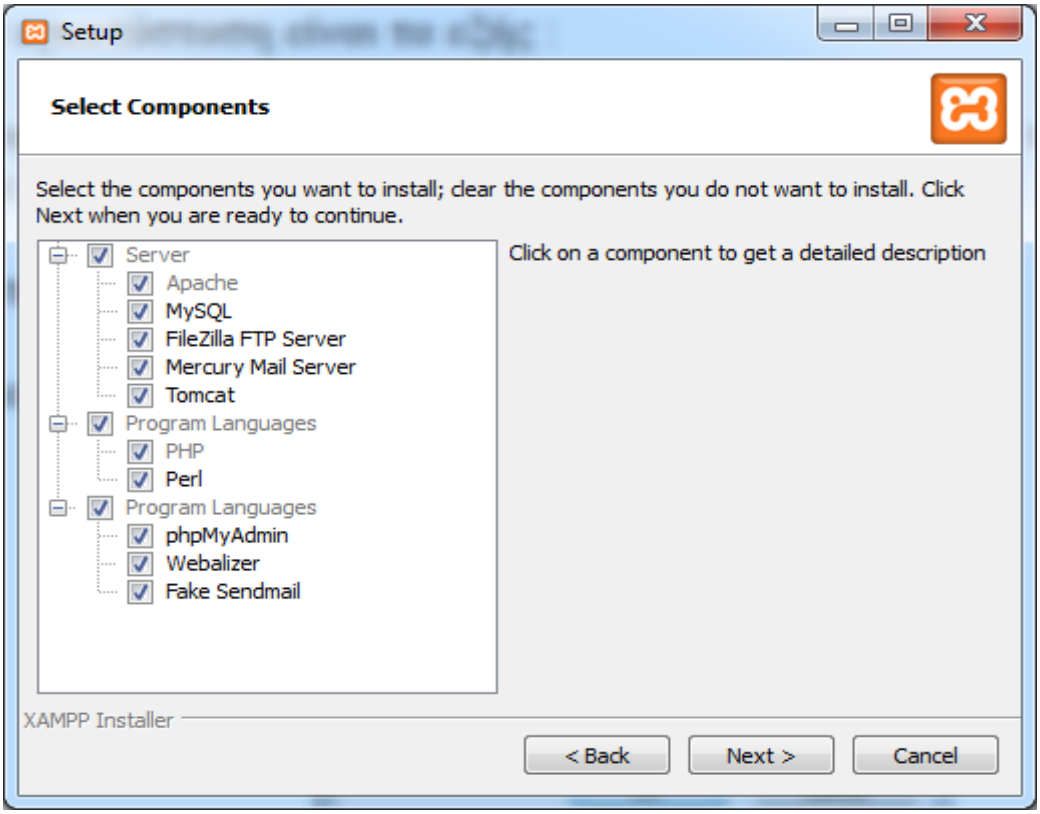
3.9.1 Εγκατάσταση XAMPP

Για την εγκατάσταση του XAMPP στον υπολογιστή μου έκανα τα εξής:
Κατέβασα το XAMPP 1.8.3 από την διεύθυνση:
http://www.apachefriends.org/en/xampp.html.

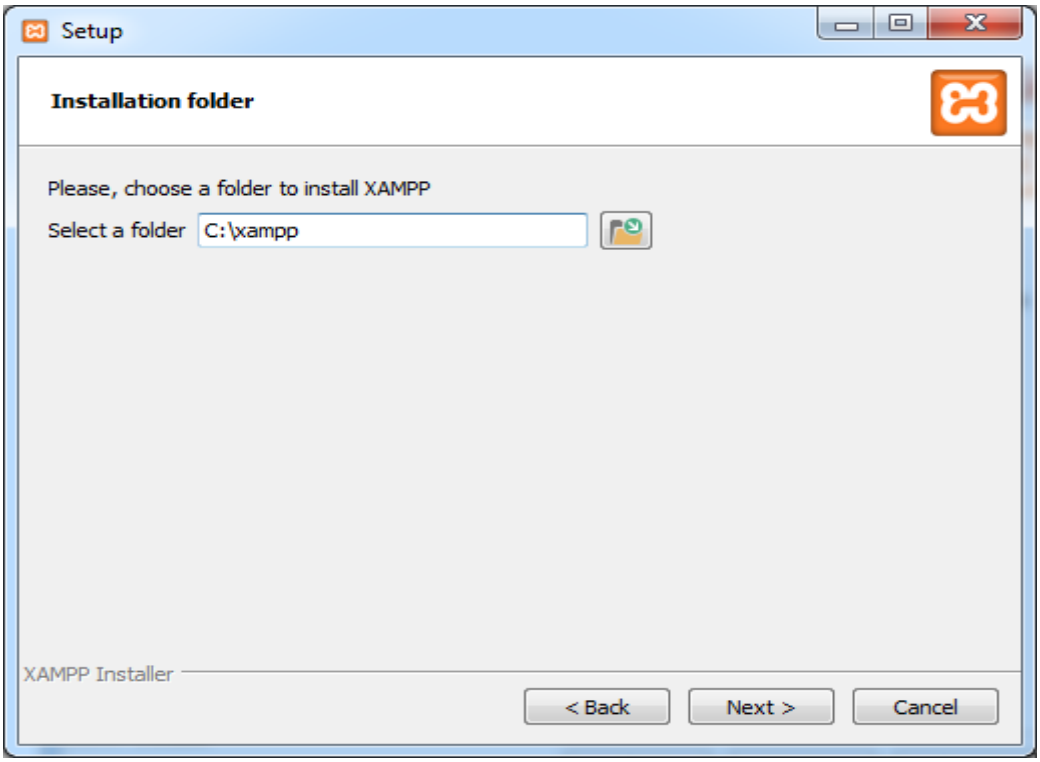
Μετά ξεκίνησα την εγκατάσταση.



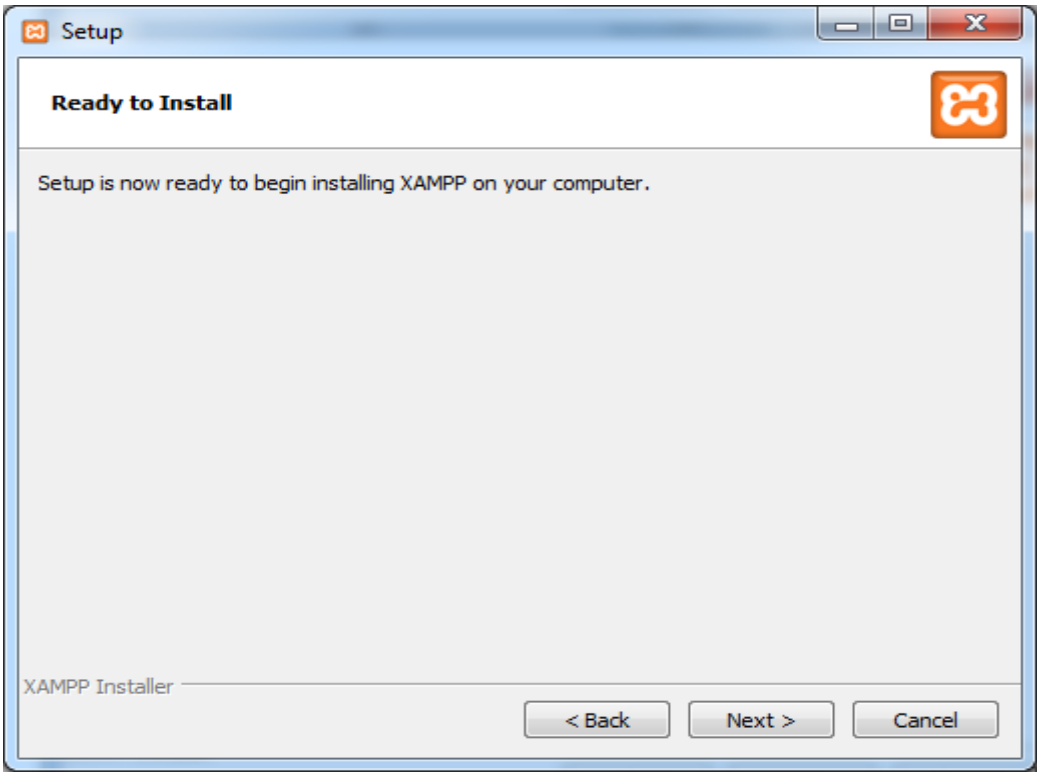
Εικόνα 1: Πατάω Next



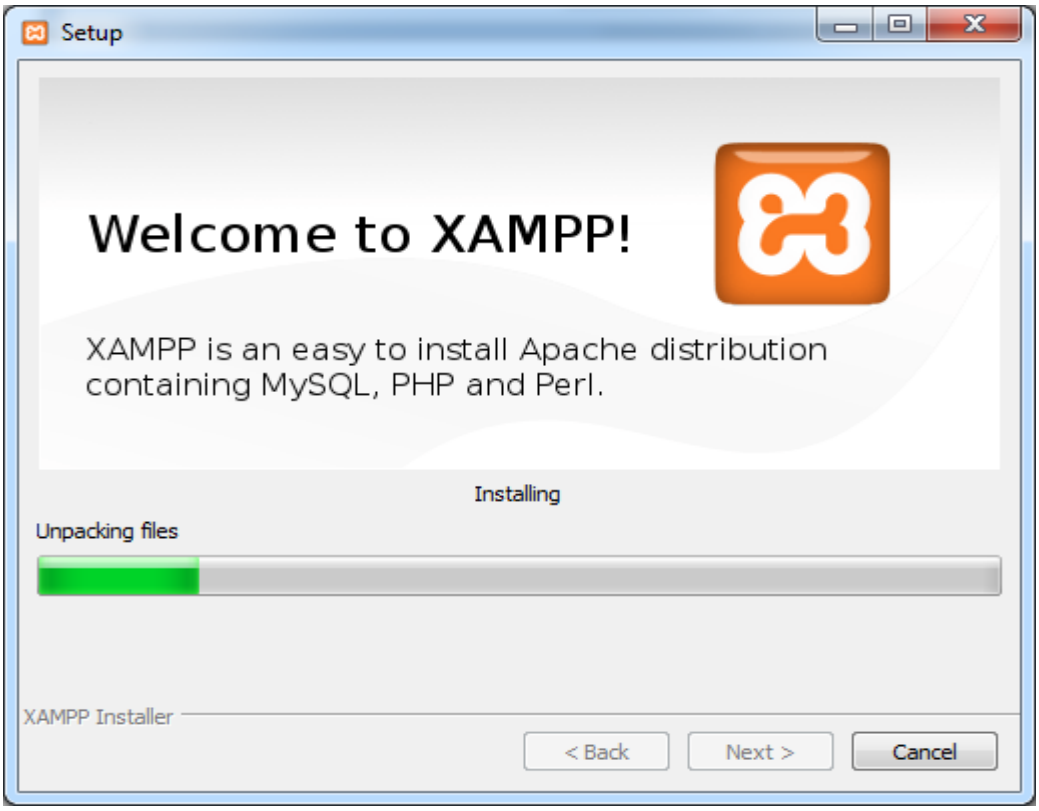
Εικόνα 2: Επιλογή Components



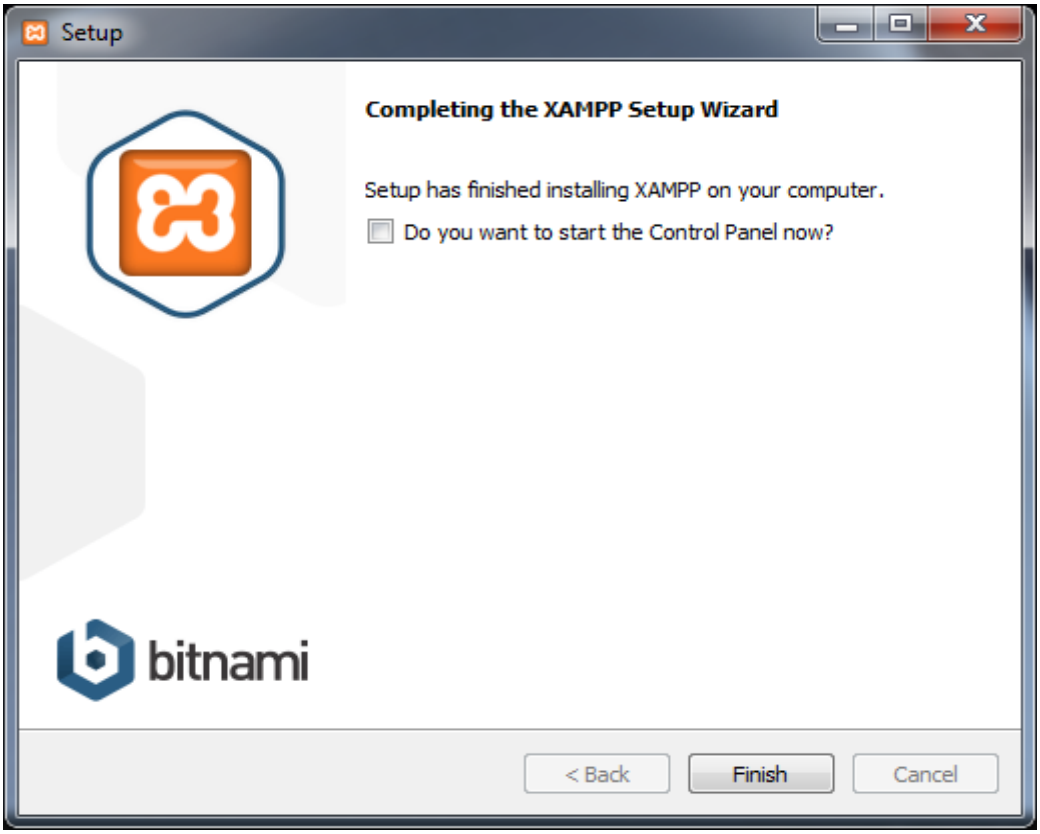
Εικόνα 3: Επιλογή προορισμού εγκατάστασης



Εικόνα 4: Έτοιμο για εγκατάσταση

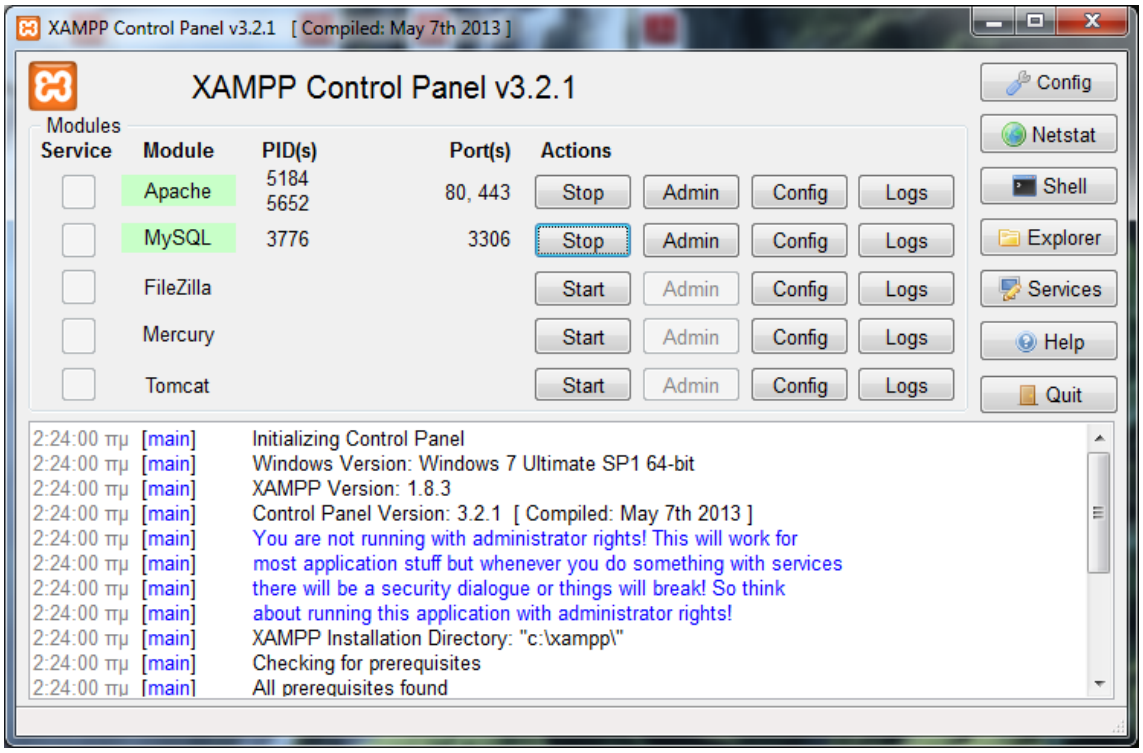


Εικόνα 5:Γίνεται η εγκατάσταση



Εικόνα 6:Τέλος εγκατάστασης

Αφού τελείωσε η εγκατάσταση του XAMPP,ανοίγω το XAMPP control panel και ενεργοποιώ τον Apache και την MySQL.

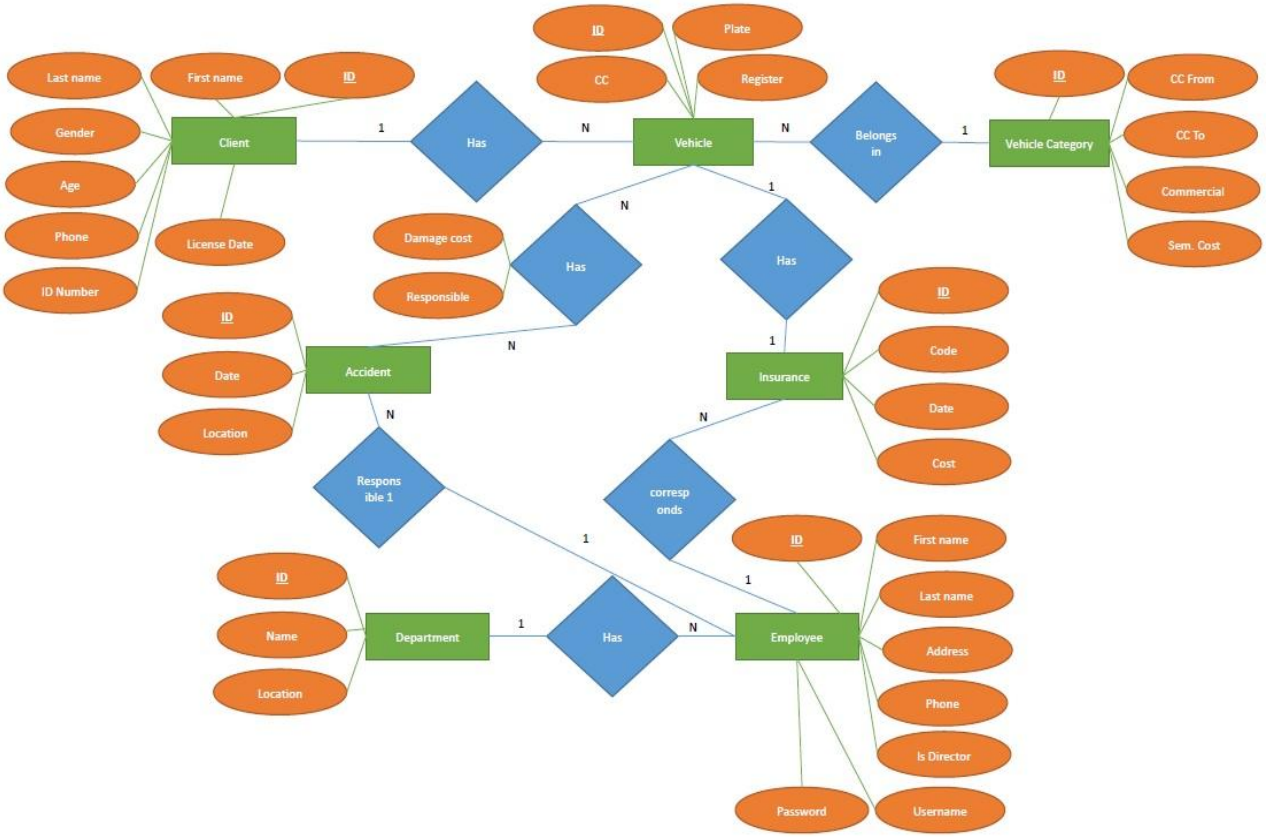


Εικόνα 7:Ενεργοποίηση Apache και MySQL

Πληκτρολογώντας στον browser localhost/xampp/index.php εμφανίζεται το εξής παράθυρο που μας επιβεβαιώνει την επιτυχή εγκατάσταση του XAMPP:

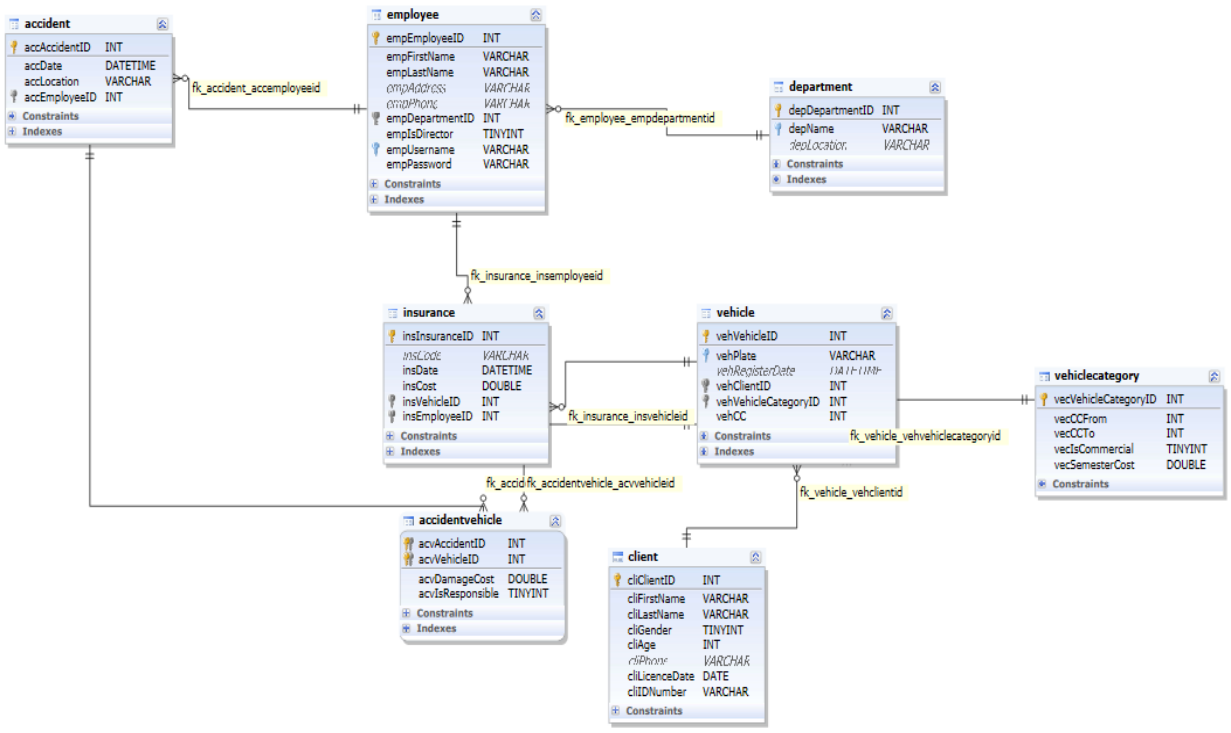
Εικόνα 8:Επιβεβαίωση επιτυχούς εγκατάστασης XAMPP

4 Μοντέλο οντοτήτων-συσχετίσεων



Εικόνα 9: ER-model

5 Σχεσιακό Μοντέλο



Εικόνα 10: Σχεσιακό Μοντέλο

6 Βάση δεδομένων

6.1 Δημιουργία των πινάκων τις βάσης

Με τις εντολές που ακολουθούν δημιουργώ τους πίνακες τις βάσης που προέκυψαν από το σχεσιακό μοντέλο.

Δομή πίνακα για τον πίνακα client:

```
CREATE TABLE client (
  cliClientID INT NOT NULL AUTO_INCREMENT,
  cliFirstName VARCHAR(50) NOT NULL,
  cliLastName VARCHAR(50) NOT NULL,
  cliGender BOOLEAN NOT NULL,
  cliAge INT NOT NULL,
  cliPhone VARCHAR(20) DEFAULT NULL,
  cliLicenceDate DATETIME NOT NULL,
  cliIDNumber VARCHAR(10) NOT NULL UNIQUE,
  PRIMARY KEY (cliClientID)
)
ENGINE = INNODB
CHARACTER SET utf8
COLLATE utf8_general_ci;
```

Δομή πίνακα για τον πίνακα department:

```
CREATE TABLE department (
  depDepartmentID INT NOT NULL AUTO_INCREMENT,
  depName VARCHAR(50) NOT NULL UNIQUE,
  depLocation VARCHAR(100) NULL,
  PRIMARY KEY (depDepartmentID)
)
ENGINE = INNODB
CHARACTER SET utf8
COLLATE utf8_general_ci;
```

Δομή πίνακα για τον πίνακα employee:

```
CREATE TABLE employee (
  empEmployeeID INT NOT NULL AUTO_INCREMENT,
  empFirstName VARCHAR(50) NOT NULL,
  empLastName VARCHAR(50) NOT NULL,
  empAddress VARCHAR(50) DEFAULT NULL,
  empPhone VARCHAR(20) DEFAULT NULL,
  empDepartmentID INT NOT NULL,
  empIsDirector BOOLEAN NOT NULL,
  empUsername VARCHAR(30) NOT NULL UNIQUE,
  empPassword VARCHAR(100) NOT NULL,
  PRIMARY KEY (empEmployeeID),
  CONSTRAINT fk_employee_empdepartmentid FOREIGN KEY (empDepartmentID)
    REFERENCES department(depDepartmentID) ON DELETE RESTRICT ON UPDATE
RESTRICT
)
ENGINE = INNODB
CHARACTER SET utf8
COLLATE utf8_general_ci;
```

Δομή πίνακα για τον πίνακα accident:

```
CREATE TABLE accident (
  accAccidentID INT NOT NULL AUTO_INCREMENT,
  accDate DateTime NOT NULL,
  accLocation VARCHAR(50) NOT NULL,
  accEmployeeID INT NOT NULL,
  PRIMARY KEY (accAccidentID),
  CONSTRAINT fk_accident_accemployeeid FOREIGN KEY (accEmployeeID)
    REFERENCES employee(empEmployeeID) ON DELETE RESTRICT ON UPDATE RESTRICT
)
ENGINE = INNODB
CHARACTER SET utf8
COLLATE utf8_general_ci;
```

Δομή πίνακα για τον πίνακα vehiclecategory:

```
CREATE TABLE vehiclecategory (
  vecVehicleCategoryID INT NOT NULL AUTO_INCREMENT,
  vecCCFrom INT NOT NULL,
  vecCCTo INT NOT NULL,
  vecIsCommercial BOOLEAN NOT NULL,
  vecSemesterCost DOUBLE NOT NULL,
  PRIMARY KEY (vecVehicleCategoryID)
)
ENGINE = INNODB
CHARACTER SET utf8
COLLATE utf8_general_ci;
```

Δομή πίνακα για τον πίνακα vehicle:

```
CREATE TABLE vehicle (
  vehVehicleID INT NOT NULL AUTO_INCREMENT,
  vehPlate VARCHAR(20) NOT NULL UNIQUE,
  vehRegisterDate DateTime NULL,
  vehClientID INT NOT NULL,
  vehVehicleCategoryID INT NOT NULL,
  vehCC INT NOT NULL,
  PRIMARY KEY (vehVehicleID),
  CONSTRAINT fk_vehicle_vehclientid FOREIGN KEY (vehClientID)
    REFERENCES client(cliClientID) ON DELETE RESTRICT ON UPDATE RESTRICT,
  CONSTRAINT fk_vehicle_vehvehiclecategoryid FOREIGN KEY (vehVehicleCategoryID)
    REFERENCES vehiclecategory(vecVehicleCategoryID) ON DELETE RESTRICT ON UPDATE
RESTRICT
)
ENGINE = INNODB
CHARACTER SET utf8
COLLATE utf8_general_ci;
```

Δομή πίνακα για τον πίνακα insurance:

```
CREATE TABLE insurance (
  insInsuranceID INT NOT NULL AUTO_INCREMENT,
  insCode VARCHAR(20) NULL,
```

```
  insDate DateTime NOT NULL,
  insCost DOUBLE NOT NULL,
  insVehicleID INT NOT NULL UNIQUE,
  insEmployeeID INT NOT NULL,
  PRIMARY KEY (insInsuranceID),
  CONSTRAINT fk_insurance_insvehicleid FOREIGN KEY (insVehicleID)
    REFERENCES vehicle(vehVehicleID) ON DELETE RESTRICT ON UPDATE RESTRICT,
  CONSTRAINT fk_insurance_insemployeeid FOREIGN KEY (insEmployeeID)
    REFERENCES employee(empEmployeeID) ON DELETE RESTRICT ON UPDATE RESTRICT
)
ENGINE = INNODB
CHARACTER SET utf8
COLLATE utf8_general_ci;
```

Δομή πίνακα για τον πίνακα accidentvehicle:

```
acvAccidentID INT NOT NULL,
  acvVehicleID INT NOT NULL,
  acvDamageCost DOUBLE NOT NULL,
  acvIsResponsible BOOLEAN NOT NULL,
  PRIMARY KEY (acvAccidentID, acvVehicleID),
  CONSTRAINT fk_accidentvehicle_acvaccidentid FOREIGN KEY (acvAccidentID)
    REFERENCES accident(accAccidentID) ON DELETE RESTRICT ON UPDATE RESTRICT,
  CONSTRAINT fk_accidentvehicle_acvvehicleid FOREIGN KEY (acvVehicleID)
    REFERENCES vehicle(vehVehicleID) ON DELETE RESTRICT ON UPDATE RESTRICT
)
ENGINE = INNODB
CHARACTER SET utf8
COLLATE utf8_general_ci;
```

6.2 Περιγραφή των πινάκων τις βάσης

Πίνακας "client":
Αποθηκεύει τους πελάτες (εταιρικούς ή μη).
- ***cliClientID*** :Μοναδικό αναγνωριστικό πελάτη (πρωτεύων κλειδί).
- ***cliFirstName, cliLastName***: Ονοματεπώνυμο πελάτη (υποχρεωτικό).
- ***cliGender, cliAge***: Φύλο/ηλικία πελάτη (υποχρεωτικά).
- ***cliPhone***: Τηλέφωνο (όχι υποχρεωτικό)
- ***cliLicenceDate***: Ημερομηνία απόκτησης διπλώματος (υποχρεωτικό). Χρησιμοποιείται κυρίως για να ελεγχθεί συνθήκη νέου οδηγού (6 μήνες από αυτή την ημερομηνία θεωρείται νέος).
- ***cliIDNumber***: Αριθμός ταυτότητας (υποχρεωτικό και μοναδικό).

Πίνακας "department":
Αποθηκεύει τα τμήματα της εταιρείας.
- ***depDepartmentID*** : Μοναδικό αναγνωριστικό τμήματος (πρωτεύων κλειδί).
- ***depName***: Ονομασία τμήματος (υποχρεωτικό και μοναδικό).
- **depLocation**: Τοποθεσία τμήματος (όχι υποχρεωτικό).

Πίνακας "employee":
Αποθηκεύει τους υπαλλήλους εταιρείας.
- **empEmployeeID** : Μοναδικό αναγνωριστικό υπαλλήλου (πρωτεύων κλειδί).
- ***empFirstName, empLastName***: Ονοματεπώνυμο υπαλλήλου (υποχρεωτικό).
- ***empAddress, empPhone***: Διεύθυνση, τηλέφωνο υπαλλήλου (όχι υποχρεωτικά).
- ***empIsDirector***: Εάν είναι διευθυντής τμήματος (υποχρεωτικό).
- ***empUsername***: Username υπαλλήλου για σύνδεση στη web-εφαρμογή (υποχρεωτικό και μοναδικό).
- ***empPassword:*** Password υπαλλήλου για σύνδεση στη web-εφαρμογή (υποχρεωτικό).

- ***empDepartmentID***: Ξένο κλειδί προς το τμήμα (depDepartmentID) στο οποίο ανήκει ο υπάλληλος.

Πίνακας "vehiclecategory":
Αποθηκεύει τις κατηγορίες οχημάτων.
- ***vecVehicleCategoryID*** :Μοναδικό αναγνωριστικό κατηγορίας (πρωτεύων κλειδί).
- ***vecCCFrom, vecCCTo***: Εύρος κυβικών εκατοστών κινητήρα από-έως (υποχρεωτικά).
- ***vecIsCommercial*:** Εάν είναι επαγγελματική ή όχι η κατηγορία(υποχρεωτικό).
-***vecSemesterCost***: Κόστος κατηγορίας ανά εξάμηνο (υποχρεωτικό). Χρησιμοποιείται στον υπολογισμό κόστους ασφαλιστηρίου κατά τη διάρκεια του πρώτου χρόνου.

Πίνακας "vehicle":
Αποθηκεύει τα οχήματα (εταιρικά και μη).
- ***vehVehicleID*** :Μοναδικό αναγνωριστικό οχήματος (πρωτεύων κλειδί).
- ***vehPlate***: Πινακίδα οχήματος (υποχρεωτικό και μοναδικό).
- ***vehCC:*** Κυβικά εκατοστά κινητήρα οχήματος (υποχρεωτικό).
- ***vehRegisterDate***: Ημερομηνία καταχώρισης οχήματος στην εταιρεία (μη υποχρεωτικό). Εάν το πεδίο αυτό είναι κενό, τότε το αυτοκίνητο είναι μη εταιρικό, άρα αυτό συνυπολογίζεται σε περίπτωσης απόδοσης ευθύνης και εξόδων της εταιρείας.
- ***vehClientID***: Ξένο κλειδί προς τον πελάτη (cliClientID) στον οποίο ανήκει το όχημα.
- ***vehVehicleCategoryID***: Ξένο κλειδί προς την κατηγορία οχήματος (vecVehicleCategoryID) στην οποία ανήκει το όχημα βάσει των κυβικών εκατοστών του.

Πίνακας "insurance":
Αποθηκεύει τα ασφαλιστήρια (συμβόλαια).
- ***insInsuranceID*** :Μοναδικό αναγνωριστικό ασφαλιστηρίου (πρωτεύων κλειδί).
- ***insCode***: Κωδικός ασφαλιστηρίου (υποχρεωτικό).
- ***insDate***: Ημερομηνία έναρξης ασφαλιστηρίου (υποχρεωτικό).
- ***insCost***: Ετήσιο κόστος ασφαλιστηρίου (υποχρεωτικό). Παίζει ρόλο στον υπολογισμό κόστους από το δεύτερο χρόνο και μετά.
- ***insVehicleID***: Ξένο κλειδί προς το όχημα (vehVehicleID) του ασφαλιστηρίου. Το πεδίο είναι επίσης και μοναδικό, δηλαδή δε μπορεί να υπάρξει διπλότυπο insVehicleID μέσα στον πίνακα, άρα όχημα συνδεδεμένο με ασφαλιστήριο παραπάνω από μια φορά.
- ***insEmployeeID***: Ξένο κλειδί προς τον υπάλληλο (empEmployeeID) που δημιούργησε και χειρίζεται το ασφαλιστήριο.

Πίνακας "accident":
Αποθηκεύει τα ατυχήματα.
- ***accAccidentID*** : Μοναδικό αναγνωριστικό ατυχήματος (πρωτεύων κλειδί).
- ***accLocation***: Τοποθεσία ατυχήματος (υποχρεωτικό).
- ***accDate***: Ημερομηνία-ώρα που συνέβη το ατύχημα (υποχρεωτικό).
- ***accEmployeeID***: Ξένο κλειδί προς τον υπάλληλο (empEmployeeID) που κατέγραψε και χειρίζεται το ατύχημα.

Πίνακας "accidentvehicle":
Αποθηκεύει τα οχήματα που εμπλέκονται σε ατυχήματα.
- ***acvAccidentID***: Ξένο κλειδί προς το ατύχημα (accAccidentID).
- ***acvVehicleID***:Ξένο κλειδί προς το όχημα (vehVehicleID) που εμπλέκεται στο συγκεκριμένο ατύχημα.
Τα δύο πρώτα πεδία είναι διπλό συνδυαστικό κλειδί, δηλαδή το ζευγάρι ατύχημα, όχημα είναι μοναδικό.
- ***acvIsResponsible*** : Εάν το όχημα φέρει ευθύνη (υποχρεωτικό)
- ***acvDamageCost***: Κόστος ζημιάς (υποχρεωτικό)

7 Λειτουργικότητα του συστήματος

7.1 Σύντομη περιγραφή του συστήματος

Στο συγκεκριμένο σύστημα το γραφικό κομμάτι (διεπαφές/φόρμες/πίνακες) που βλέπουν οι χρήστες, είναι εξ' ολοκλήρου υλοποιημένο με την JavaScript βιβλιοθήκη ExtJs την οποία κατέβασα από εδώ: http://www.sencha.com/legal/GPL/
Το γραφικό κομμάτι επικοινωνεί με Ajax calls με τις php σελίδες οι οποίες βρίσκονται στον φάκελο PHP . Όλες οι σελίδες php κάνουν ένα κοινό script το PHP/inc/data.php το οποίο και επικοινωνεί με τη MySQL και επιστρέφει/εισάγει/διαγράφει/επεξεργάζεται δεδομένα. Στο αρχείοPhp/inc/confing.php υπάρχουν τα στοιχεία σύνδεσης με την βάση, καθώς και το adminusername και adminpassword για την προστασία της σελίδας http://localhost/carinsurance/Admin.php που πρέπει να χρησιμοποιήσει κάποιος για να κάνει είσοδο σε αυτή. Η σελίδα προσφέρει κάποιες λειτουργείς που χρειάζονται και θα περιγραφούν παρακάτω. Στο αρχείο DbScripts/CreateDB.sql υπάρχει ο κώδικας για την υλοποίηση των συγκεντρωτικών αναφορών. Στο συγκεκριμένο σύστημα δεν υπάρχει διαχωρισμός ρόλων.

7.2 Λειτουργίες που υποστηρίζει το σύστημα

Το σύστημα υποστηρίζει τις εξής λειτουργίες:

- Login-Logout
- Δημιουργία-Τροποποίηση-Διαγραφή Τμήματος.
- Δημιουργία-Τροποποίηση-Διαγραφή Υπαλλήλου.
- Δημιουργία-Τροποποίηση-Διαγραφή Κατηγορίας οχήματος.
- Δημιουργία-Τροποποίηση-Διαγραφή Πελατών(εταιρικών και μη).
- Δημιουργία-Τροποποίηση-Διαγραφή Οχημάτων(εταιρικών και μη).
- Δημιουργία-Τροποποίηση-Διαγραφή Ατυχημάτων.
- Δημιουργία-Τροποποίηση-Διαγραφή Ασφαλιστηρίων.
- Συγκεντρωτική αναφορά για τα έσοδα και έξοδα κάθε μήνα ανά πελάτη.
- Συγκεντρωτική αναφορά για τα έσοδα και έξοδα κάθε μήνα ανά κατηγορία οχήματος, ανά φύλο και ανά ηλικία.
- Συγκεντρωτική αναφορά τα ατυχήματα αν φύλλο πελατών και αν ηλικία.
- Συγκεντρωτική αναφορά για τα έσοδα, έξοδα και συμβόλαια που κάνει κάθε υπάλληλος/τμήμα σε ένα συγκεκριμένο χρονικό διάστημα.

Για να κάνει κάποιος είσοδο στην σελίδα http://localhost/carinsurance/Admin.php χρησιμοποιεί τα στοιχεία που αναφέρονται στην προηγούμενη υποενότητα. Σε αυτή την σελίδα δίνονται οι εξής δυνατότητες:

- Logout
- Δημιουργία-Τροποποίηση-Διαγραφή Τμήματος.
- Δημιουργία-Τροποποίηση-Διαγραφή Υπαλλήλου.
- Δημιουργία-Τροποποίηση-Διαγραφή Κατηγορίας οχήματος.

Στην σελίδα http://localhost/carinsurance/Home.php δίνονται οι εξής δυνατότητες:

- Logout
- Δημιουργία-Τροποποίηση-Διαγραφή Πελατών(εταιρικών και μη).
- Δημιουργία-Τροποποίηση-Διαγραφή Οχημάτων(εταιρικών και μη).
- Δημιουργία-Τροποποίηση-Διαγραφή Ατυχημάτων.
- Δημιουργία-Τροποποίηση-Διαγραφή Ασφαλιστηρίων.
- Συγκεντρωτική αναφορά για τα έσοδα και έξοδα κάθε μήνα ανά πελάτη.
- Συγκεντρωτική αναφορά για τα έσοδα και έξοδα κάθε μήνα ανά κατηγορία οχήματος, ανά φύλο και ανά ηλικία.
- Συγκεντρωτική αναφορά τα ατυχήματα αν φύλλο πελατών και αν ηλικία.
- Συγκεντρωτική αναφορά για τα έσοδα, έξοδα και συμβόλαια που κάνει κάθε υπάλληλος/τμήμα σε ένα συγκεκριμένο χρονικό διάστημα.

Στην σελίδα αυτή μπορεί να μπει κάποιος σαν υπάλληλος με το username και το password που του έχει δοθεί κατά την δημιουργία ή την τροποποίηση υπαλλήλου στην σελίδα:
http://localhost/carinsurance/Admin.php

Σημείωση

Ο κάθε υπάλληλος μπορεί να κάνει Τροποποίηση-Διαγραφή μόνο όσων ατυχημάτων και ασφαλιστηρίων έχει Δημιουργήσει αυτός και όχι οι υπόλοιποι υπάλληλοι.

Τέλος δημιουργήθηκε η σελίδα http://localhost/carinsurance/index.php που έχει την εξής δυνατότητα:

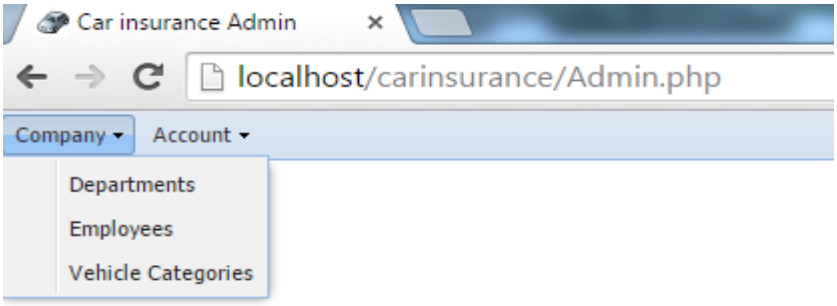
- Login

για να μπορώ να επιλέξω σε ποια από τις δύο προηγούμενες σελίδες θέλω να κάνω είσοδο.

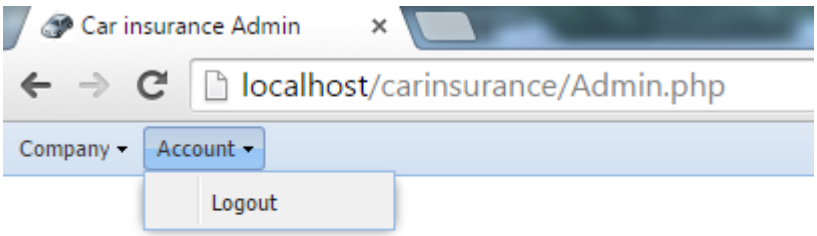
7.3 Σελίδα Intex.php

-Επιλέγω σε ποια σελίδα θέλω να κάνω είσοδο.
-Πληκτρολογώ το username και password.
-Αν ο συνδυασμός username και password είναι σωστός γίνεται είσοδος στην αντίστοιχη σελίδα.

7.4 Σελίδα Admin.php



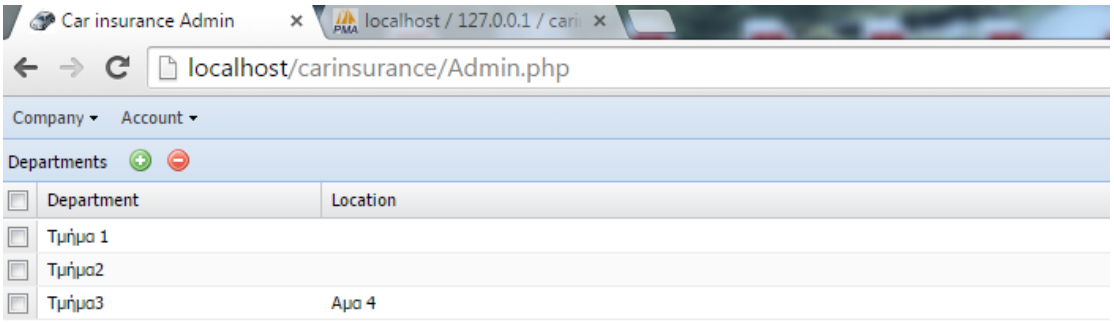
Εικόνα 11:Σελίδα Admin.php



Εικόνα 12:Logout από Admin.php

7.4.1 Δημιουργία -Τροποποίηση- Διαγραφή τμήματος

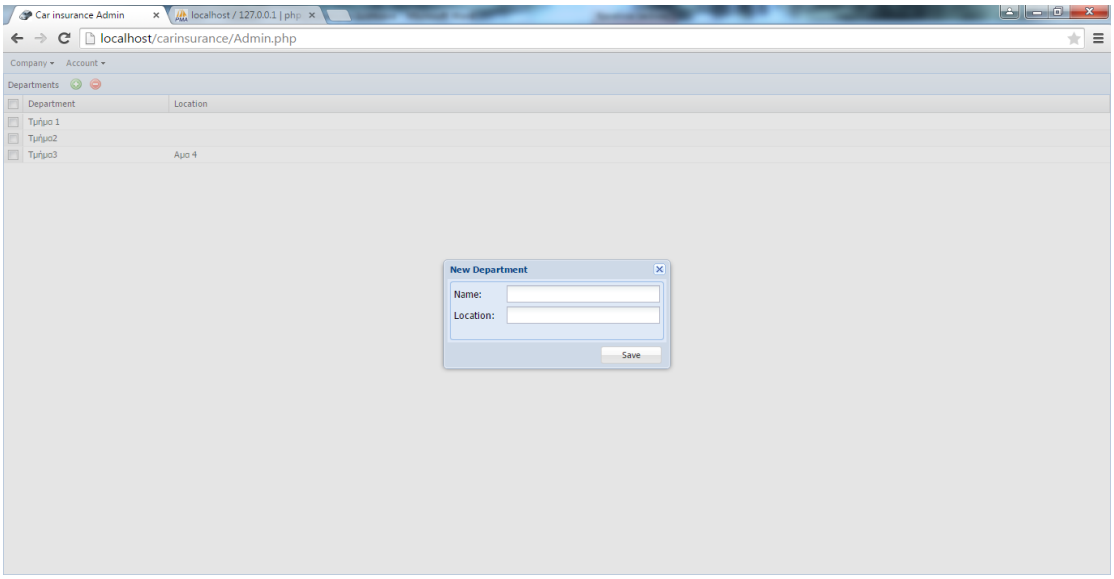
Στην αρχική οθόνη των τμημάτων εμφανίζεται η λίστα με τα υπάρχοντα τμήματα και το + για την δημιουργία και το – για την διαγραφή όπως φαίνετε στο παρακάτω screenshots:



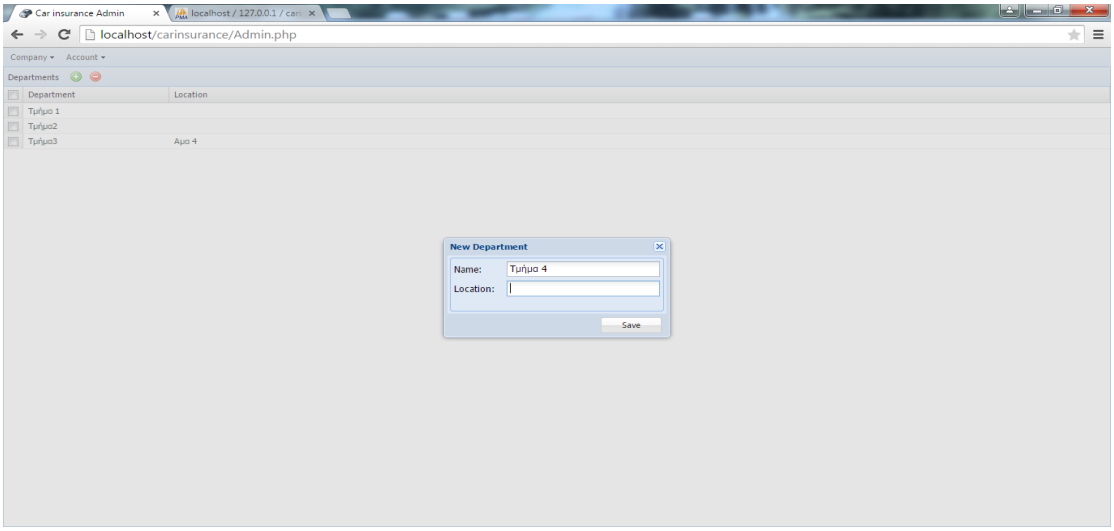
Εικόνα 13:Αρχική οθόνη τμημάτων

Δημιουργία τμήματος

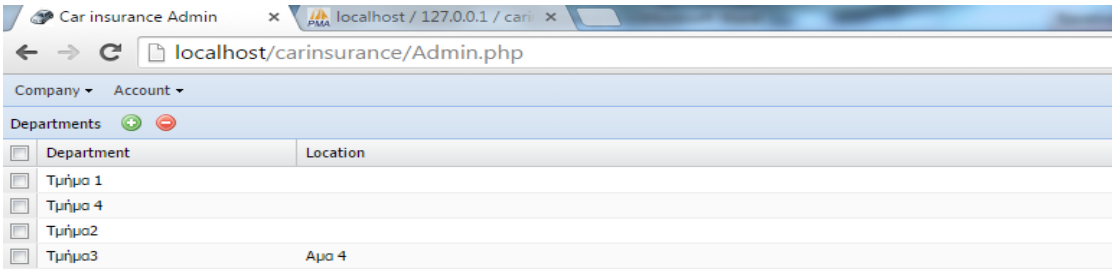
Για την δημιουργία του τμήματος πρέπει να επιλεγεί το + ,έτσι ανοίγει η φόρμα για την δημιουργία του τμήματος. Πρέπει να πληκτρολογηθούν τα στοιχεία και μετά να πατηθεί το save.



Εικόνα 14:Φόρμα εισαγωγής τμήματος



Εικόνα 15:Εισαγωγή στοιχείων για την δημιουργία τμήματος

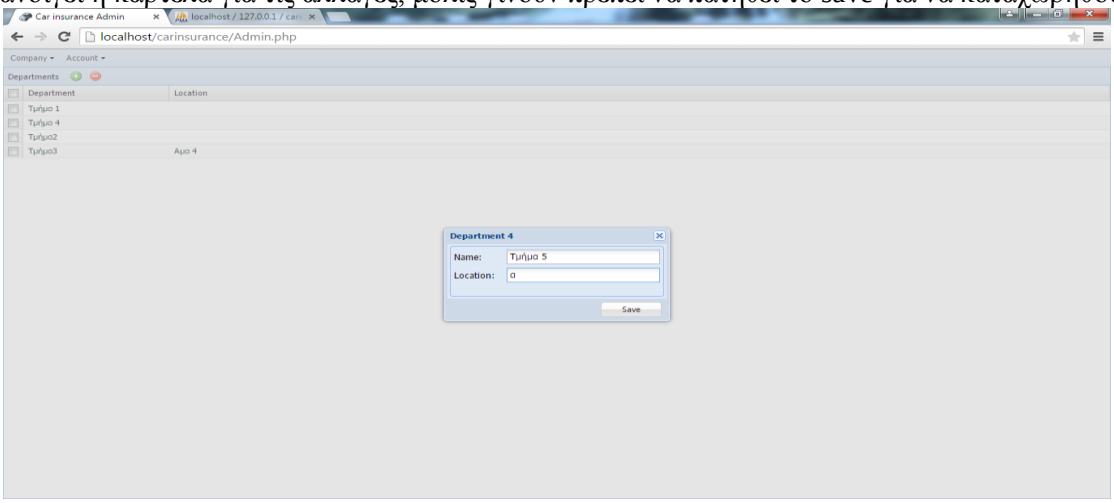


Εικόνα 16:Εμφάνιση νέου τμήματος

Όπως φαίνετε στο παραπάνω screenshots το τμήμα εμφανίζεται πλέον στην λίστα με τα τμήματα.

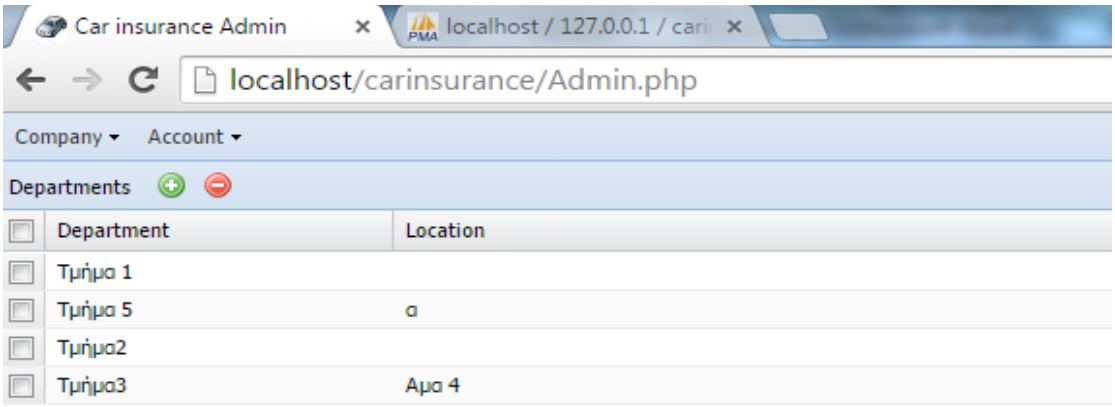
Τροποποίηση τμήματος

Για την τροποποίηση του τμήματος πρέπει να γίνει κλικ στο επιθυμητό τμήμα για τροποποίηση, ανοίγει η καρτέλα για τις αλλαγές, μόλις γίνουν πρέπει να πατηθεί το save για να καταχωρηθούν.



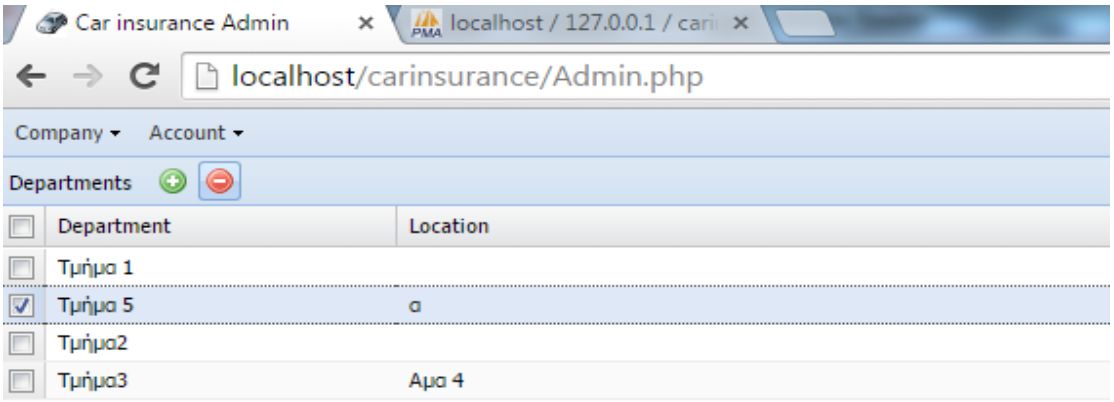
Εικόνα 17:Τροποποίηση τμήματος

Στην θέση του παλιού εμφανίζετε πλέον το νέο τμήμα μετά την τροποποίηση.

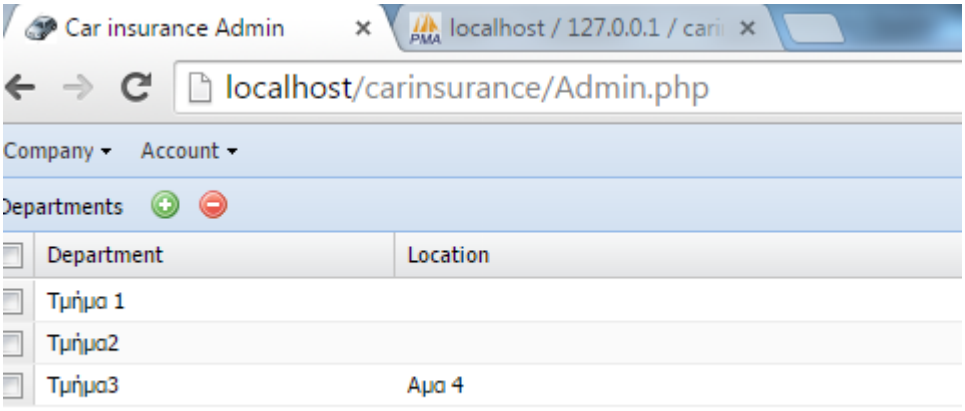


Διαγραφή τμήματος

Για την διαγραφή του τμήματος πρέπει να επιλεγεί το τμήμα προς διαγραφή και μετά να πατηθεί το - .

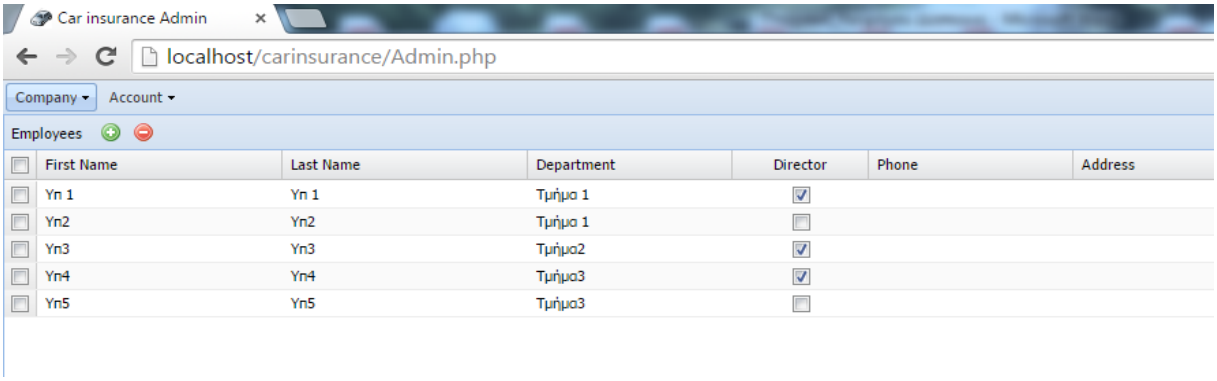


Εικόνα 18: Διαγραφή τμήματος



7.4.2 Δημιουργία-Τροποποίηση-Διαγραφή Υπαλλήλου

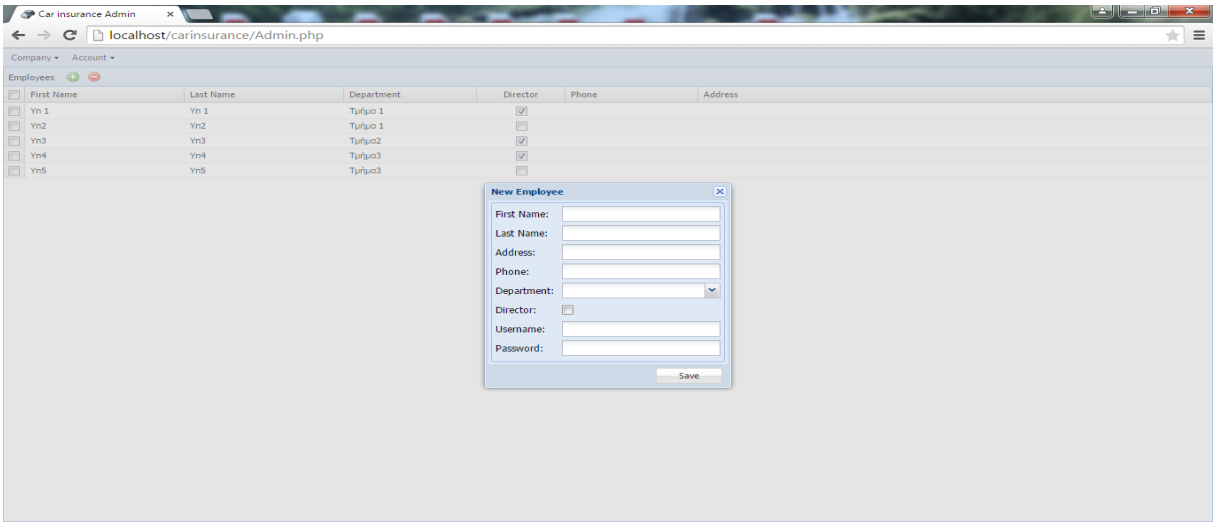
Στην αρχική οθόνη των υπαλλήλων εμφανίζεται η λίστα με τους υπάρχοντες υπαλλήλους και το + για την δημιουργία και το – για την διαγραφή όπως φαίνετε στο παρακάτω screenshots:



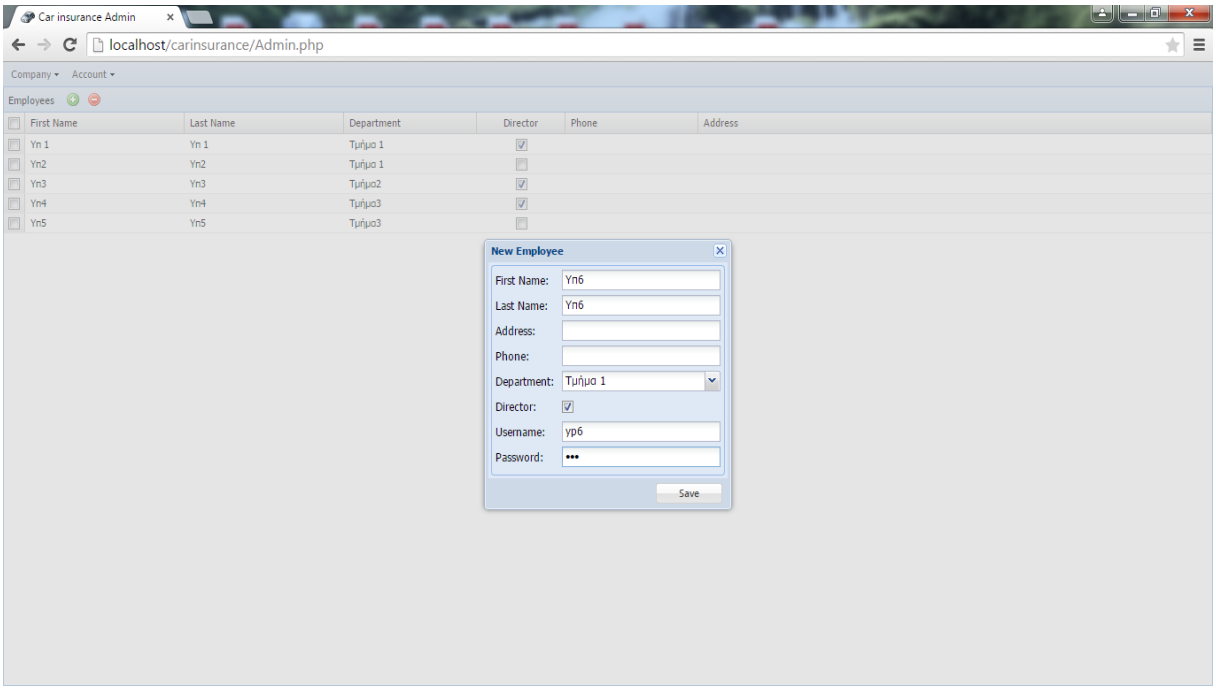
Εικόνα 19:Αρχική οθόνη υπαλλήλων

Δημιουργία υπαλλήλου

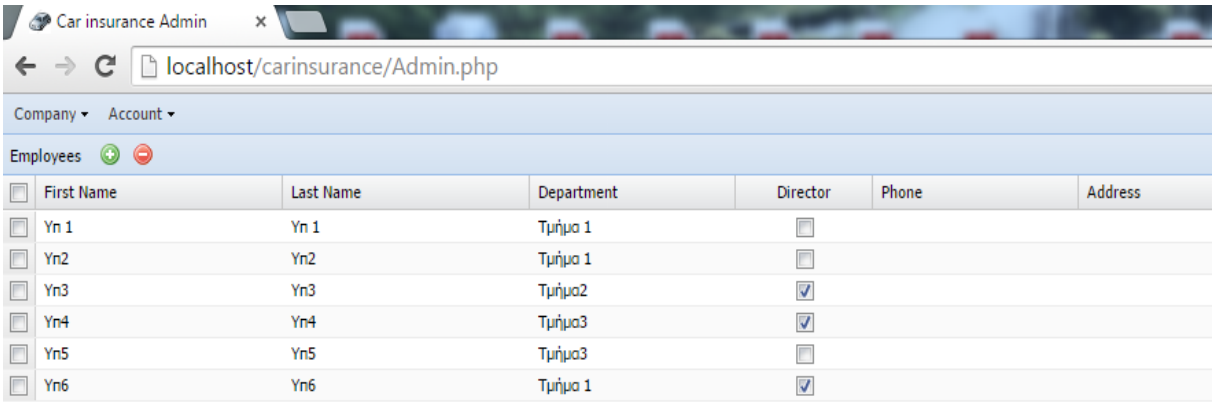
Για την δημιουργία του υπαλλήλου πρέπει να επιλεγεί το + ,έτσι ανοίγει η φόρμα για την δημιουργία του υπαλλήλου. Πρέπει να πληκτρολογηθούν τα στοιχεία και μετά να πατηθεί το save .



Εικόνα 20:Φόρμα δημιουργίας υπαλλήλου



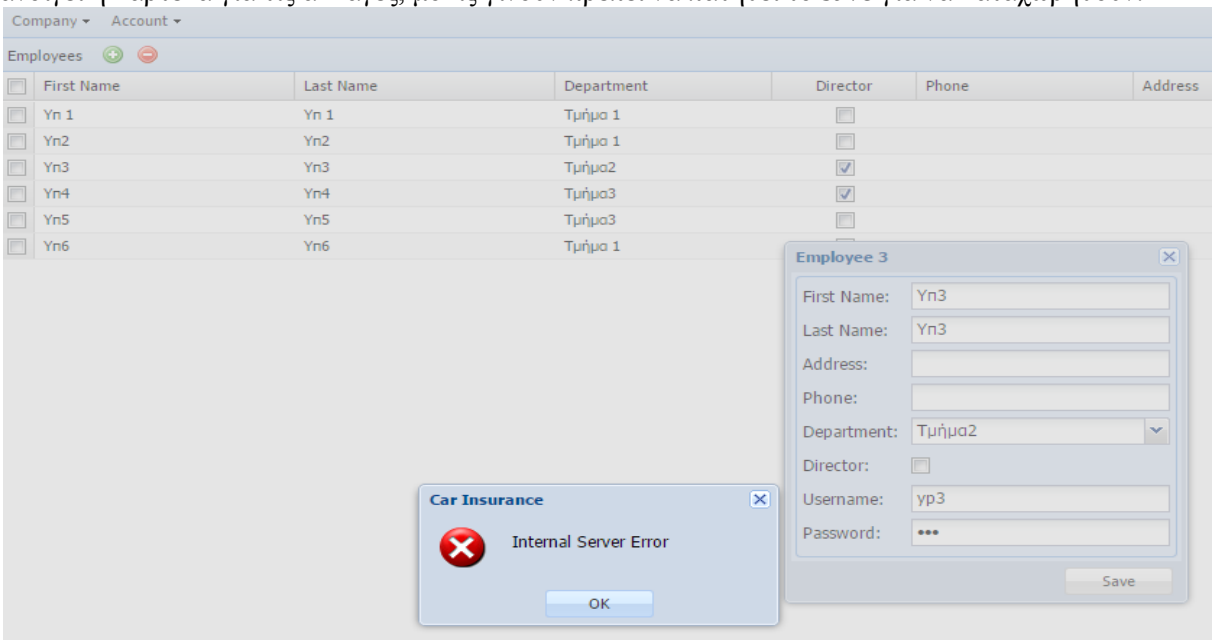
Εικόνα 21:Εισαγωγή στοιχείων για την δημιουργία υπαλλήλου



Όπως φαίνετε στο παραπάνω screenshots ο υπάλληλος εμφανίζεται πλέον στην λίστα με τους υπαλλήλους. Επίσης επειδή ορίστηκε ως director είναι πλέον αυτός ο διευθυντής του τμήματος 1 και όχι ο Υπ1 που ήταν πριν.

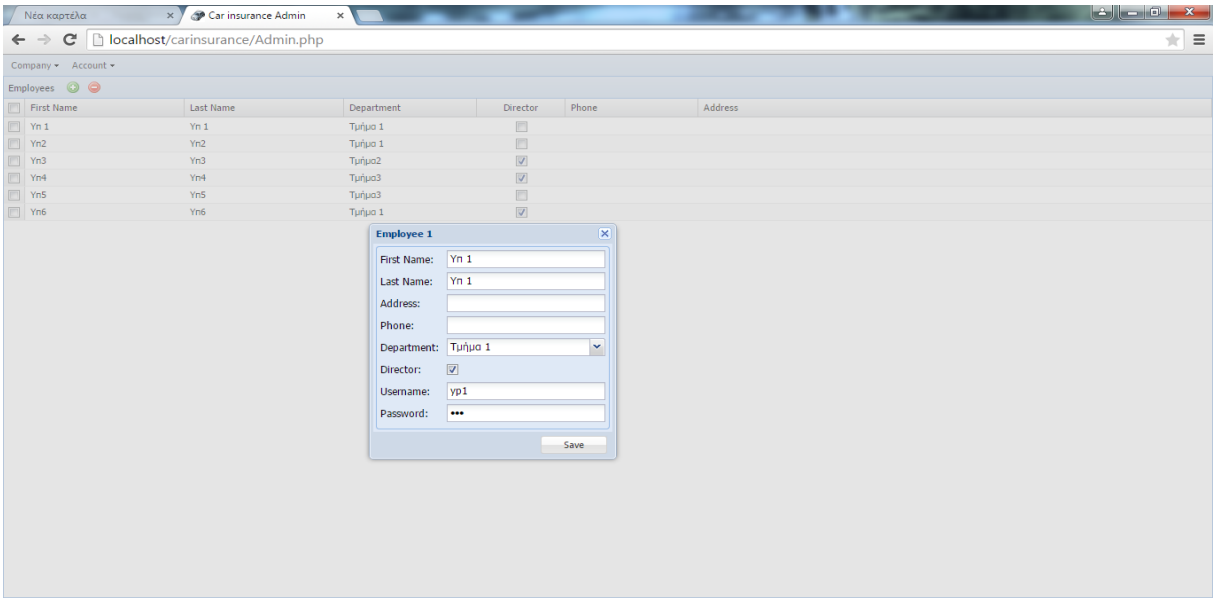
Τροποποίηση υπαλλήλου

Για την τροποποίηση του υπαλλήλου πρέπει να γίνει κλικ στον επιθυμητό υπάλληλο για τροποποίηση, ανοίγει η καρτέλα για τις αλλαγές, μόλις γίνουν πρέπει να πατηθεί το save για να καταχωρηθούν.



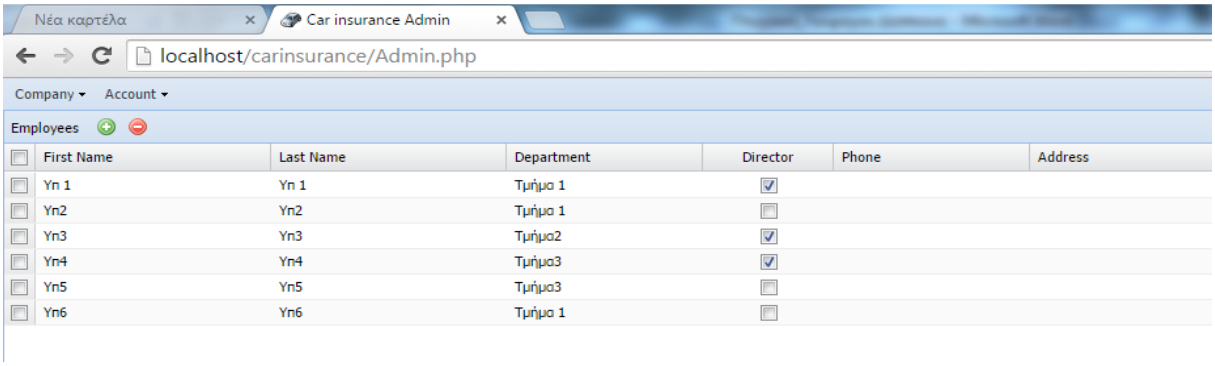
Εικόνα 22:Ανεπιτυχής τροποποίηση

Η παραπάνω τροποποίηση ήταν ανεπιτυχής γιατί πήγα να βάλω τον Υπ3 να μην είναι πλέον διευθυντής και πετάει σφάλμα γιατί δεν γίνεται κάποιο τμήμα να μην έχει διευθυντή.



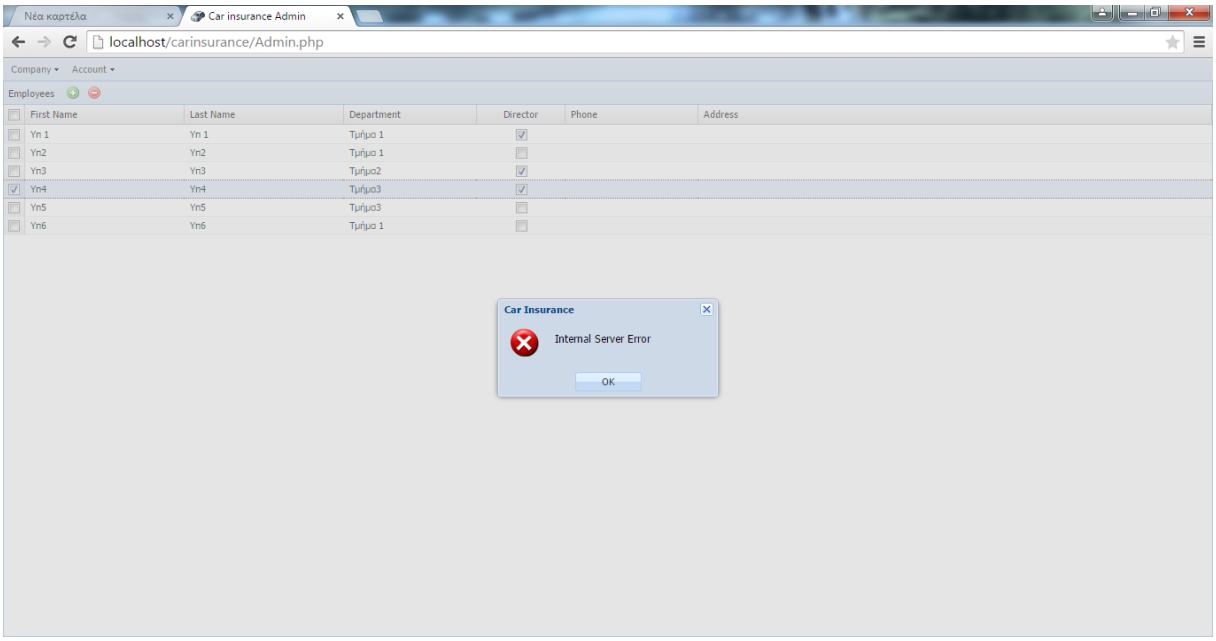
Εικόνα 23:Τροποποίηση Υπαλλήλου

Ορισμός ξανά του Υπ1 ως διευθυντή.



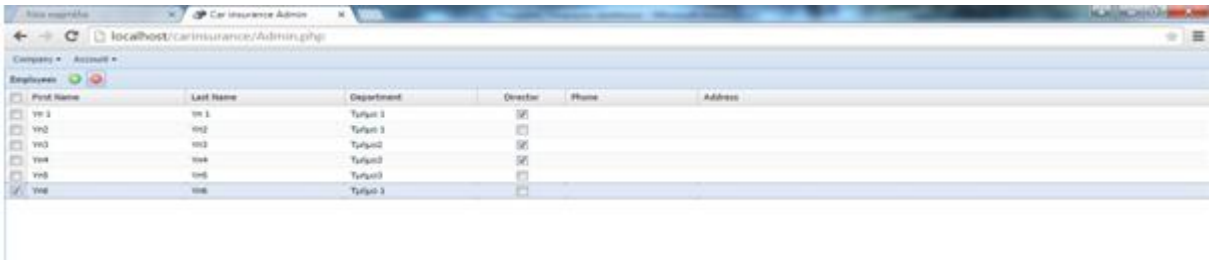
Διαγραφή υπαλλήλου

Για την διαγραφή του υπαλλήλου πρέπει να επιλεγεί ο υπάλληλος προς διαγραφή και μετά να πατηθεί το - .

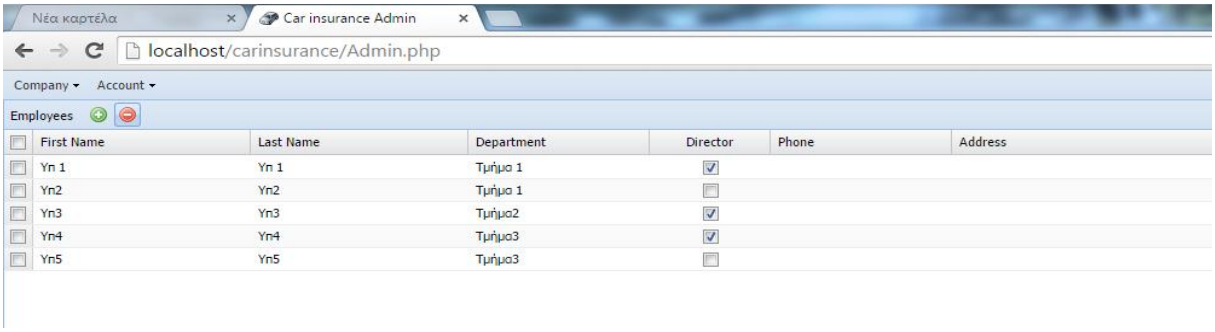


Εικόνα 24:Ανεπιτυχής διαγραφή υπαλλήλου

Η παραπάνω διαγραφή ήταν ανεπιτυχής γιατί πήγα να διαγράψω τον διευθυντή τμήματος. Θα ήταν επιτυχής αν είχε οριστεί πρώτα κάποιος άλλος ως διευθυντής του συγκεκριμένου τμήματος και μετά γινόταν η διαγραφή.

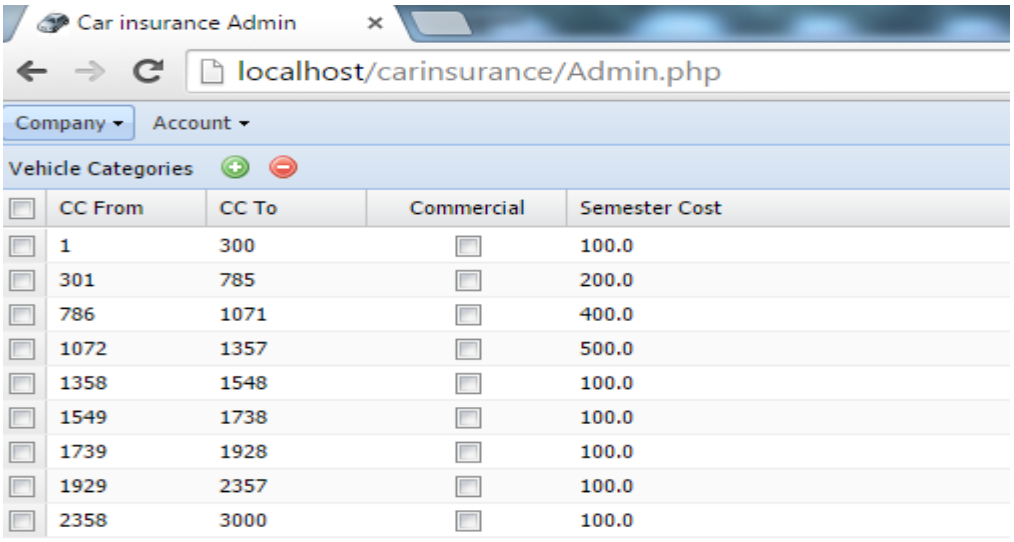


Εικόνα 25:Διαγραφή Υπ6



7.4.3 Δημιουργία-Τροποποίηση-Διαγραφή Κατηγορίας οχήματος

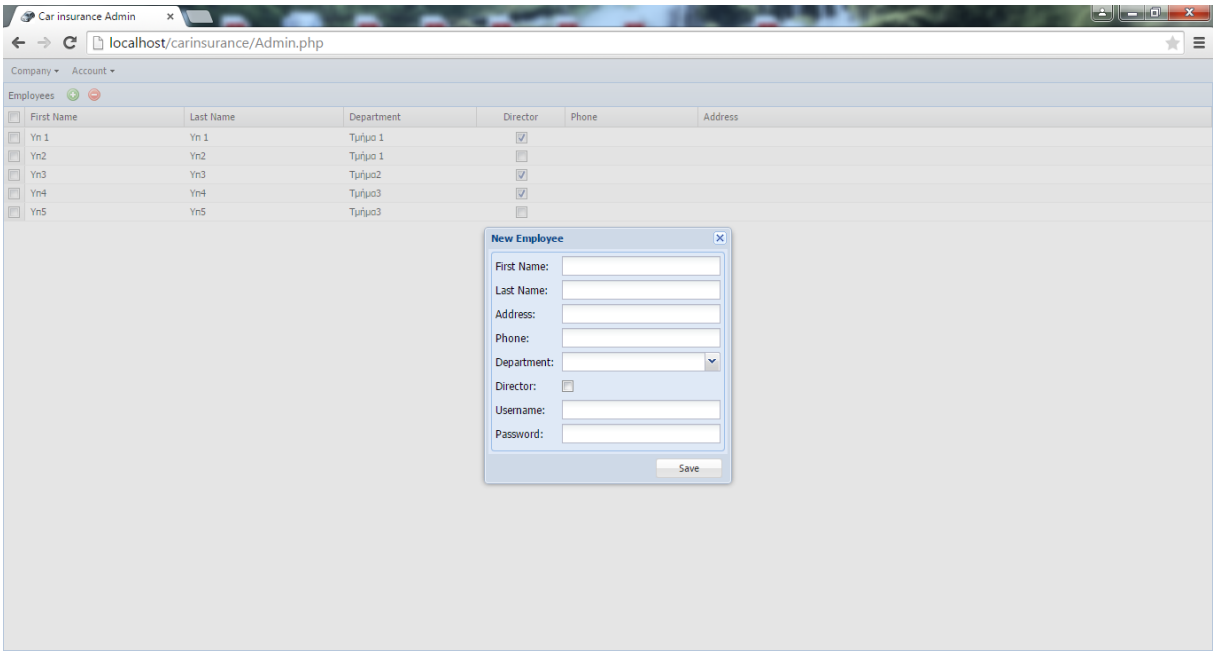
Στην αρχική οθόνη των κατηγοριών οχημάτων εμφανίζεται η λίστα με τις υπάρχουσες κατηγορίες αυτοκινήτων και το + για την δημιουργία και το – για την διαγραφή όπως φαίνετε στο παρακάτω screenshots:



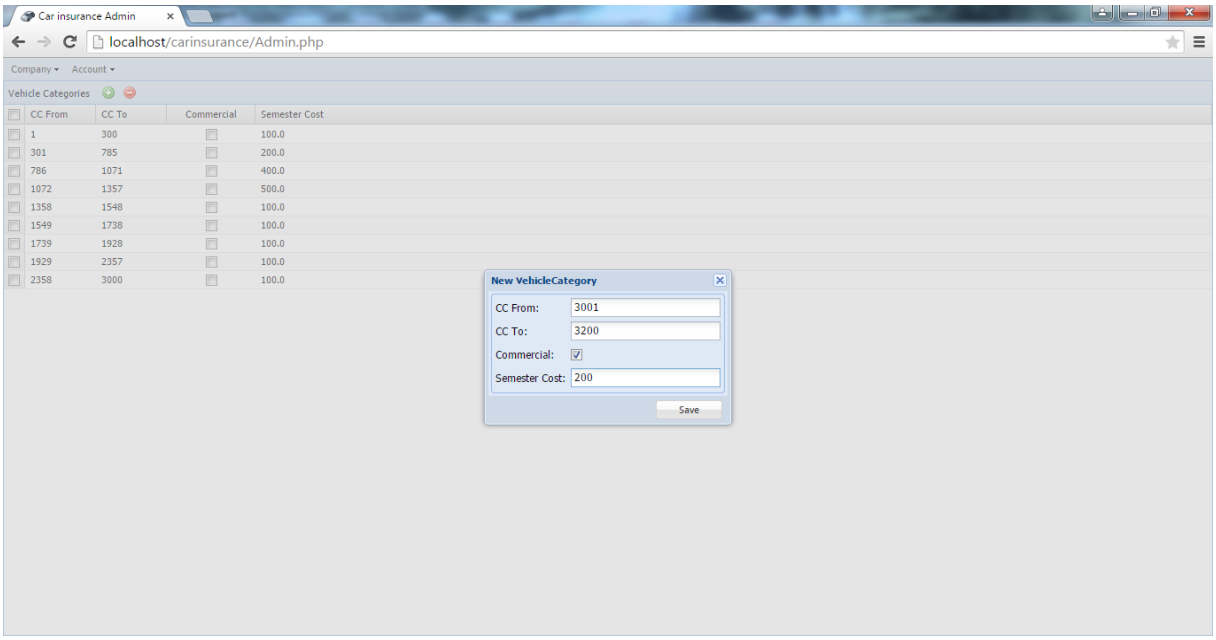
Εικόνα 26:Αρχική οθόνη κατηγοριών οχημάτων

Δημιουργία κατηγορίας οχημάτων

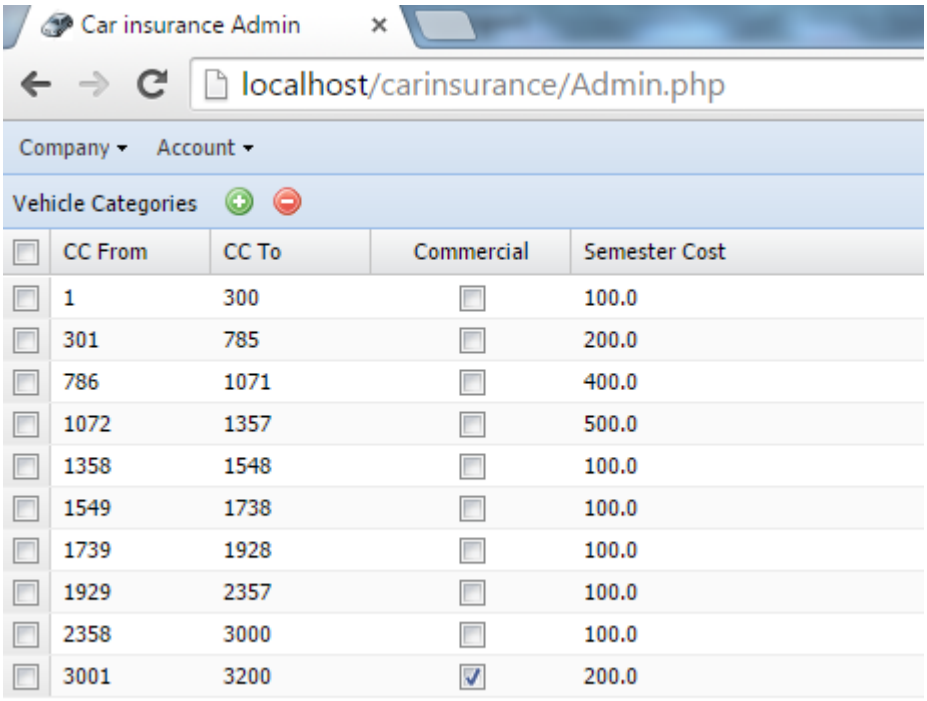
Για την δημιουργία κατηγορίας αυτοκινήτου πρέπει να επιλεγεί το + ,έτσι ανοίγει η φόρμα για την δημιουργία κατηγορίας αυτοκινήτου. Πρέπει να πληκτρολογηθούν τα στοιχεία και μετά να πατηθεί το save.



Εικόνα 27:Φόρμα δημιουργίας κατηγορίας οχημάτων



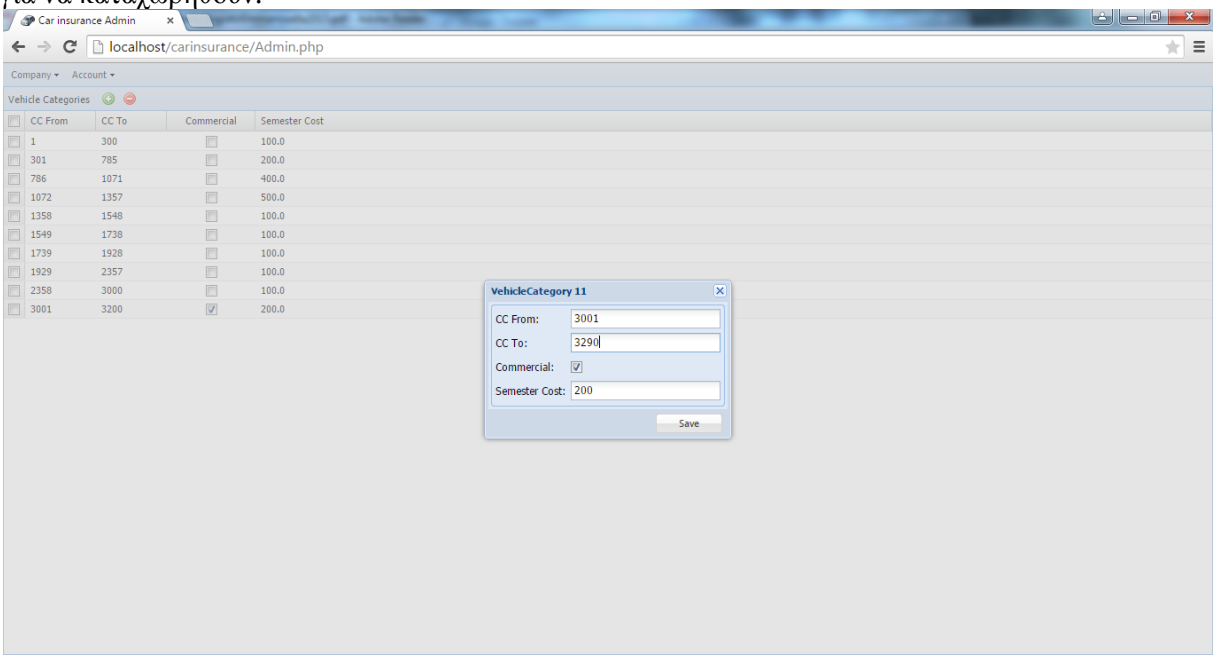
Εικόνα 28:Εισαγωγή στοιχείων στην φόρμα κατηγορίας οχημάτων



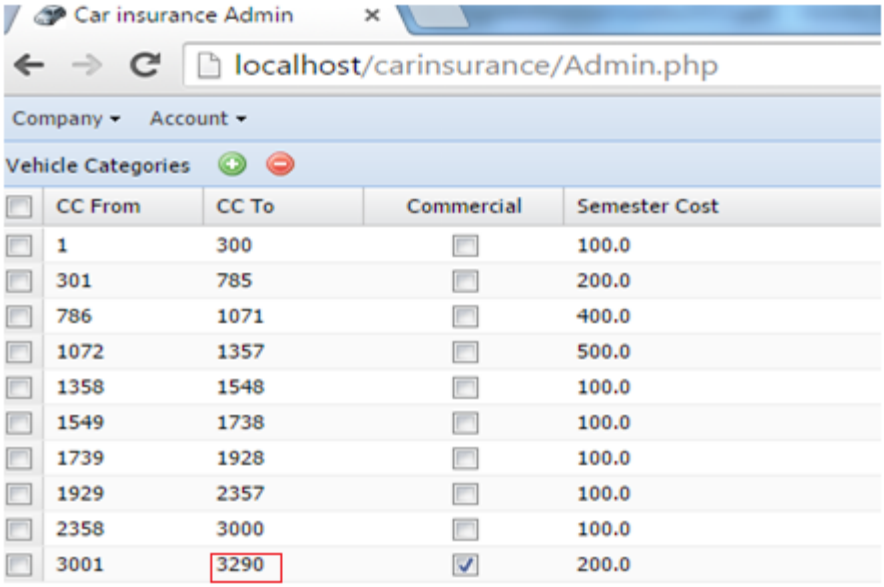
Όπως φαίνετε στο παραπάνω screenshots η νέα κατηγορία οχήματος εμφανίζεται πλέον στην λίστα με τις κατηγορίες των οχημάτων.

Τροποποίηση κατηγορίας οχημάτων

Για την τροποποίηση της κατηγορίας οχημάτων πρέπει να γίνει κλικ στην επιθυμητή κατηγορία οχήματος για τροποποίηση, ανοίγει η καρτέλα για τις αλλαγές, μόλις γίνουν πρέπει να πατηθεί το save για να καταχωρηθούν.

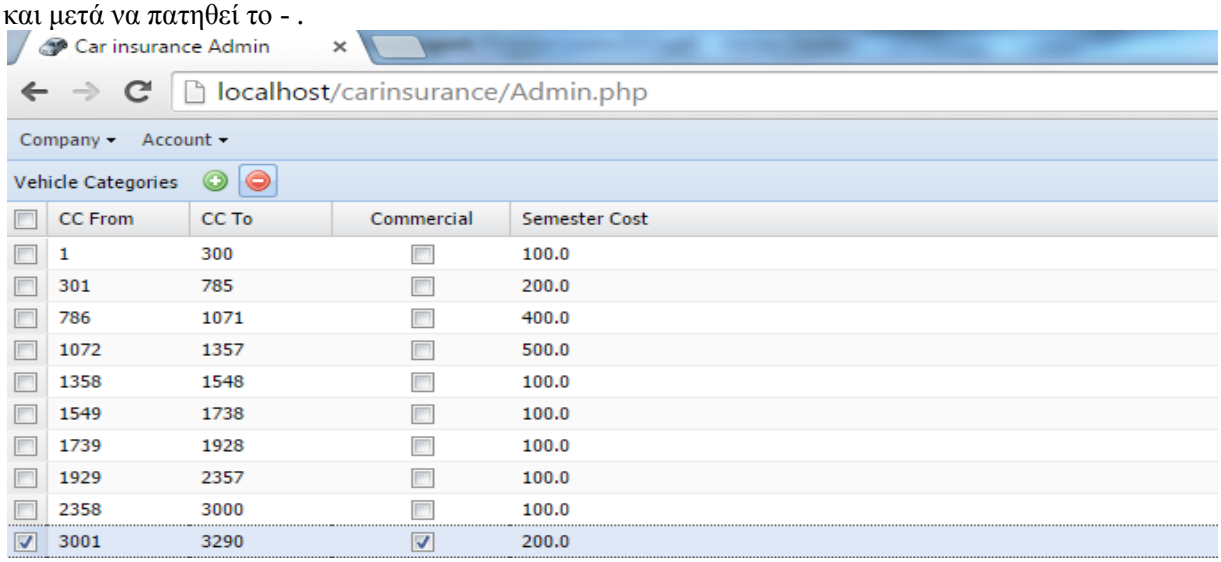


Εικόνα 29:Τροποποίηση κατηγορίας οχήματος

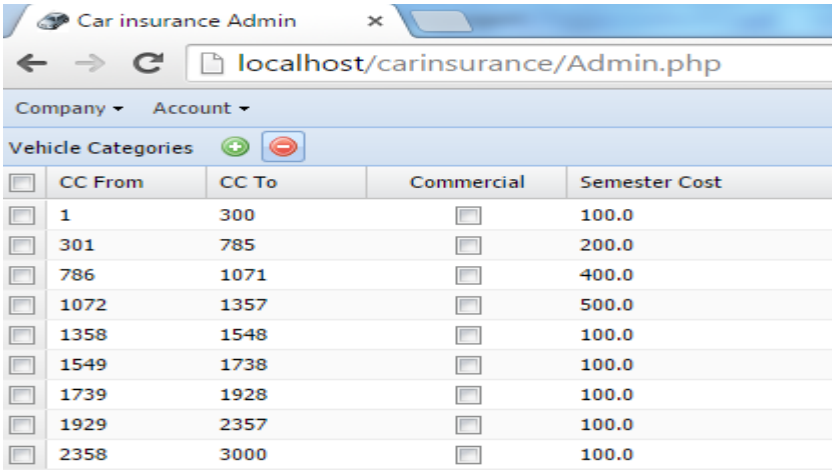


Διαγραφή κατηγορίας οχήματος

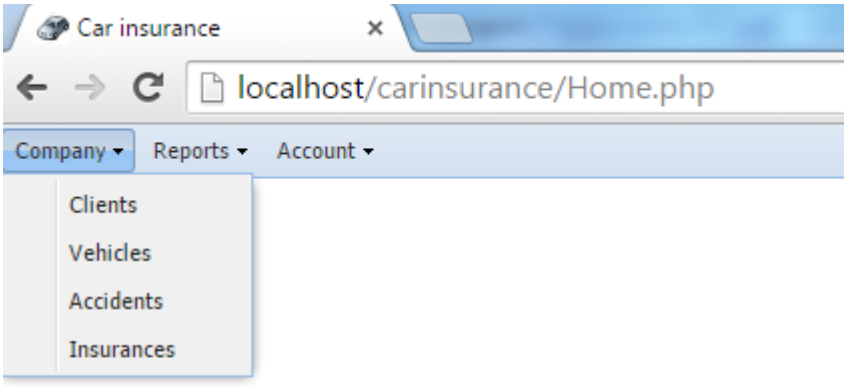
Για την διαγραφή της κατηγορίας οχήματος πρέπει να επιλεγεί η κατηγορία οχήματος προς διαγραφή και μετά να πατηθεί το - .



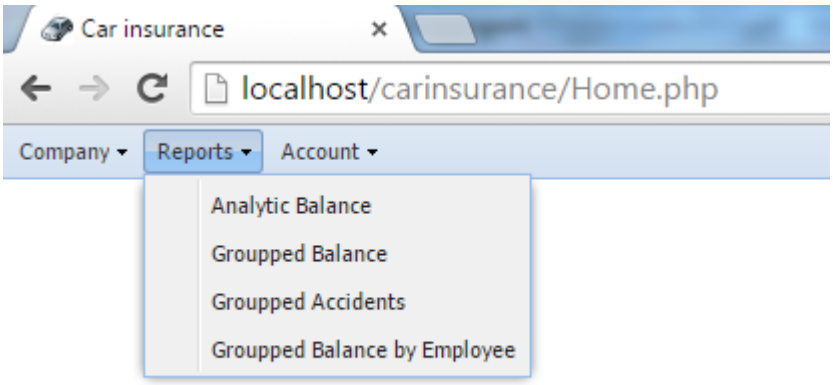
Εικόνα 30:Διαγραφή επιλεγμένης κατηγορίας οχήματος



7.5 Σελίδα Home.php



Εικόνα 31:Κατηγορίες που διαχειρίζεται η σελίδα

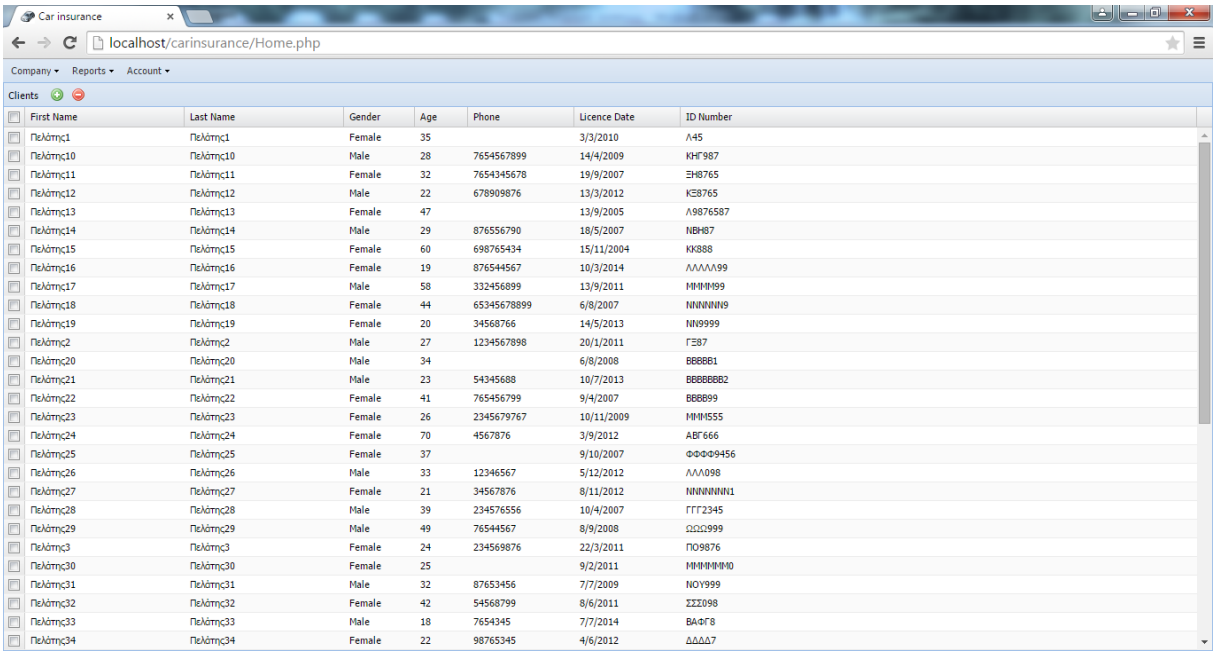


Εικόνα 32:Συγκεντρωτικές αναφορές

Εικόνα 33:Έξοδος απο την Home.php

7.5.1 Δημιουργία-Τροποποίηση-Διαγραφή Πελατών(εταιρικών και μη)

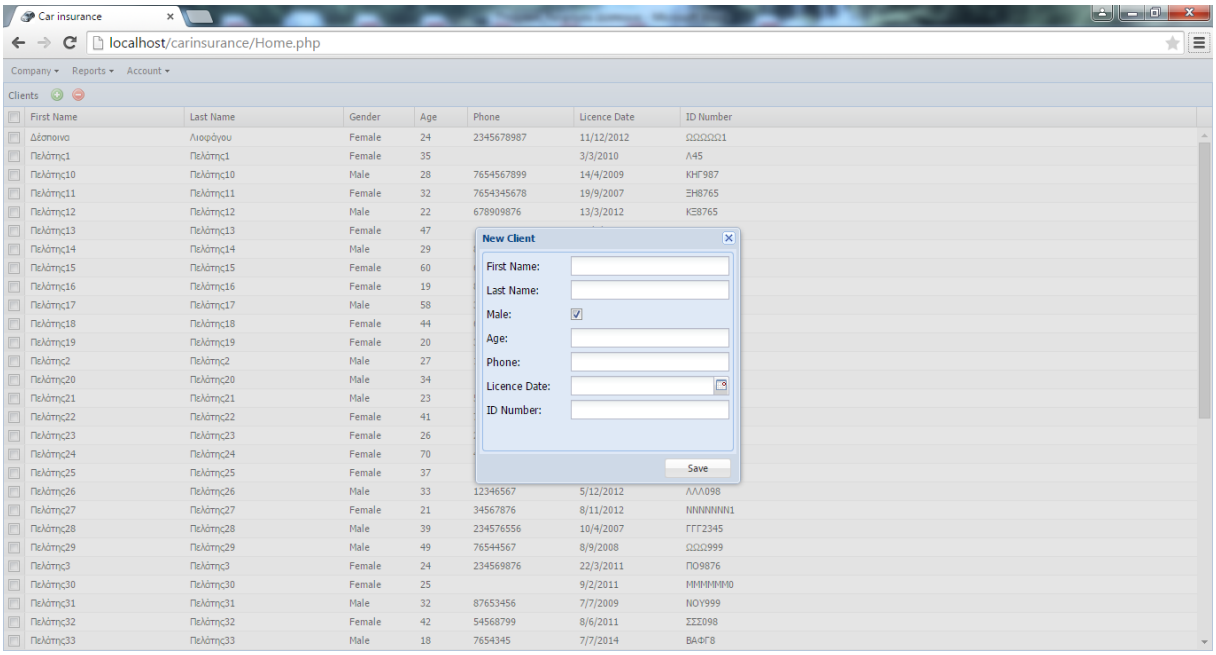
Στην αρχική οθόνη των πελατών εμφανίζεται η λίστα με τους υπάρχοντες πελάτες και το + για την δημιουργία και το – για την διαγραφή όπως φαίνετε στο παρακάτω screenshots:



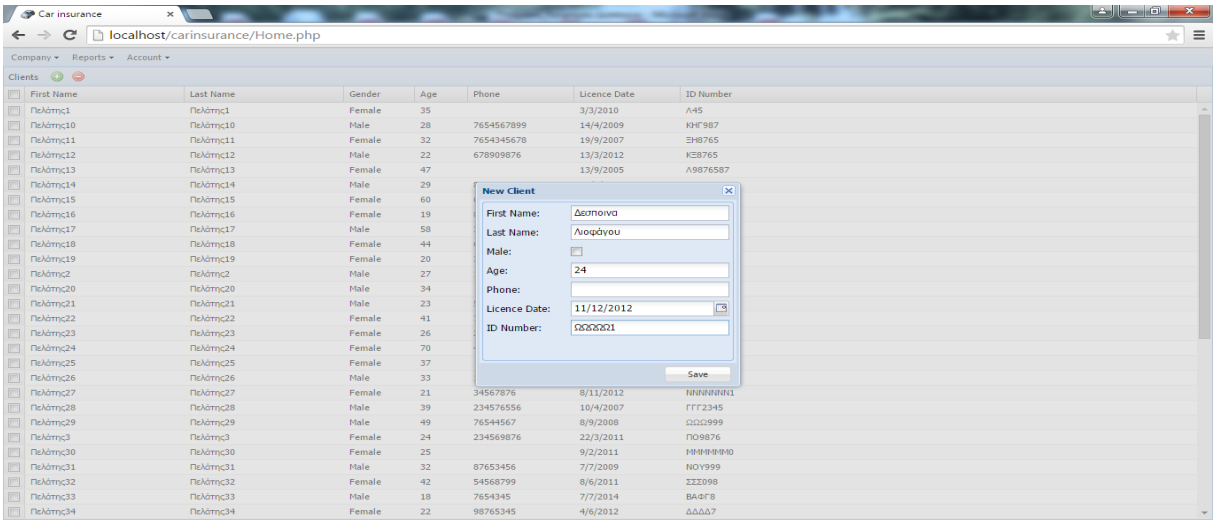
Εικόνα 34:Αρχική οθόνη πελατών

Δημιουργία πελάτη

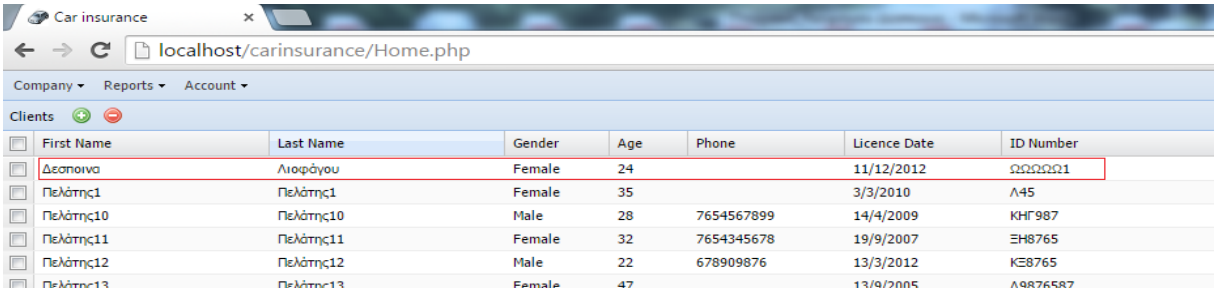
Για την δημιουργία του πελάτη πρέπει να επιλεγεί το + ,έτσι ανοίγει η φόρμα για την δημιουργία του πελάτη. Πρέπει να πληκτρολογηθούν τα στοιχεία και μετά να πατηθεί το save .



Εικόνα 35:Φόρμα δημιουργίας πελάτη



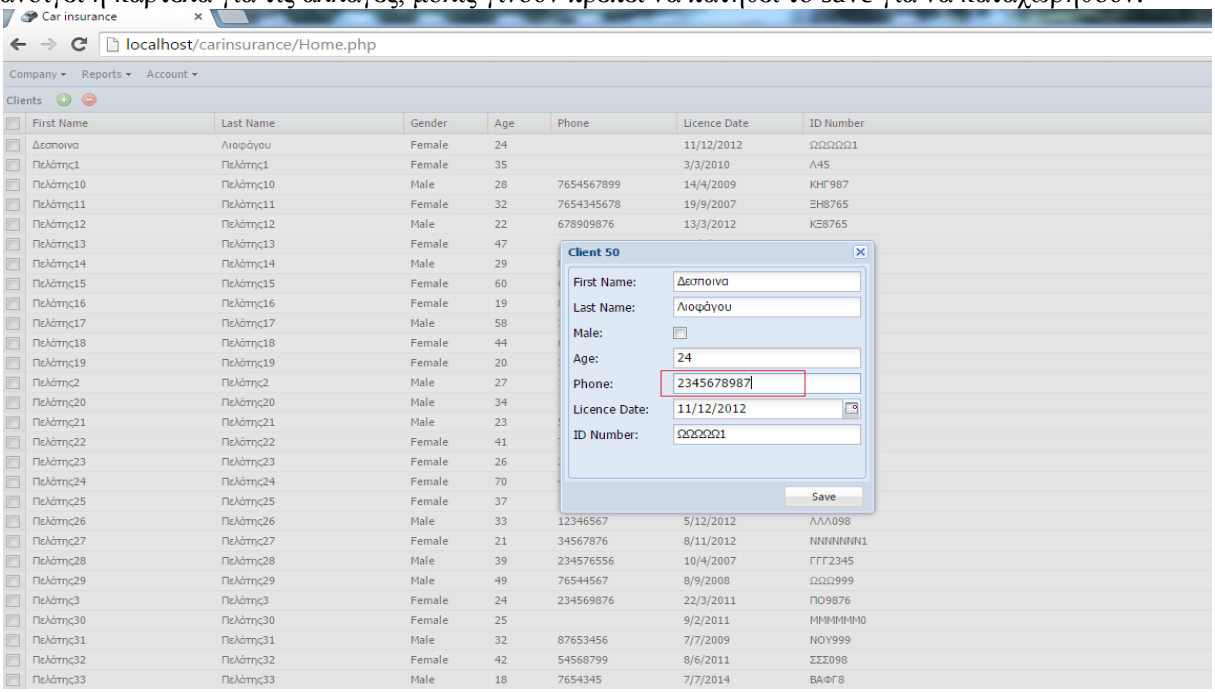
Εικόνα 36:Εισαγωγή στοιχείων στην φόρμα δημιουργίας πελάτη



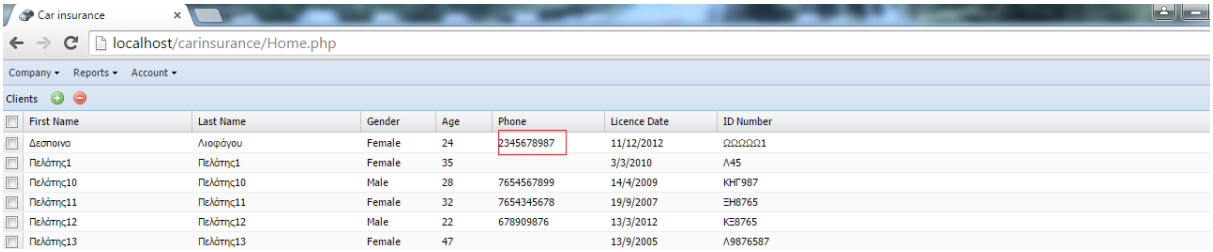
Όπως φαίνετε στο παραπάνω screenshots ο πελάτης εμφανίζεται πλέον στην λίστα με τους πελάτες.

Τροποποίηση πελάτη

Για την τροποποίηση του πελάτη πρέπει να γίνει κλικ στον επιθυμητό πελάτη για τροποποίηση, ανοίγει η καρτέλα για τις αλλαγές, μόλις γίνουν πρέπει να πατηθεί το save για να καταχωρηθούν.

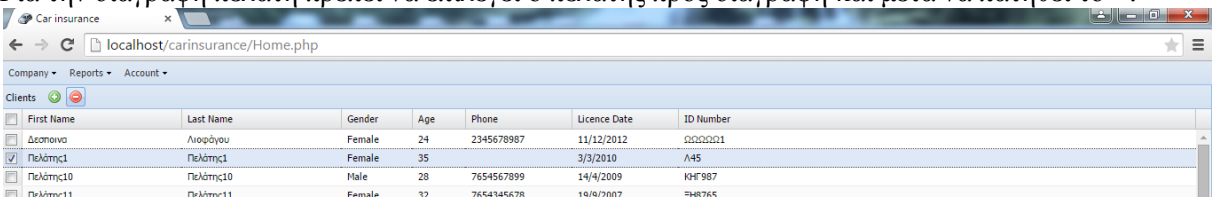


Εικόνα 37:Τροποποίηση πελάτη



Διαγραφή πελάτη

Για την διαγραφή πελάτη πρέπει να επιλεγεί ο πελάτης προς διαγραφή και μετά να πατηθεί το - .

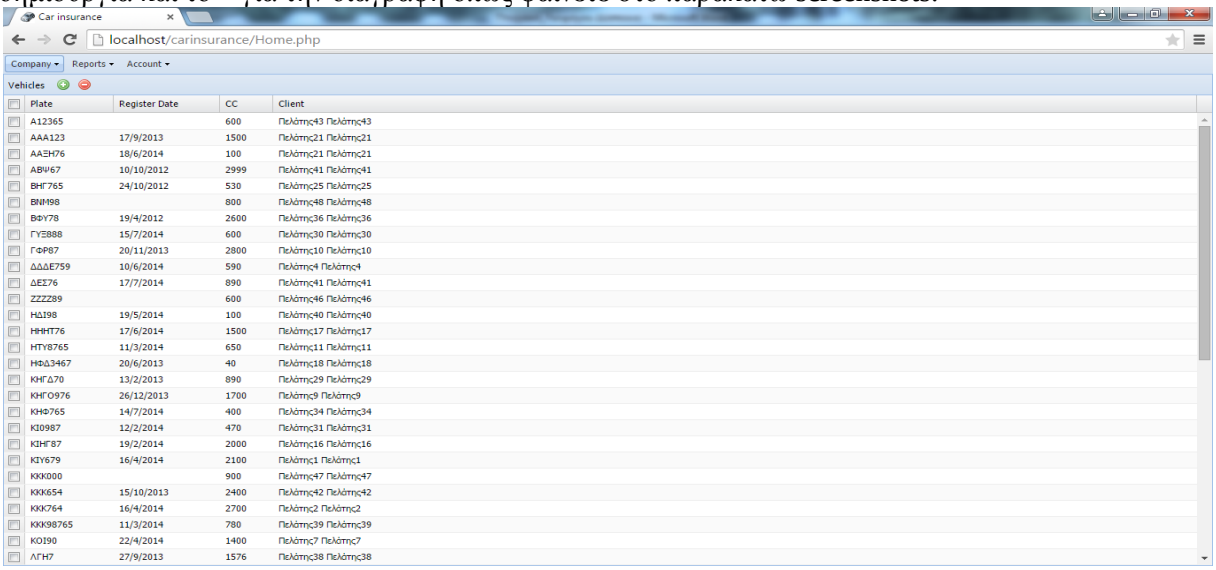


Εικόνα 38:Ανεπιτυχής διαγραφή πελάτη

Αν κάποιος πελάτης έχει κάποιο όχημα, κάποιο συμβόλαιο, ή κάποιο ατύχημα δεν γίνετε να διαγραφή. Γι αυτό τον λόγο ο ποιο πάνω πελάτης δεν μπορεί να διαγράφει γιατί πριν προσπαθήσω να τον διαγράψω τον είχα συνδέσει με όλες τις παραπάνω κατηγορίες. Πρέπει πρώτα να διαγραφεί από τις άλλες και μετά και από την κατηγορία πελάτη.

7.5.2 Δημιουργία-Τροποποίηση-Διαγραφή Οχημάτων(εταιρικών και μη)

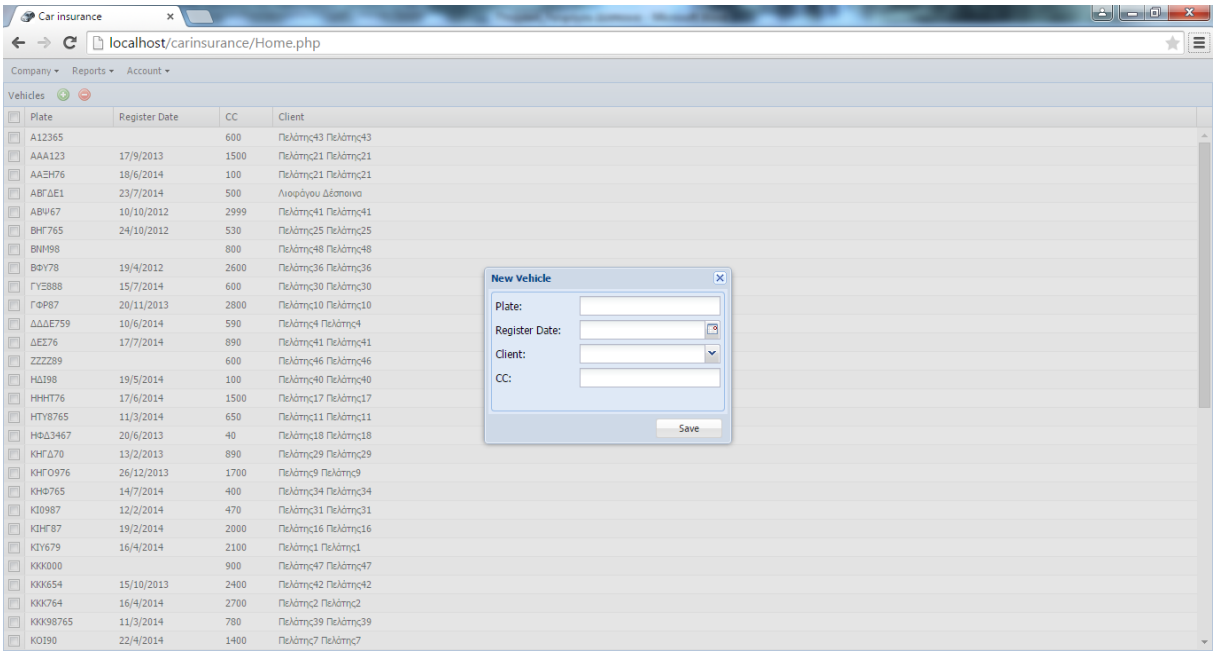
Στην αρχική οθόνη των οχημάτων εμφανίζεται η λίστα με τα υπάρχοντα οχήματα και το + για την δημιουργία και το – για την διαγραφή όπως φαίνετε στο παρακάτω screenshots:



Εικόνα 39:Αρχική οθόνη οχημάτων

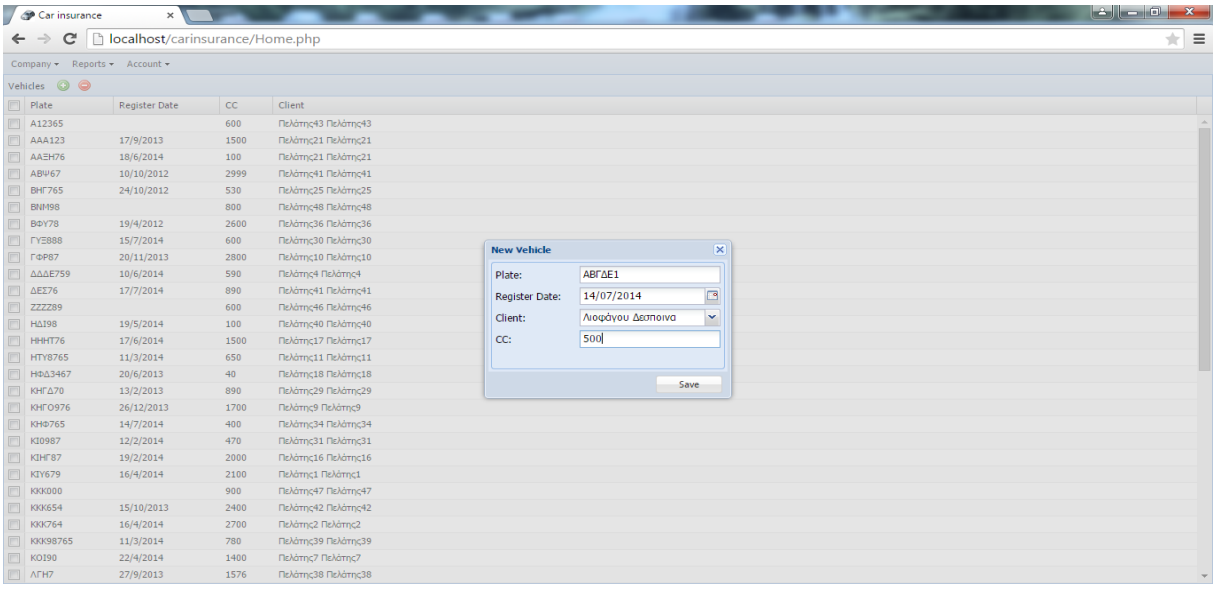
Δημιουργία οχήματος

Για την δημιουργία του οχήματος πρέπει να επιλεγεί το + ,έτσι ανοίγει η φόρμα για την δημιουργία του οχήματος. Πρέπει να πληκτρολογηθούν τα στοιχεία και μετά να πατηθεί το save .

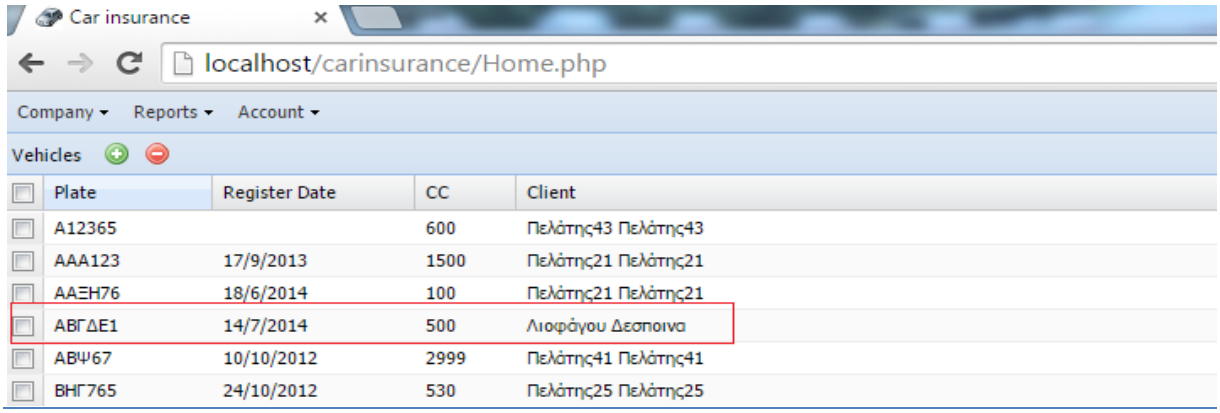


Εικόνα 40:Φόρμα δημιουργίας οχήματος

Στην παραπάνω φόρμα δίνεται η δυνατότητα επιλογής πελάτη στον οποίο θα ανήκει το όχημα από αυτούς που έχουν είδη δημιουργηθεί . Αν δεν καταχωρηθεί register date τότε το όχημα θα θεωρηθεί μη εταιρικό.



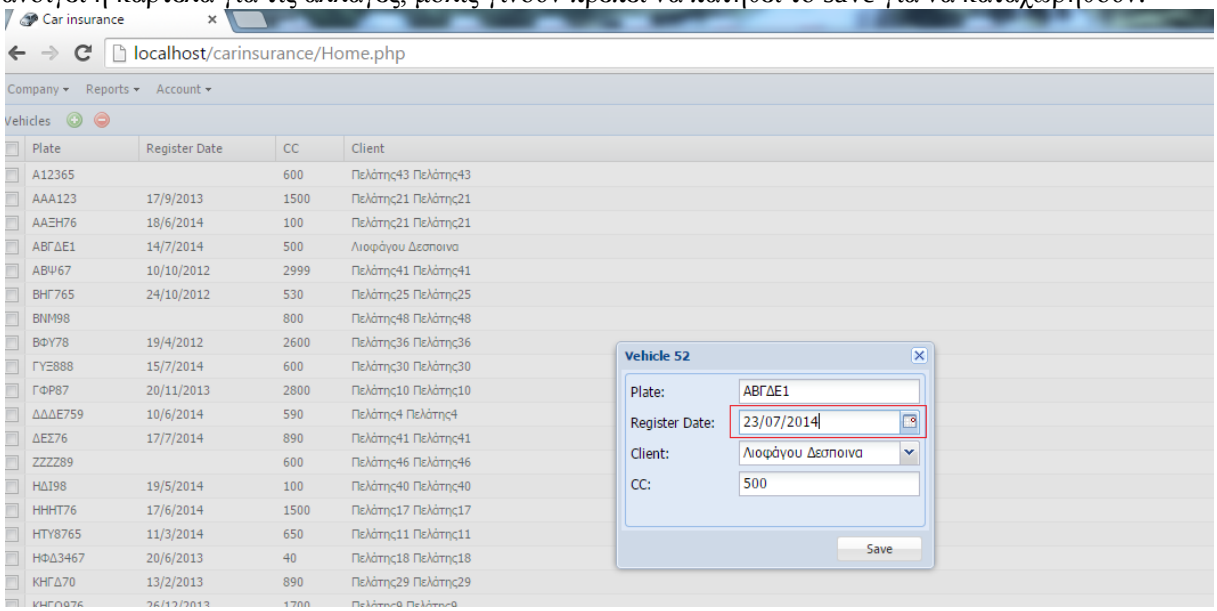
Εικόνα 41:Εισαγωγή στοιχείων στην φόρμα δημιουργίας οχήματος



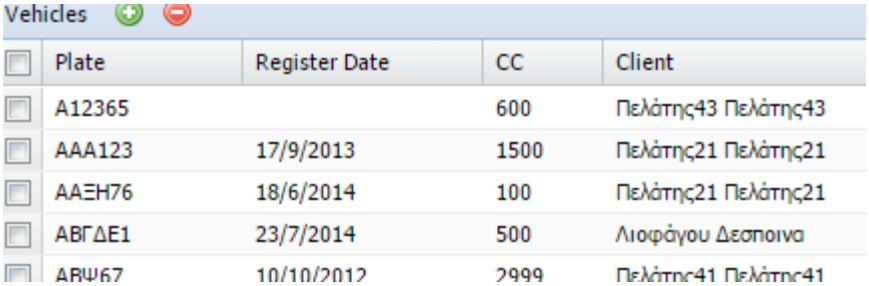
Όπως φαίνετε στο παραπάνω screenshots το όχημα εμφανίζεται πλέον στην λίστα με τα οχήματα.

Τροποποίηση κατηγορίας οχημάτων

Για την τροποποίηση του οχήματος πρέπει να γίνει κλικ στο επιθυμητό όχημα για τροποποίηση, ανοίγει η καρτέλα για τις αλλαγές, μόλις γίνουν πρέπει να πατηθεί το save για να καταχωρηθούν.

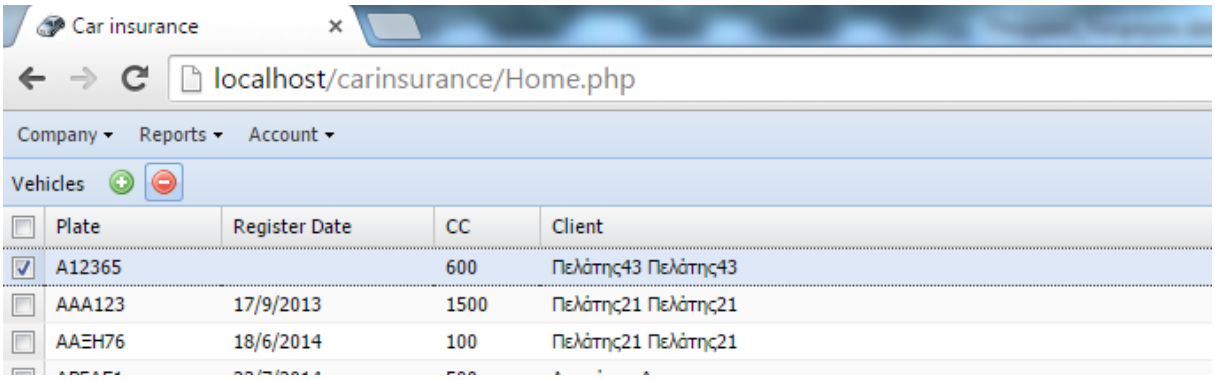


Εικόνα 42:Τροποποίηση οχήματος



Διαγραφή οχήματος

Για την διαγραφή οχήματος πρέπει να επιλεγεί το όχημα προς διαγραφή και μετά να πατηθεί το - .

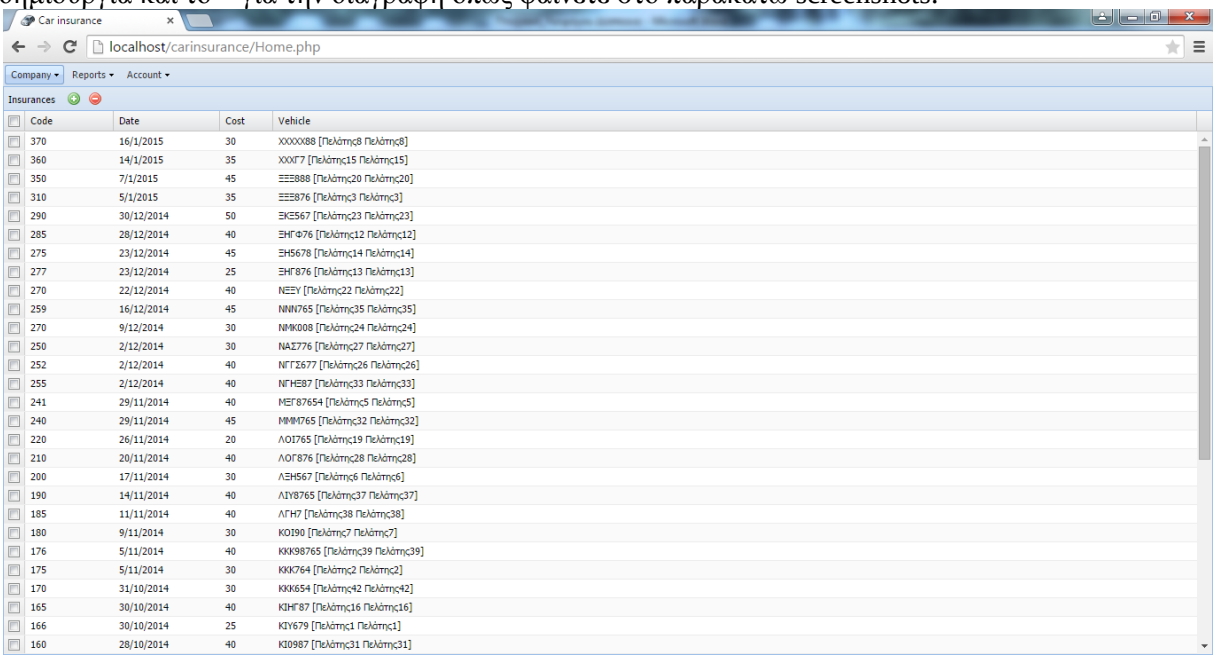


Εικόνα 43:Ανεπιτυχής διαγραφή οχήματος

Αν κάποιο όχημα συνδέεται με κάποιο συμβόλαιο, ή κάποιο ατύχημα δεν γίνετε να διαγραφή. Γι αυτό τον λόγο ο ποιο πάνω όχημα δεν μπορεί να διαγράφει γιατί πριν προσπαθήσω να τον διαγράψω το είχα συνδέσει με ατύχημα. Πρέπει πρώτα να διαγραφεί από το ατύχημα και μετά και από τα οχήματα.

7.5.3 Δημιουργία-Τροποποίηση-Διαγραφή Συμβολαίων

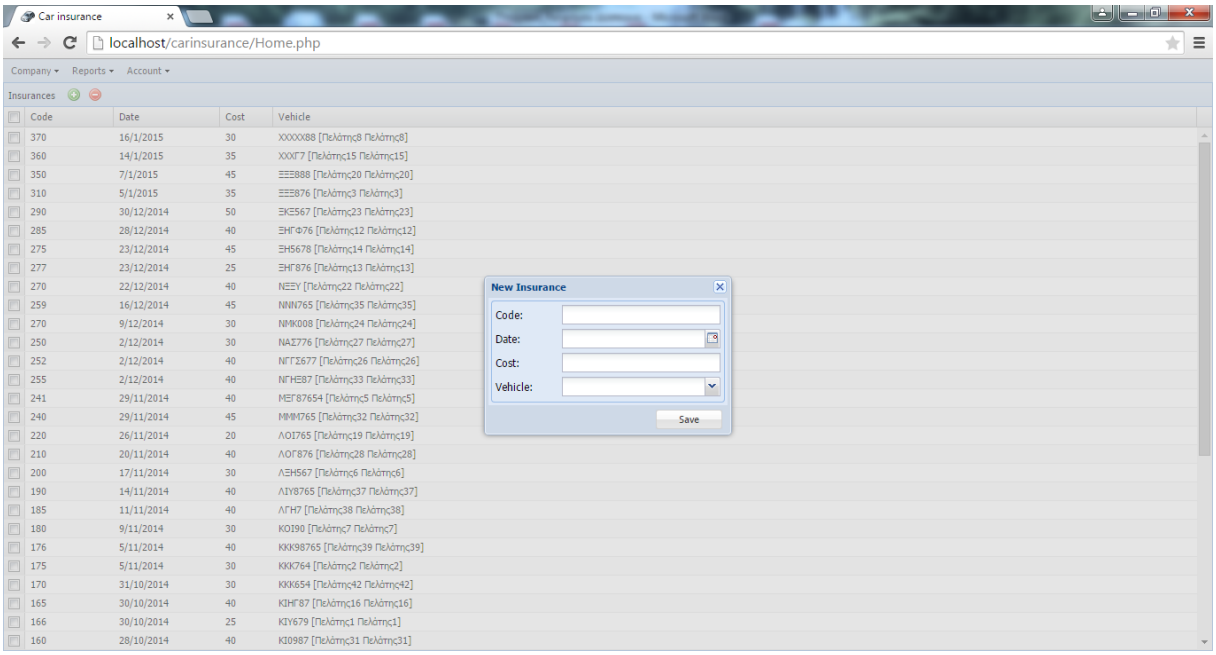
Στην αρχική οθόνη των συμβολαίων εμφανίζεται η λίστα με τα υπάρχοντα συμβόλαια και το + για την δημιουργία και το – για την διαγραφή όπως φαίνετε στο παρακάτω screenshots:



Εικόνα 44:Αρχική οθόνη Συμβολαίων

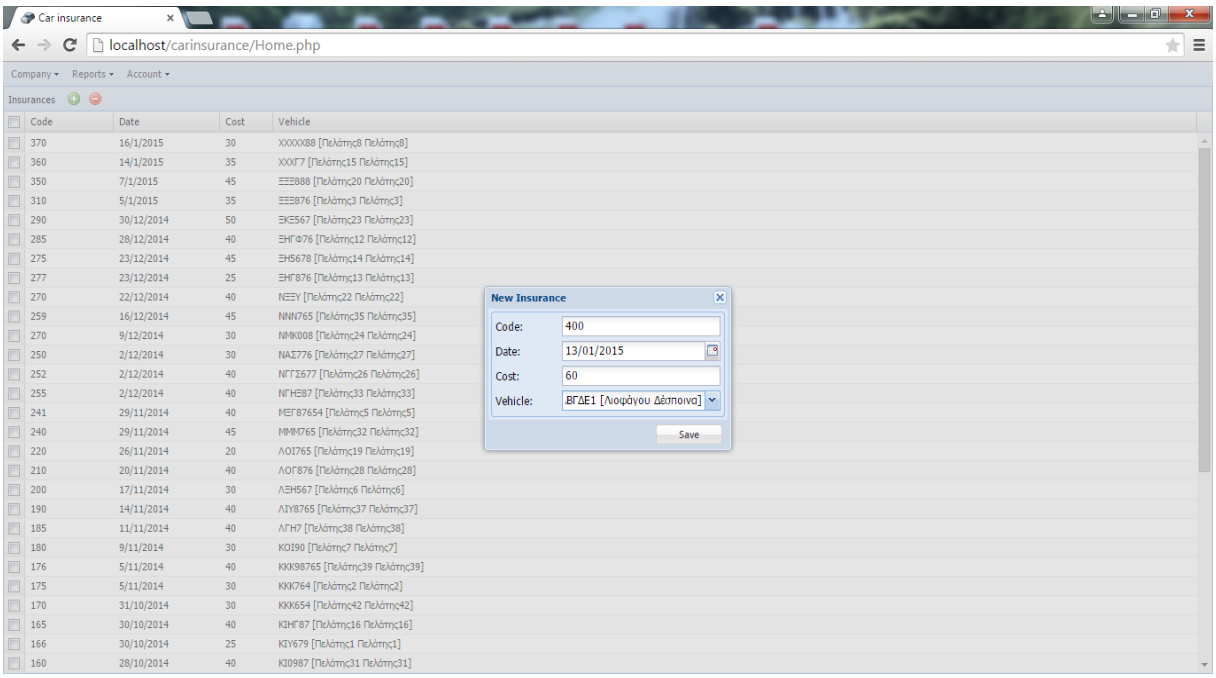
<u>Δημιουργία συμβολαίου</u>

Για την δημιουργία του συμβολαίου πρέπει να επιλεγεί το + ,έτσι ανοίγει η φόρμα για την δημιουργία του συμβολαίου. Πρέπει να πληκτρολογηθούν τα στοιχεία και μετά να πατηθεί το save.

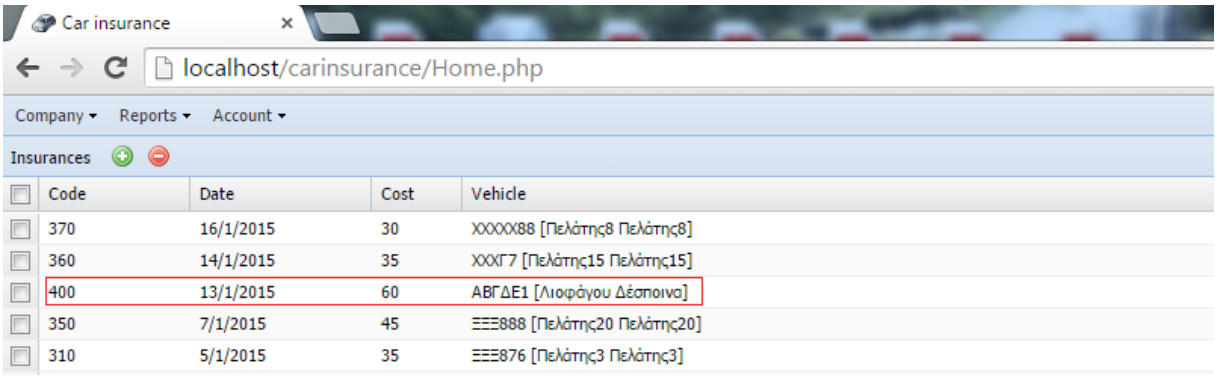


Εικόνα 45:Φόρμα δημιουργίας συμβολαίου

Στην παραπάνω φόρμα δίνεται η δυνατότητα επιλογής πελάτη στον οποίο θα ανήκει το όχημα από αυτούς που έχουν είδη δημιουργηθεί.

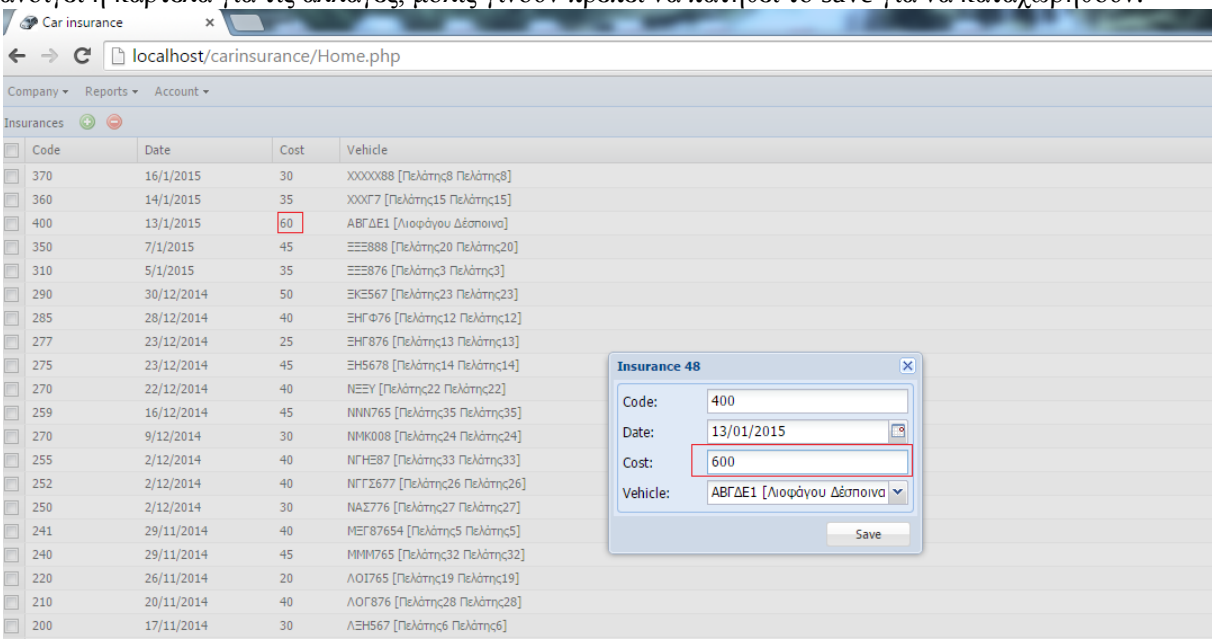


Εικόνα 46:Εισαγωγή στοιχείων στην φόρμα ασφαλιστηρίου

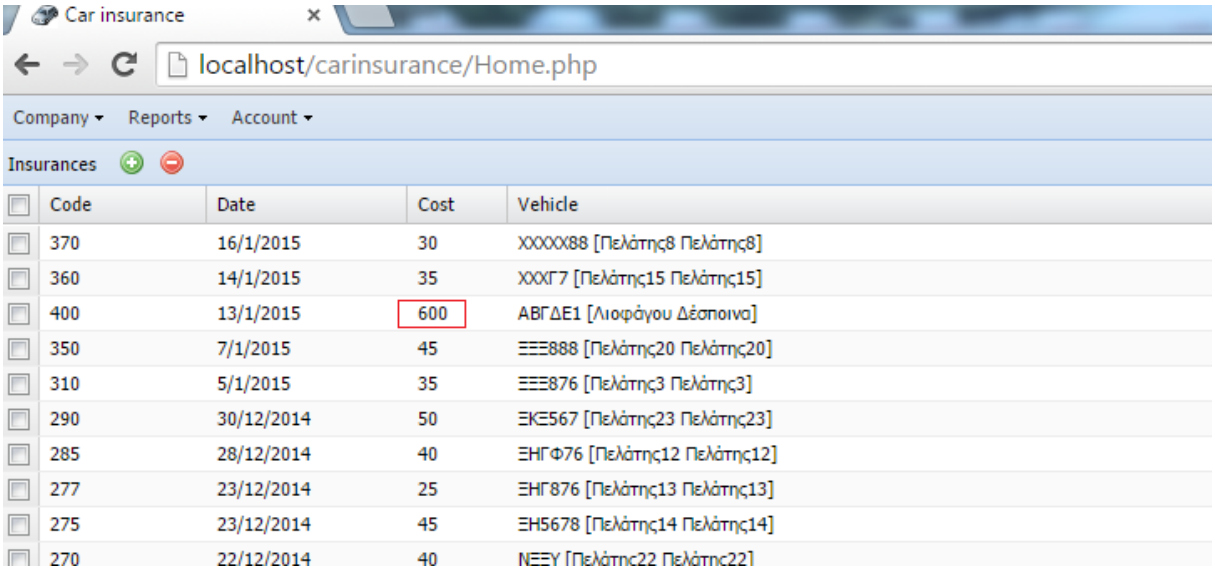


Τροποποίηση συμβολαίου

Για την τροποποίηση του ασφαλιστηρίου πρέπει να γίνει κλικ στο επιθυμητό όχημα για τροποποίηση, ανοίγει η καρτέλα για τις αλλαγές, μόλις γίνουν πρέπει να πατηθεί το save για να καταχωρηθούν.

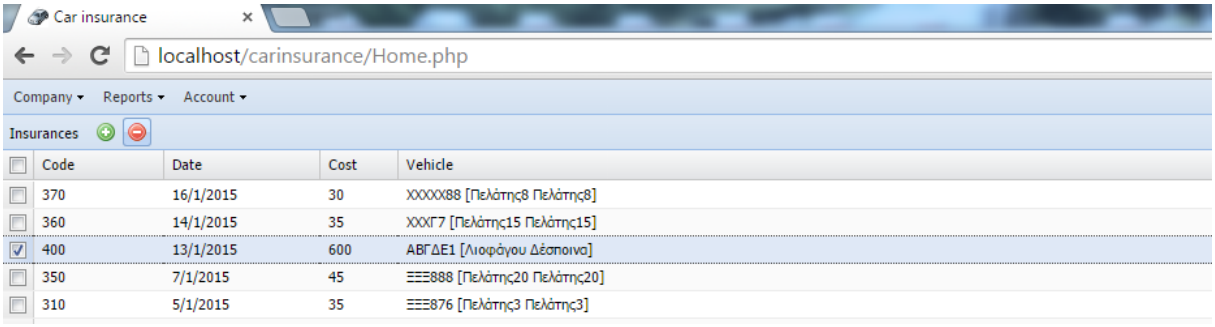


Εικόνα 47:Τροποποίηση συμβολαίου

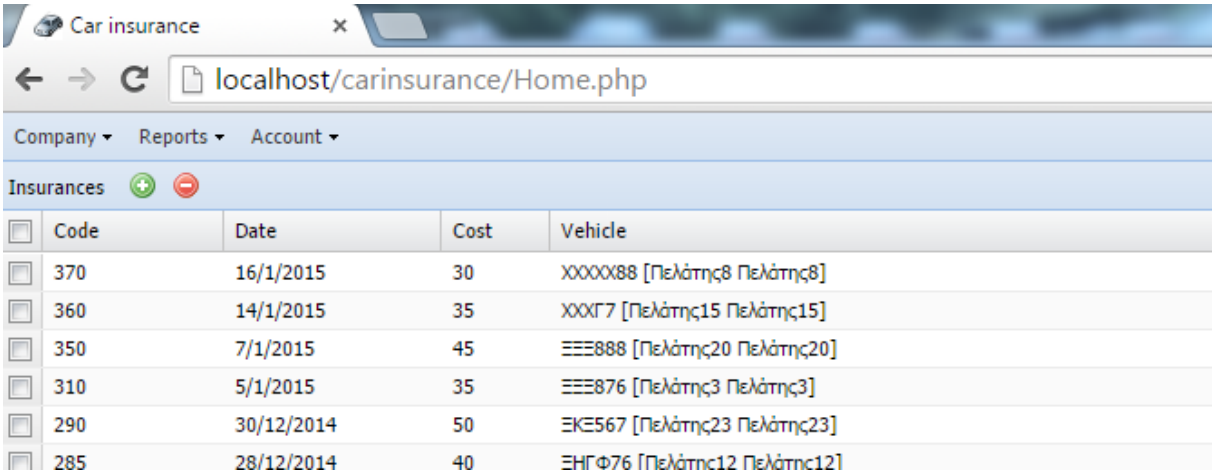


Διαγραφή συμβολαίου

Για την διαγραφή συμβολαίου πρέπει να επιλεγεί το συμβόλαιο προς διαγραφή και μετά να πατηθεί το - .

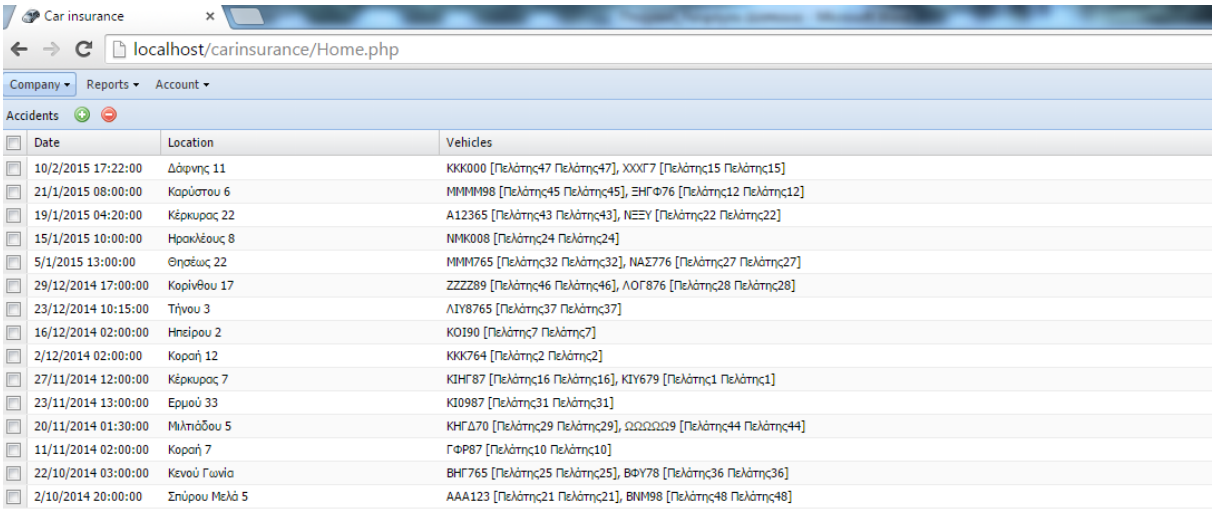


Εικόνα 48:Διαγραφή συμβολαίου



7.5.4 Δημιουργία-Τροποποίηση-Διαγραφή Ατυχημάτων

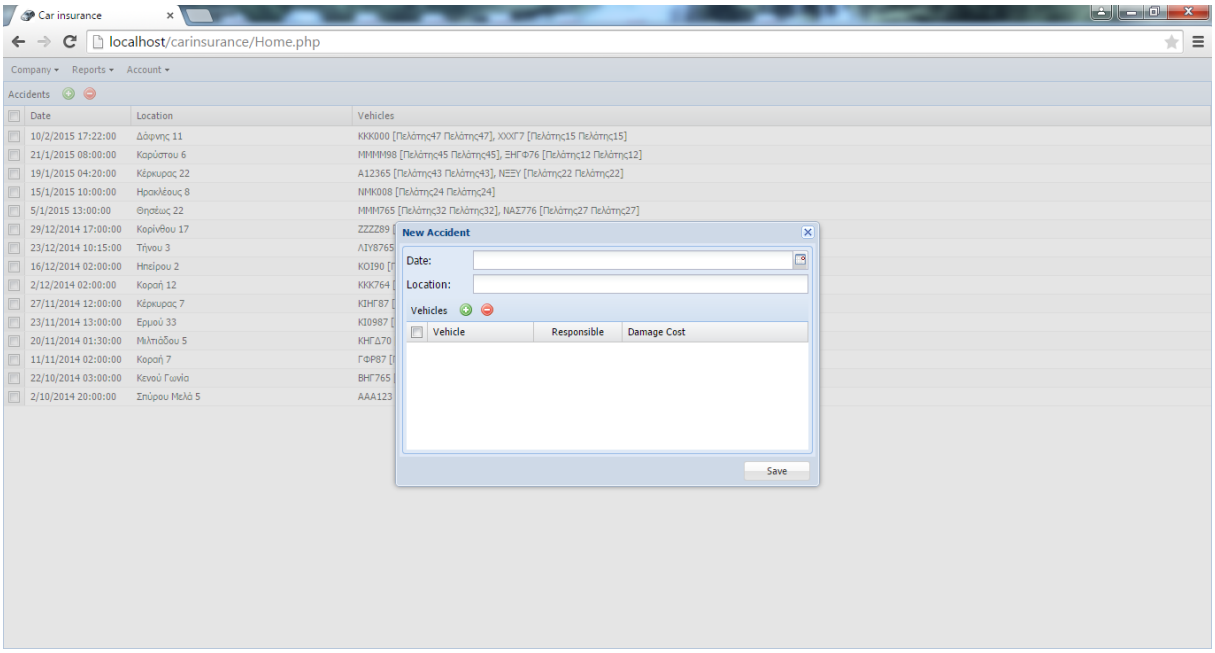
Στην αρχική οθόνη των ατυχημάτων εμφανίζεται η λίστα με τα υπάρχοντα ατυχήματα και το + για την δημιουργία και το – για την διαγραφή όπως φαίνετε στο παρακάτω screenshots:



Εικόνα 49:Αρχική οθόνη ατυχημάτων

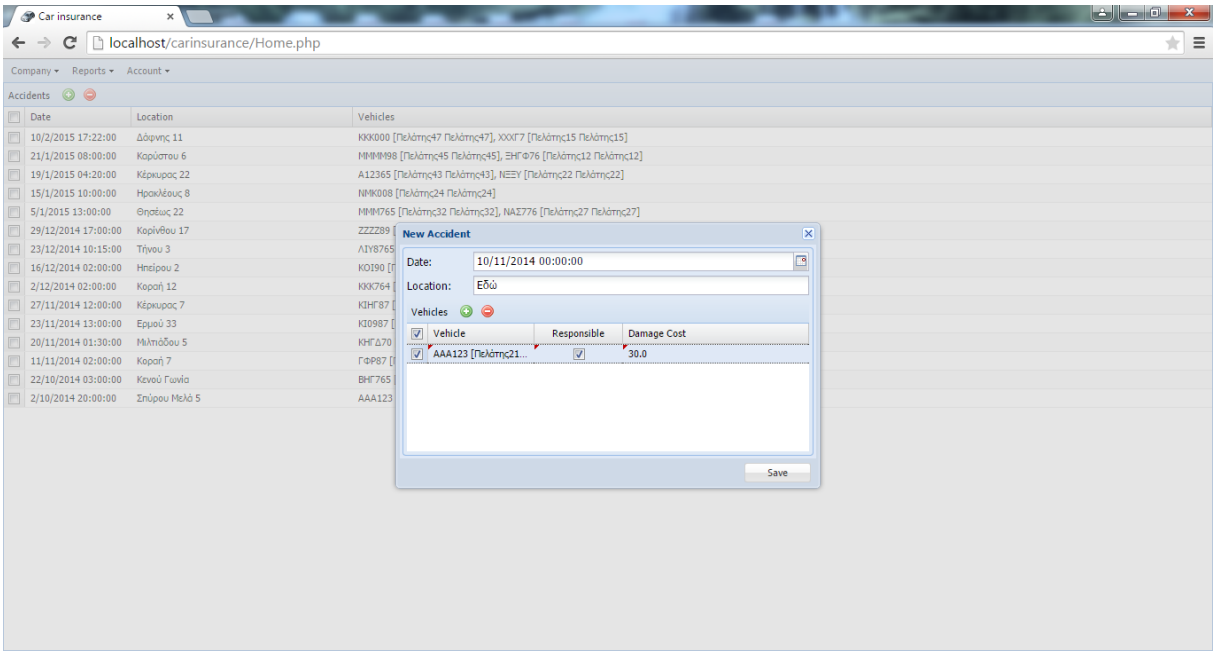
Δημιουργία ατυχήματος

Για την δημιουργία του ατυχήματος πρέπει να επιλεγεί το + ,έτσι ανοίγει η φόρμα για την δημιουργία του ατυχήματος. Πρέπει να πληκτρολογηθούν τα στοιχεία και μετά να πατηθεί το save.

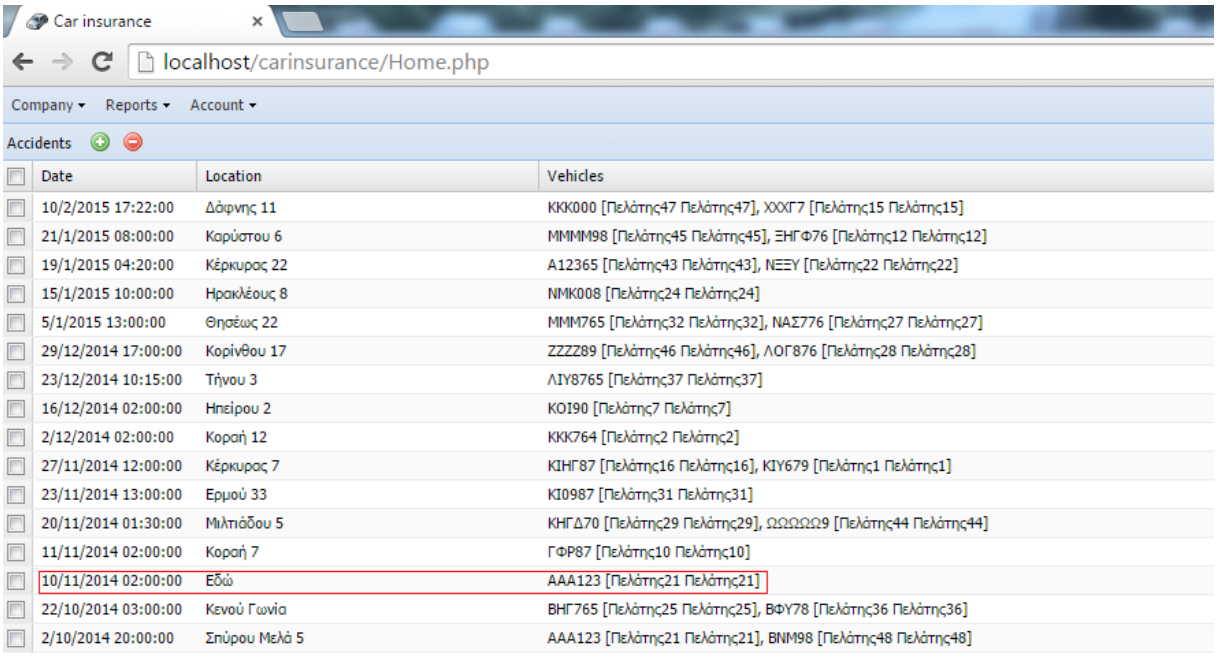


Εικόνα 50:Φόρμα δημιουργίας ατυχήματος

Στην παραπάνω φόρμα δίνεται η δυνατότητα πατώντας το + να γίνει εισαγωγή οχήματος(εταιρικού ή μη) από αυτά που έχουν είδη δημιουργηθεί στο ατύχημα ή πατώντας το – να γίνει διαγραφή οχήματος(εταιρικού ή μη) από ένα ατύχημα.



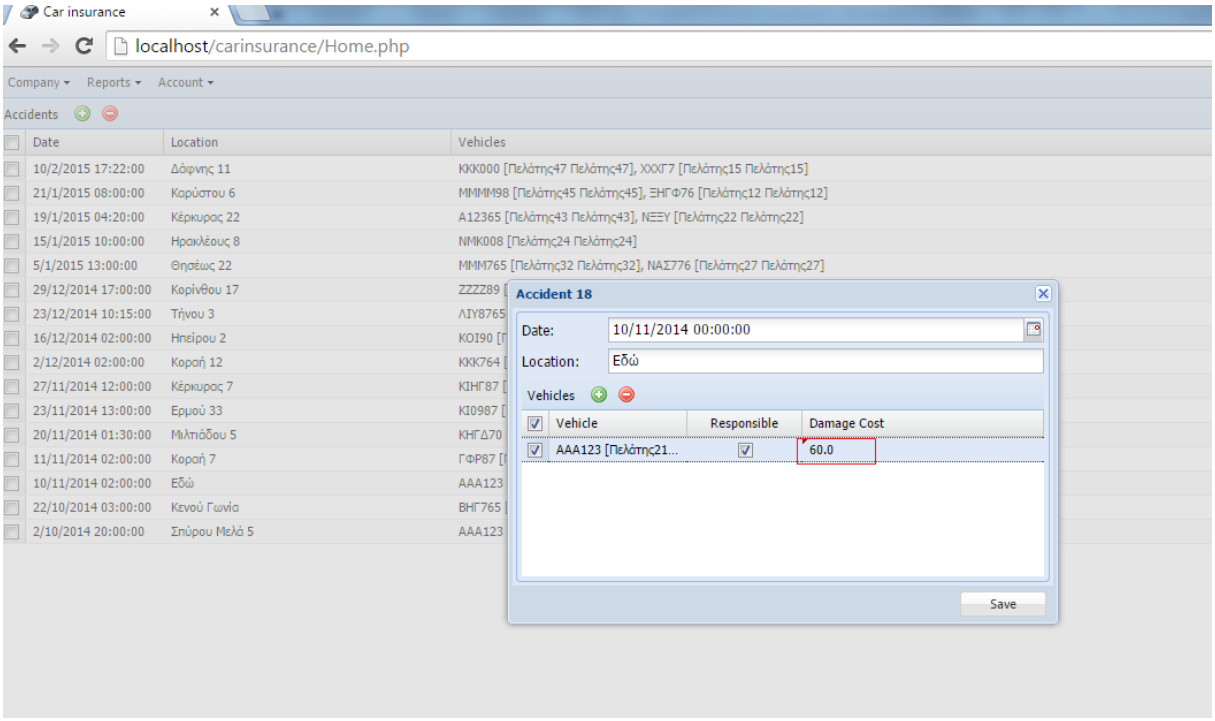
Εικόνα 51:Εισαγωγή στοιχείων στην φόρμα δημιουργίας ατυχήματος



Date	Location	Vehicles
10/2/2015 17:22:00	Δάφνης 11	ΚΚΚ000 [Πελάτης47 Πελάτης47], ΧΧΧΓ7 [Πελάτης15 Πελάτης15]
21/1/2015 08:00:00	Καρύστου 6	ΜΜΜΜ98 [Πελάτης45 Πελάτης45], ΞΗΓΦ76 [Πελάτης12 Πελάτης12]
19/1/2015 04:20:00	Κέρκυρας 22	Α12365 [Πελάτης43 Πελάτης43], ΝΞΞΥ [Πελάτης22 Πελάτης22]
15/1/2015 10:00:00	Ηρακλέους 8	ΝΜΚ008 [Πελάτης24 Πελάτης24]
5/1/2015 13:00:00	Θησέως 22	ΜΜΜ765 [Πελάτης32 Πελάτης32], ΝΑΣ776 [Πελάτης27 Πελάτης27]
29/12/2014 17:00:00	Κορίνθου 17	ΖΖΖΖ89 [Πελάτης46 Πελάτης46], ΛΟΓ876 [Πελάτης28 Πελάτης28]
23/12/2014 10:15:00	Τήνου 3	ΛΙΥ8765 [Πελάτης37 Πελάτης37]
16/12/2014 02:00:00	Ηπείρου 2	ΚΟΙ90 [Πελάτης7 Πελάτης7]
2/12/2014 02:00:00	Κοραή 12	ΚΚΚ764 [Πελάτης2 Πελάτης2]
27/11/2014 12:00:00	Κέρκυρας 7	ΚΙΗΓ87 [Πελάτης16 Πελάτης16], ΚΙΥ679 [Πελάτης1 Πελάτης1]
23/11/2014 13:00:00	Ερμού 33	ΚΙ0987 [Πελάτης31 Πελάτης31]
20/11/2014 01:30:00	Μιλτιάδου 5	ΚΗΓΔ70 [Πελάτης29 Πελάτης29], ΩΩΩΩΩ9 [Πελάτης44 Πελάτης44]
11/11/2014 02:00:00	Κοραή 7	ΓΦΡ87 [Πελάτης10 Πελάτης10]
10/11/2014 02:00:00	Εδώ	ΑΑΑ123 [Πελάτης21 Πελάτης21]
22/10/2014 03:00:00	Κενού Γωνία	ΒΗΓ765 [Πελάτης25 Πελάτης25], ΒΦΥ78 [Πελάτης36 Πελάτης36]
2/10/2014 20:00:00	Σπύρου Μελά 5	ΑΑΑ123 [Πελάτης21 Πελάτης21], ΒΝΜ98 [Πελάτης48 Πελάτης48]

<u>Τροποποίηση ατυχήματος</u>

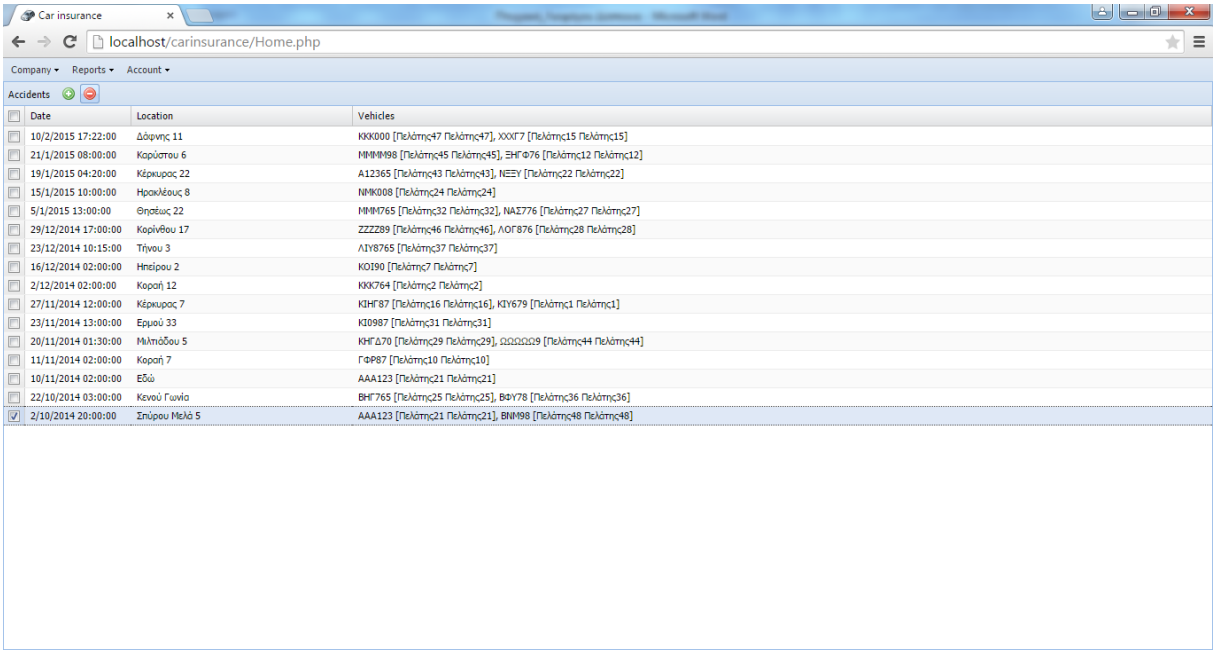
Για την τροποποίηση του ατυχήματος πρέπει να γίνει κλικ στο επιθυμητό ατύχημα για τροποποίηση, ανοίγει η καρτέλα για τις αλλαγές, μόλις γίνουν πρέπει να πατηθεί το save για να καταχωρηθούν.



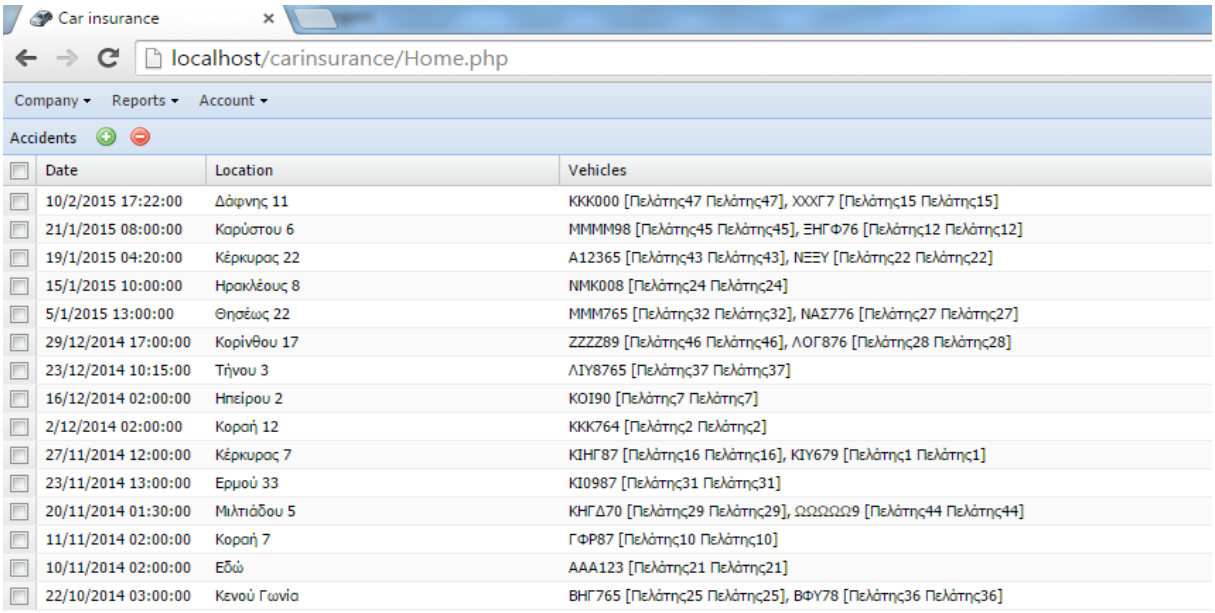
Εικόνα 52:Τροποποίηση ατυχήματος

Διαγραφή ατυχήματος

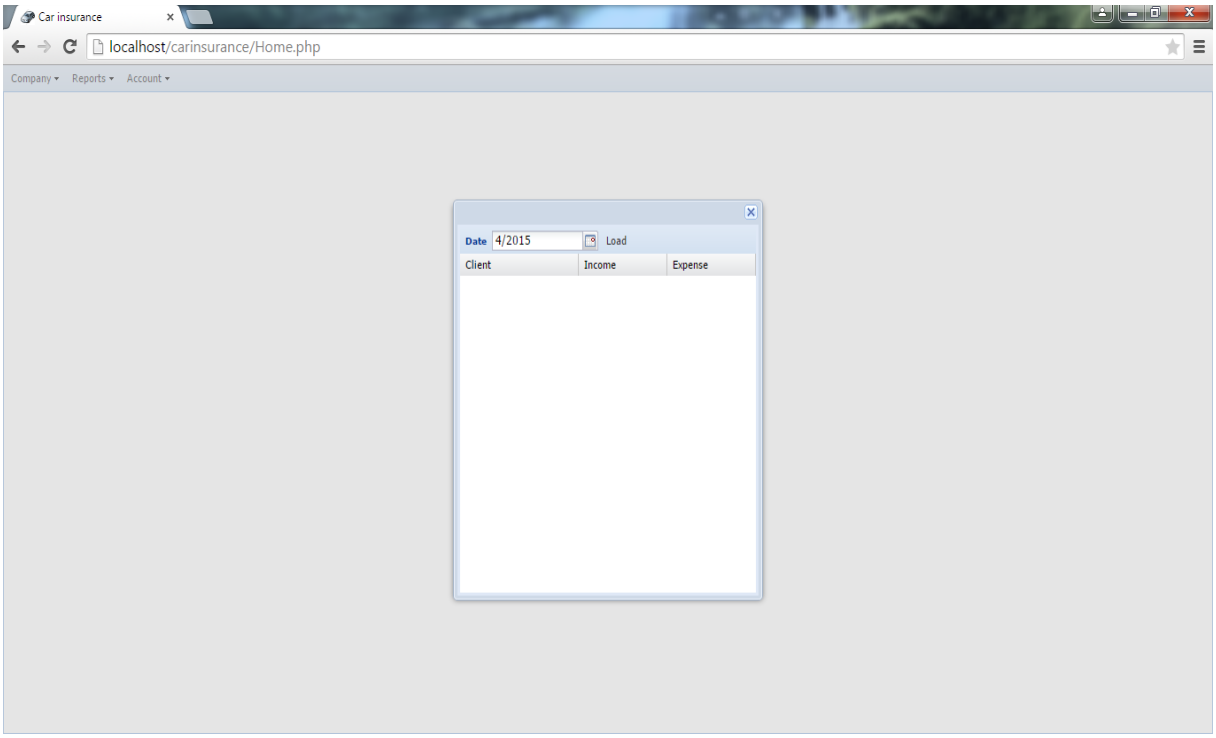
Για την διαγραφή ατυχήματος πρέπει να επιλεγεί το ατύχημα προς διαγραφή και μετά να πατηθεί το -
.



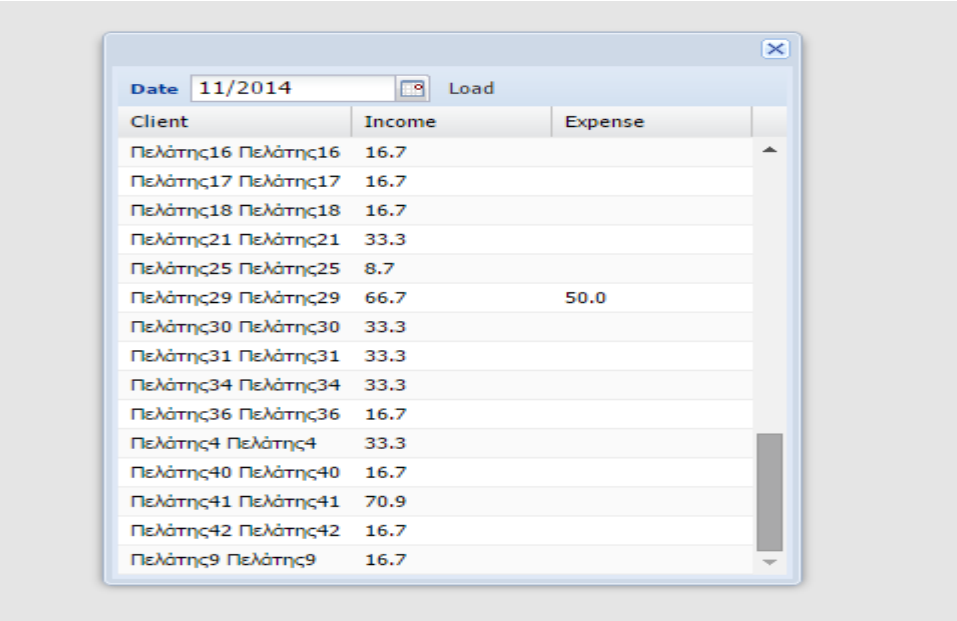
Εικόνα 53:Διαγραφή ατυχήματος



7.5.5 Συγκεντρωτική αναφορά για τα έσοδα και έξοδα κάθε μήνα ανά πελάτη(Analytic Balance)



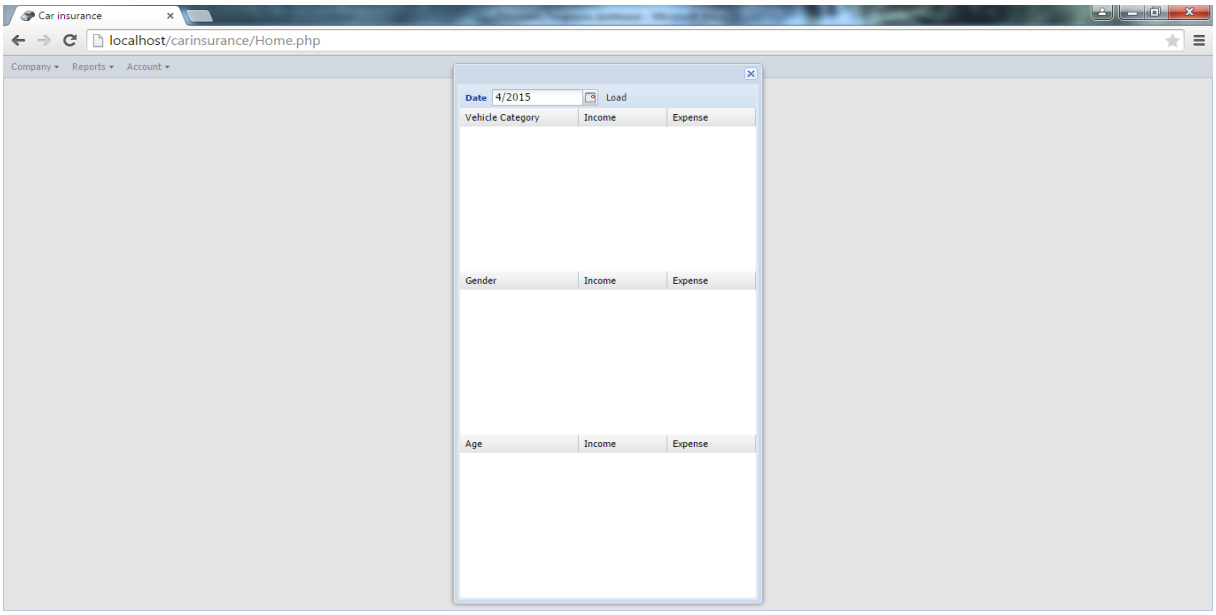
Εικόνα 54:Analytic Balance



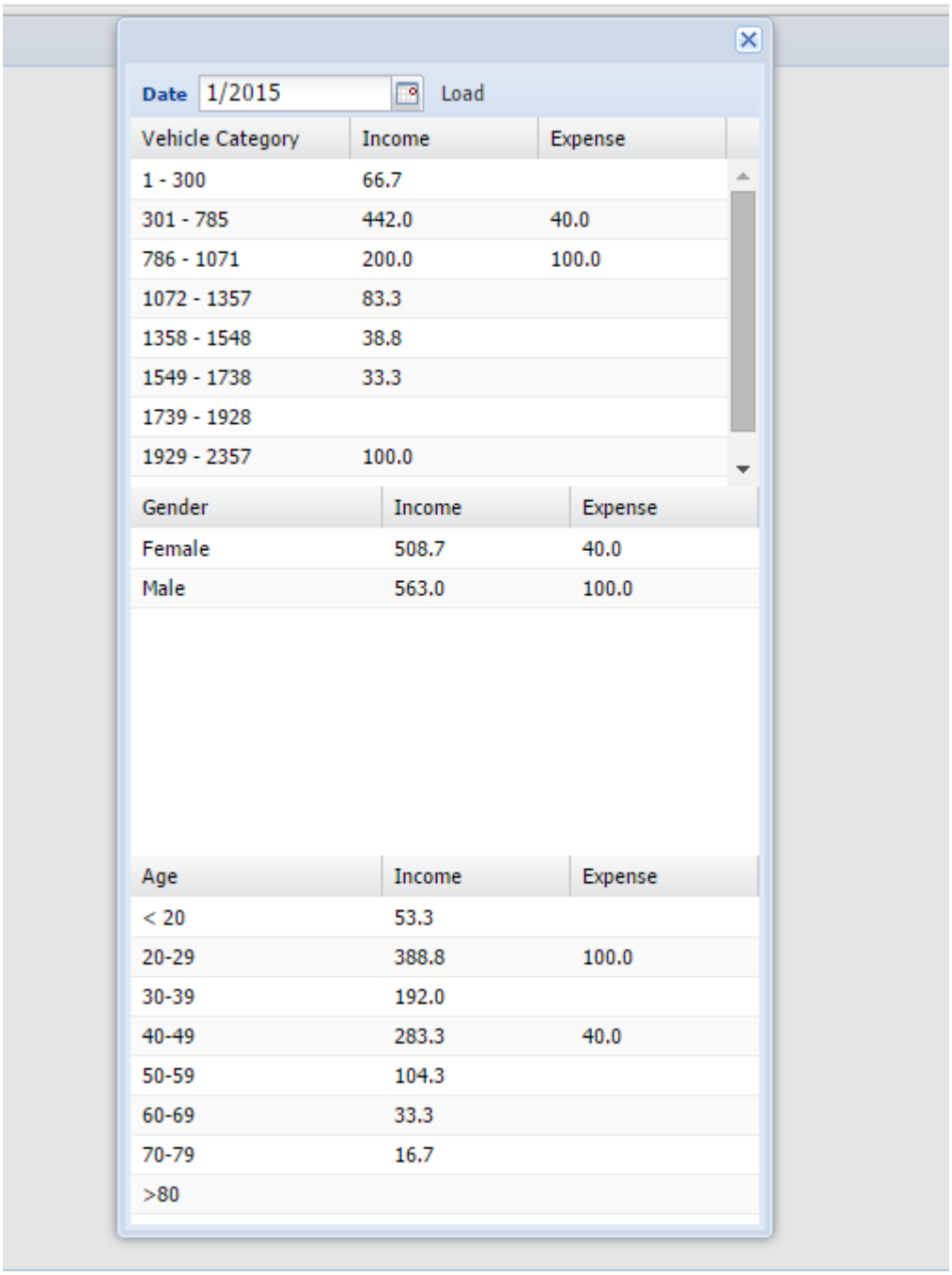
Εικόνα 55:Έσοδα - έξοδα ανά πελάτη 11/2014

Στον παραπάνω πίνακα φαίνονται τα έσοδα και τα έξοδα ανά πελάτη τον 11/2014.
Ο χρήστης μπορεί να επιλέξει όποιο μήνα θέλει και να πατήσει load για να δει τα έσοδα και τα έξοδα ανά πελάτη τον συγκεκριμένο μήνα που θέλει.

7.5.6 Συγκεντρωτική αναφορά για τα έσοδα και έξοδα κάθε μήνα ανά κατηγορία οχήματος, ανά φύλο και ανά ηλικία(Groupped Balance).



Εικόνα 56:Groupped Balance



Vehicle Category	Income	Expense
1 - 300	66.7	
301 - 785	442.0	40.0
786 - 1071	200.0	100.0
1072 - 1357	83.3	
1358 - 1548	38.8	
1549 - 1738	33.3	
1739 - 1928		
1929 - 2357	100.0	

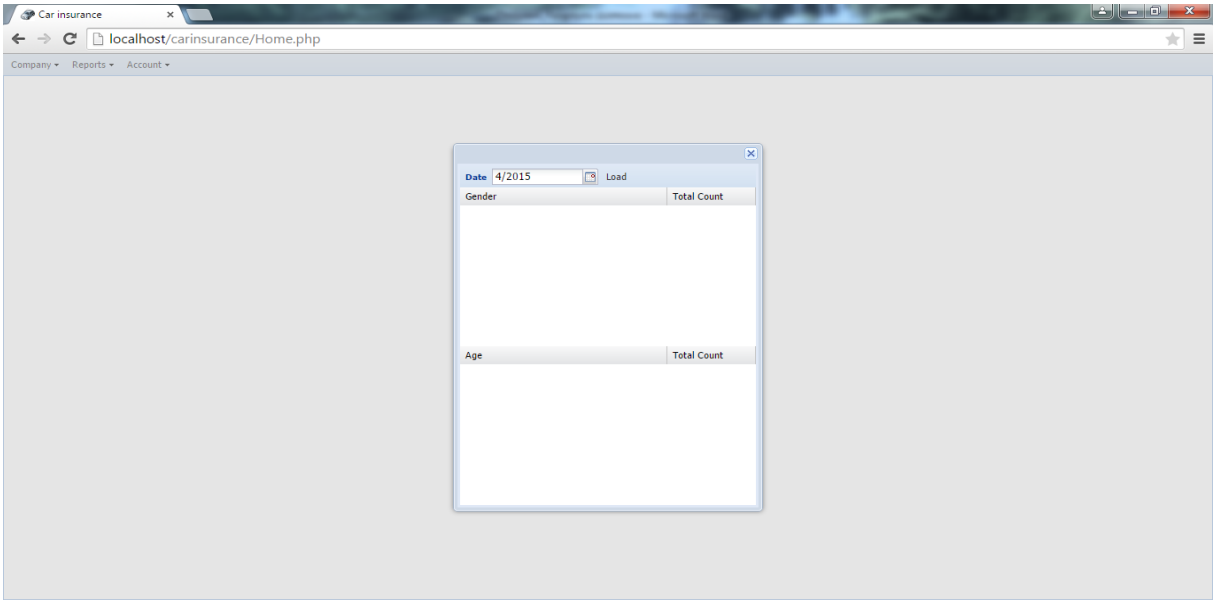
Gender	Income	Expense
Female	508.7	40.0
Male	563.0	100.0

Age	Income	Expense
< 20	53.3	
20-29	388.8	100.0
30-39	192.0	
40-49	283.3	40.0
50-59	104.3	
60-69	33.3	
70-79	16.7	
>80		

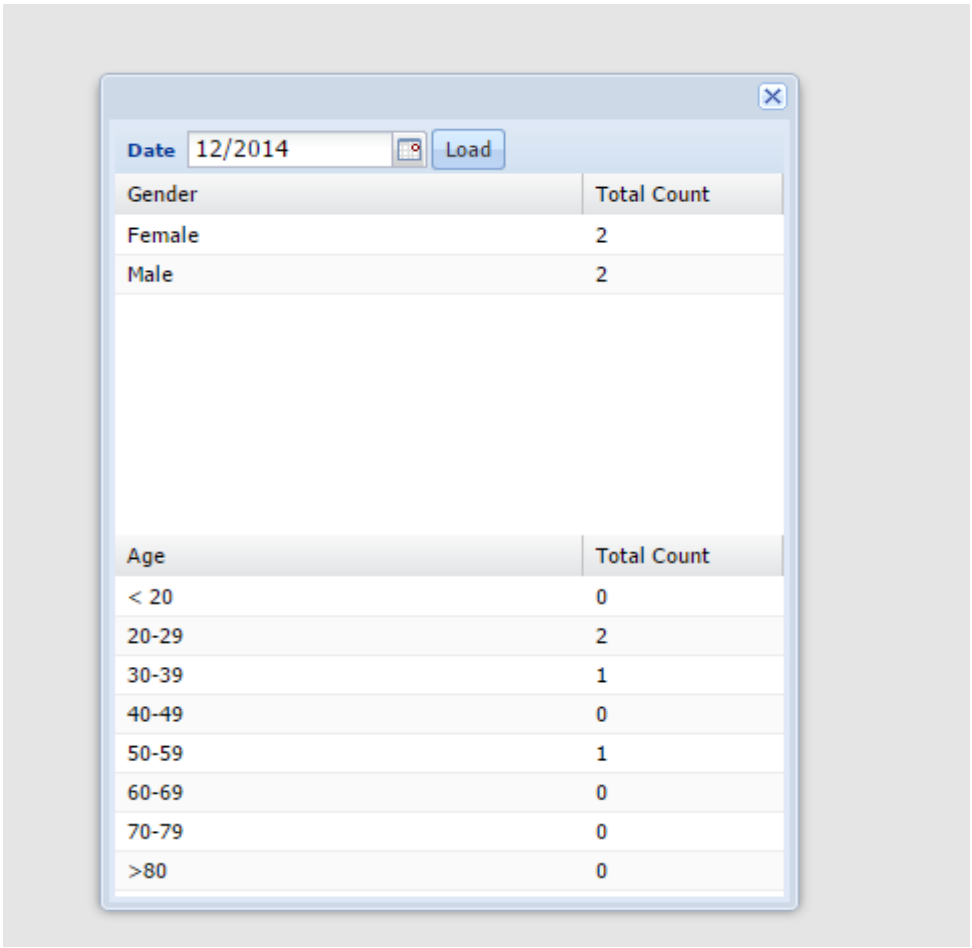
Εικόνα 57:Έσοδα - έξοδα ανά κατηγορία οχήματος ανα φύλο και ανα ηλικία 1/2015

Στον παραπάνω πίνακα φαίνονται τα έσοδα και τα έξοδα ανά κατηγορία οχήματος, ανά φύλο και ανά ηλικία τον 1/2015.
Ο χρήστης μπορεί να επιλέξει όποιο μήνα θέλει και να πατήσει load για να δει τα έσοδα και τα έξοδα ανά κατηγορία οχήματος, ανά φύλο και ανά ηλικία τον συγκεκριμένο μήνα που θέλει.

7.5.7 Συγκεντρωτική αναφορά τα ατυχήματα αν φύλλο πελατών και αν ηλικία(Groupped Accidents)



Εικόνα 58:Groupped Accidents

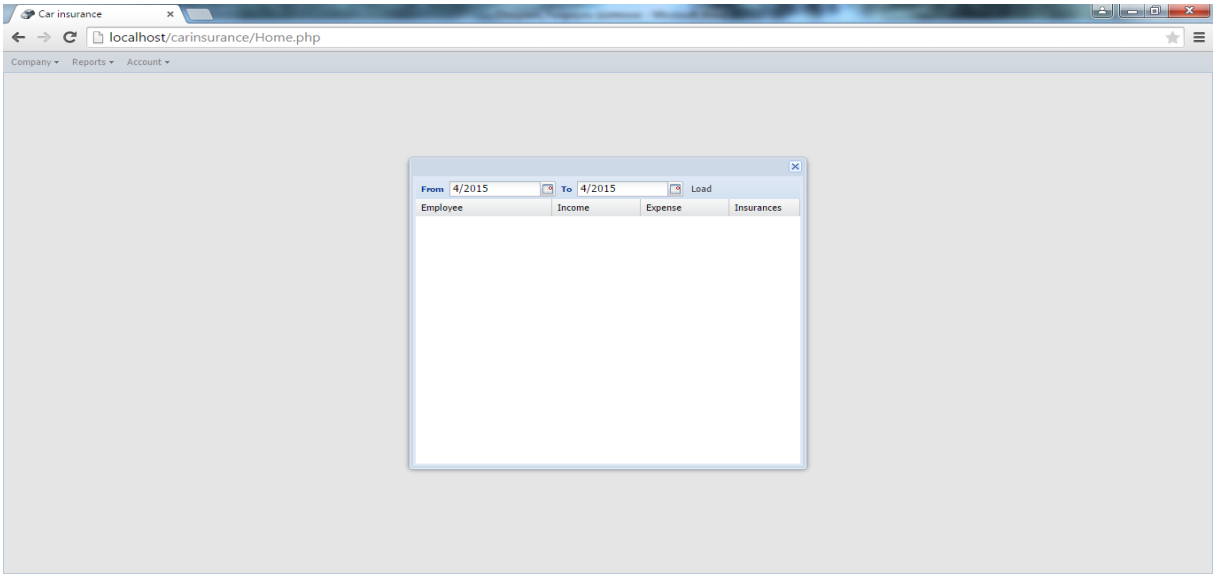


Εικόνα 59:Αριθμός ατυχημάτων ανά φύλο και ανά ηλικία πελατών

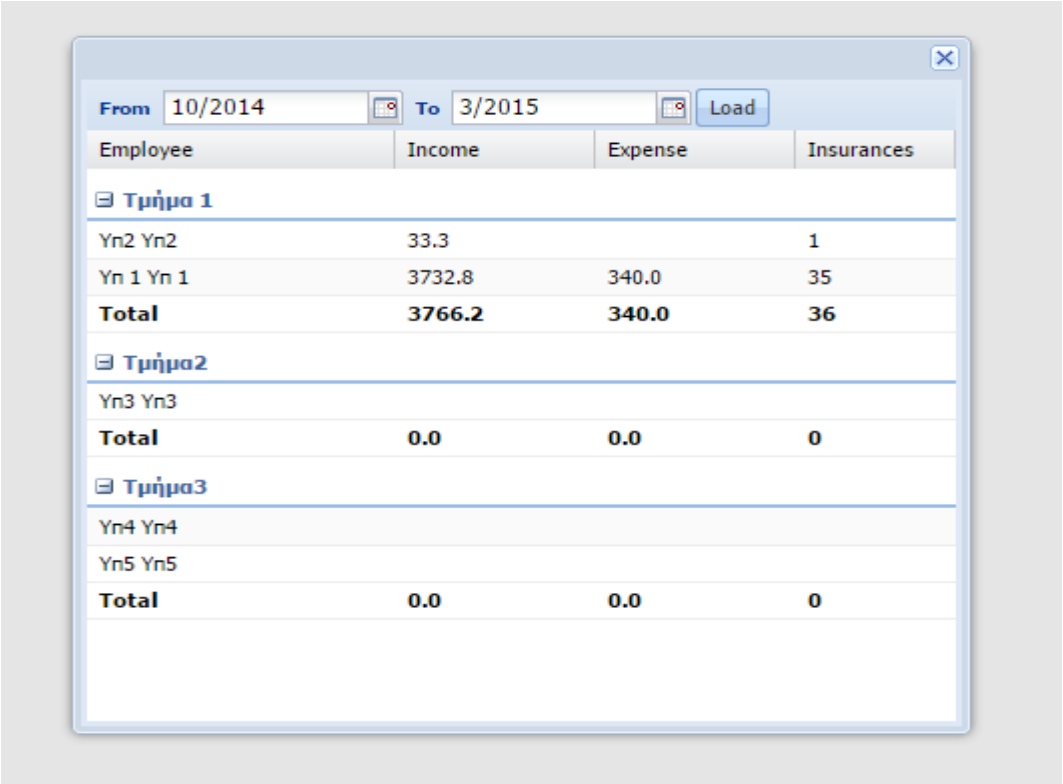
Στον παραπάνω πίνακα φαίνονται τα ατυχήματα ανά φύλο πελατών και ανά ηλικία τον 12/2014.

Ο χρήστης μπορεί να επιλέξει όποιο μήνα θέλει και να πατήσει load για να δει τα ατυχήματα ανά φύλλο πελατών και ανά ηλικία τον συγκεκριμένο μήνα που θέλει.

7.5.8 Συγκεντρωτική αναφορά για τα έσοδα, έξοδα και συμβόλαια που κάνει κάθε υπάλληλος/τμήμα σε ένα συγκεκριμένο χρονικό διάστημα(Groupped Balance by Employee).



Εικόνα 60:Groupped Balance by Employee



Εικόνα 61:Έσοδα- έξοδα- συμβόλαια υπαλλήλου ανά τμήμα

Στον παραπάνω πίνακα φαίνονται τα έσοδα, τα έξοδα και συμβόλαια που έχει κάνει ο κάθε υπάλληλος ανά τμήμα από 10/2014-3/2015

Ο χρήστης μπορεί να επιλέξει όποιο διάστημα θέλει και να πατήσει load για να δει τα έσοδα, τα έξοδα και τα συμβόλαια που έχει κάνει ο κάθε υπάλληλος ανά τμήμα το συγκεκριμένο διάστημα που θέλει.

Βιβλιογραφία

Διαδίκτυο

http://el.wikipedia.org/wiki/%CE%94%CE%B9%CE%B1%CE%B4%CE%AF%CE%BA%CF%84%CF%85%CE%BF

http://en.wikipedia.org/wiki/World_Wide_Web

http://el.wikipedia.org/wiki/WorldWideWeb

http://el.wikipedia.org/wiki/%CE%99%CF%83%CF%84%CE%BF%CF%83%CE%B5%CE%BB%CE%AF%CE%B4%CE%B1

http://el.wikipedia.org/wiki/%CE%A3%CF%84%CE%B1%CF%84%CE%B9%CE%BA%CE%AE_%CE%B9%CF%83%CF%84%CE%BF%CF%83%CE%B5%CE%BB%CE%AF%CE%B4%CE%B1

http://el.wikipedia.org/wiki/%CE%94%CF%85%CE%BD%CE%B1%CE%BC%CE%B9%CE%BA%CE%AE_%CE%B9%CF%83%CF%84%CE%BF%CF%83%CE%B5%CE%BB%CE%AF%CE%B4%CE%B1

http://el.wikipedia.org/wiki/PHP

http://el.wikipedia.org/wiki/JavaScript

http://en.wikipedia.org/wiki/Ajax_%28programming%29

http://el.wikipedia.org/wiki/CSS

http://el.wikipedia.org/wiki/MySQL

http://el.wikipedia.org/wiki/Apache_HTTP_%CE%B5%CE%BE%CF%85%CF%80%CE%B7%CF%81%CE%B5%CF%84%CE%B7%CF%84%CE%AE%CF%82

http://en.wikipedia.org/wiki/PhpMyAdmin

http://el.wikipedia.org/wiki/XAMPP

http://docs.sencha.com/extjs/4.2.0/#!/example

Βιβλία

“Ext JS in Action”, Second Edition Jesus Garcia, Grgur Grisogono , and JacobK.Andresen

“PHP5 and MySQL Bible” Tim Converse, Joyce Park, Clark Morgan

ΠΑΡΑΡΤΗΜΑ

Αρχείο config.php

```
<?php
	$config['dbhost'] = "localhost";
	$config['dbport'] = 3306;
	$config['dbuser'] = "root";
	$config['dbpassword'] = "a1b2c3";
	$config['dbname'] = "carinsurance";
	$config['adminusername'] = "admin";
	$config['adminpassword'] = "admin123";
?>
```

Αρχείο CreateDB.sql

```
/* INNER STORED PROCEDURES */
/* Calculate monthly income for specific insurance */
/* Result is stored in tmpIncomeUngroupped temporary table */

DELIMITER $$

CREATE PROCEDURE CalculateInsuranceMonthlyIncome(IN fromDate DATETIME, IN insuranceID INT)
BEGIN

  DECLARE clientID INT DEFAULT 0;
  DECLARE firstName VARCHAR(50);
  DECLARE lastName VARCHAR(50);
  DECLARE gender BIT DEFAULT FALSE;
  DECLARE age INT DEFAULT 0;
  DECLARE vehicleCategoryID INT DEFAULT 0;
  DECLARE ccFROM INT DEFAULT 0;
  DECLARE ccTO INT DEFAULT 0;
  DECLARE iCost DOUBLE DEFAULT 0;
  DECLARE iDate DATETIME;
  DECLARE employeeID INT DEFAULT 0;
  DECLARE eFirstName VARCHAR(50);
  DECLARE eLastName VARCHAR(50);
  DECLARE departmentID INT DEFAULT 0;
  DECLARE departmentName VARCHAR(50);

  DECLARE isFirstYear INT DEFAULT 0;
  DECLARE isNewDriver INT DEFAULT 0;
  DECLARE catCost DOUBLE DEFAULT 0;
  DECLARE discount DOUBLE DEFAULT NULL;
  DECLARE penalty DOUBLE DEFAULT NULL;
  DECLARE lastAccidentDate DATETIME DEFAULT NULL;
  DECLARE hasAccidentsLastYear INT DEFAULT 0;
  DECLARE hasTwoOrMoreCars INT DEFAULT 0;
  DECLARE income DOUBLE DEFAULT 0;

  SELECT insCost/12, insDate, DATE_ADD(insDate, INTERVAL 1 YEAR) >= fromDate, DATE_ADD(cliLicenceDate, INTERVAL 6 MONTH) >= fromDate, cliClientID, cliGender, cliAge, cliFirstName, cliLastName,vecVehicleCategoryID, vecCCFrom, vecCCTO, vecSemesterCost/6,
```

```
  empEmployeeID, empFirstName, empLastName, depDepartmentID, depName
   INTO iCost, iDate, isFirstYear, isNewDriver, clientID, gender, age, firstName, lastName,
vehicleCategoryID, ccFrom, ccTO, catCost,
   employeeID, eFirstName, eLastName, departmentID, departmentName
    FROM insurance
    JOIN vehicle ON insVehicleID=vehVehicleID
    JOIN vehiclecategory ON vehVehicleCategoryID=vecVehicleCategoryID
    JOIN client ON vehClientID=cliClientID
    JOIN employee ON insEmployeeID=empEmployeeID
    JOIN department ON empDepartmentID=depDepartmentID
    WHERE insInsuranceID=insuranceID;

 SELECT COUNT(*) > 1 INTO hasTwoOrMoreCars  FROM vehicle WHERE vehClientID=clientID
AND vehRegisterDate IS NOT NULL;

 /* special case: first year */
 IF isFirstYear = 1 THEN

  SET income = catCost;
  IF isNewDriver = 1 THEN
   SET income = income * 1.2;
  END IF;

 /* general case: not first year */
 ELSE

  /* check last accident date that client has involved to (responsible or not) */
  SELECT MAX(accDate) INTO lastAccidentDate FROM accident
  JOIN accidentvehicle ON accAccidentID=acvAccidentID
  JOIN vehicle ON acvVehicleID=vehVehicleID
  JOIN client ON vehClientID=cliClientID
  WHERE cliClientID=clientID;

  IF lastAccidentDate IS NOT NULL THEN
   SET discount = LEAST((FLOOR(DATEDIFF(fromDate, lastAccidentDate)/365)) * 0.1, 0.3);
   IF discount = 0 THEN
    SET hasAccidentsLastYear = 1;
   END IF;
  ELSE
   SET discount = LEAST((FLOOR(DATEDIFF(fromDate, iDate)/365)) * 0.1, 0.3);
  END IF;

  /* has accidents, add penalty */
  IF hasAccidentsLastYear = 1 THEN

   SELECT COUNT(*) * 0.3 INTO penalty FROM accident
   JOIN accidentvehicle ON accAccidentID=acvAccidentID
   JOIN vehicle ON acvVehicleID=vehVehicleID
   JOIN client ON vehClientID=cliClientID
   WHERE cliClientID=clientID AND acvIsResponsible=1 AND DATEDIFF(fromDate, accDate) <
365;

   SET income = iCost * (1+penalty);

  /* has not accidents, make discount */
```

```
   ELSE

     IF hasTwoOrMoreCars = 1 THEN
       SET discount = discount + 0.05;
     END IF;

     SET income = iCost * (1-discount);

   END IF;

 END IF;

 INSERT INTO tmpIncomeUngroupped(clientID, firstName, lastName, gender, age,
vehicleCategoryID, ccFROM, ccTO, employeeID, eFirstName, eLastName, departmentID,
departmentName, income)
   VALUES(clientID, firstName, lastName, gender, age, vehicleCategoryID, ccFROM, ccTO,
employeeID, eFirstName, eLastName, departmentID, departmentName, income);
END
$$

DELIMITER ;

/* Calculate monthly income for all insurances grouped by specific field */
/* Results are stored in tmpIncome temporary table */

DELIMITER $$

CREATE PROCEDURE CalculateMonthlyIncome(IN fromDate DATETIME, IN groupBy
VARCHAR(50))
BEGIN

 DECLARE insuranceID INT;
 DECLARE done INT DEFAULT 0;
 DECLARE insuranceCursor CURSOR FOR SELECT insInsuranceID FROM insurance WHERE
insDate <= fromDate;
 DECLARE CONTINUE HANDLER FOR NOT FOUND SET done=1;

 DROP TEMPORARY TABLE IF EXISTS tmpIncomeUngroupped;
 CREATE TEMPORARY TABLE IF NOT EXISTS tmpIncomeUngroupped
 (
   clientID INT, firstName VARCHAR(50), lastName VARCHAR(50), gender BIT, age INT,
vehicleCategoryID INT, ccFROM INT, ccTO INT, employeeID INT,
   eFirstName VARCHAR(50), eLastName VARCHAR(50), departmentID INT, departmentName
VARCHAR(50), income DOUBLE
 ) ENGINE=MEMORY;

 /* loop all insurances and add a row in tmpIncomeUngroupped */
 OPEN insuranceCursor;
 insurancesLoop: LOOP
   FETCH insuranceCursor INTO insuranceID;
   IF done = 1 THEN LEAVE insurancesLoop; END IF;

   /* find cost for insurance */
   CALL CalculateInsuranceMonthlyIncome(fromDate, insuranceID);
```

```
  END LOOP insurancesLoop;
  CLOSE insuranceCursor;

  /* group results */
  SET @queryText = CONCAT(CONCAT('INSERT INTO tmpIncome (SELECT clientID, firstName,
lastName, gender, age, vehicleCategoryID, ccFROM, ccTO, employeeID,
    eFirstName, eLastName, departmentID, departmentName, SUM(income) FROM
tmpIncomeUngroupped GROUP BY ', groupBy), ')');
  PREPARE stmt FROM @queryText;
  EXECUTE stmt;

  DROP TEMPORARY TABLE IF EXISTS tmpIncomeUngroupped;

END
$$

DELIMITER ;


/* Calculate expenses for specific month grouped by specific field */
/* Result is stored in tmpExpense temporary table */

DELIMITER $$

CREATE PROCEDURE CalculateMonthlyExpenses(IN fromDate DATETIME, IN groupBy
VARCHAR(50))
BEGIN

  DROP TEMPORARY TABLE IF EXISTS tmpExpenseUngroupped;
  CREATE TEMPORARY TABLE IF NOT EXISTS tmpExpenseUngroupped
  (
    clientID INT, firstName VARCHAR(50), lastName VARCHAR(50), gender BIT, age INT,
vehicleCategoryID INT, ccFROM INT, ccTO INT, employeeID INT,
    eFirstName VARCHAR(50), eLastName VARCHAR(50), departmentID INT, departmentName
VARCHAR(50), expense DOUBLE
  ) ENGINE=MEMORY;

  INSERT INTO tmpExpenseUngroupped(clientID, firstName, lastName, gender, age,
vehicleCategoryID, ccFROM, ccTO,
    employeeID, eFirstName, eLastName, departmentID, departmentName, expense)
    SELECT cliClientID, cliFirstName, cliLastName, cliGender, cliAge, vecVehicleCategoryID,
vecCCFrom, vecCCTo,
        empEmployeeID, empFirstName, empLastName, depDepartmentID, depName,
SUM(acvDamageCost) FROM client
        LEFT JOIN
        (SELECT accAccidentID, MIN(accEmployeeID) AS employeeID, MIN(cliClientID) AS
clientID FROM accident
        JOIN accidentvehicle ON accAccidentID=acvAccidentID
        JOIN vehicle ON acvVehicleID=vehVehicleID
        JOIN client ON vehClientID=cliClientID
        WHERE acvIsResponsible=1 AND vehRegisterDate IS NOT NULL AND
accDate>=fromDate AND accDate<DATE_ADD(fromDate, INTERVAL 1 MONTH)
        GROUP BY accAccidentID) ACC ON cliClientID=ACC.clientID
        JOIN accidentvehicle ON ACC.accAccidentID=acvAccidentID
        JOIN vehicle ON acvVehicleID=vehVehicleID
```

```
        JOIN vehiclecategory ON vehVehicleCategoryID=vecVehicleCategoryID
        JOIN employee ON ACC.employeeID=empEmployeeID
        JOIN department ON empDepartmentID=depDepartmentID
        WHERE vehRegisterDate IS NULL AND acvIsResponsible=0
        GROUP BY acvAccidentID;

 /* group results */
 SET @queryText = CONCAT(CONCAT('INSERT INTO tmpExpense (SELECT clientID, firstName, lastName, gender, age, vehicleCategoryID, ccFROM, ccTO, employeeID,
   eFirstName, eLastName, departmentID, departmentName, SUM(expense) FROM tmpExpenseUngroupped GROUP BY ', groupBy), ')');
 PREPARE stmt FROM @queryText;
 EXECUTE stmt;

 DROP TEMPORARY TABLE IF EXISTS tmpExpenseUngroupped;
END
$$

DELIMITER ;


/* OUTER STORED PROCEDURES (these are called from PHP code) */
/* Balance by client */

DELIMITER $$

CREATE PROCEDURE CalculateMonthlyBalanceByClient(IN fromDate DATETIME)
BEGIN

 DROP TEMPORARY TABLE IF EXISTS tmpIncome;
 CREATE TEMPORARY TABLE IF NOT EXISTS tmpIncome
 (
   clientID INT, firstName VARCHAR(50), lastName VARCHAR(50), gender BIT, age INT, vehicleCategoryID INT, ccFROM INT, ccTO INT, employeeID INT,
   eFirstName VARCHAR(50), eLastName VARCHAR(50), departmentID INT, departmentName VARCHAR(50), income DOUBLE
 ) ENGINE=MEMORY;

 DROP TEMPORARY TABLE IF EXISTS tmpExpense;
 CREATE TEMPORARY TABLE IF NOT EXISTS tmpExpense
 (
   clientID INT, firstName VARCHAR(50), lastName VARCHAR(50), gender BIT, age INT, vehicleCategoryID INT, ccFROM INT, ccTO INT, employeeID INT,
   eFirstName VARCHAR(50), eLastName VARCHAR(50), departmentID INT, departmentName VARCHAR(50), expense DOUBLE
 ) ENGINE=MEMORY;

 CALL CalculateMonthlyIncome(fromDate, 'clientID');
 CALL CalculateMonthlyExpenses(fromDate, 'clientID');

 SELECT CONCAT(cliLastName, CONCAT(' ', cliFirstName)) AS client, tmpIncome.income AS income, tmpExpense.expense AS expense FROM client
 LEFT JOIN tmpIncome ON cliClientID=tmpIncome.clientID
 LEFT JOIN tmpExpense ON tmpIncome.clientID=tmpExpense.clientID
 ORDER BY tmpIncome.lastName, tmpIncome.firstName;
```

```
  DROP TEMPORARY TABLE IF EXISTS tmpIncome;
  DROP TEMPORARY TABLE IF EXISTS tmpExpense;

END
$$

DELIMITER ;

/* Balance by vehicle category */
DELIMITER $$

CREATE PROCEDURE CalculateMonthlyBalanceByVehicleCategory(IN fromDate DATETIME)
BEGIN

  CREATE TEMPORARY TABLE IF NOT EXISTS tmpIncome
  (
    clientID INT, firstName VARCHAR(50), lastName VARCHAR(50), gender BIT, age INT, vehicleCategoryID INT, ccFROM INT, ccTO INT, employeeID INT,
    eFirstName VARCHAR(50), eLastName VARCHAR(50), departmentID INT, departmentName VARCHAR(50), income DOUBLE
  ) ENGINE=MEMORY;

  CREATE TEMPORARY TABLE IF NOT EXISTS tmpExpense
  (
    clientID INT, firstName VARCHAR(50), lastName VARCHAR(50), gender BIT, age INT, vehicleCategoryID INT, ccFROM INT, ccTO INT, employeeID INT,
    eFirstName VARCHAR(50), eLastName VARCHAR(50), departmentID INT, departmentName VARCHAR(50), expense DOUBLE
  ) ENGINE=MEMORY;

  CALL CalculateMonthlyIncome(fromDate, 'vehicleCategoryID');
  CALL CalculateMonthlyExpenses(fromDate, 'vehicleCategoryID');

  SELECT CONCAT(vecCCFrom, CONCAT(' - ', vecCCTo)) AS category, SUM(tmpIncome.income) AS income, SUM(tmpExpense.expense) AS expense FROM vehicleCategory
  LEFT JOIN tmpIncome ON vecVehicleCategoryID=tmpIncome.vehicleCategoryID
  LEFT JOIN tmpExpense ON vecVehicleCategoryID=tmpExpense.vehicleCategoryID
  GROUP BY vecVehicleCategoryID
  ORDER BY vecCCFrom;

  DROP TEMPORARY TABLE IF EXISTS tmpIncome;
  DROP TEMPORARY TABLE IF EXISTS tmpExpense;

END
$$

DELIMITER ;

/* Balance by client gender */
DELIMITER $$

CREATE PROCEDURE CalculateMonthlyBalanceByGender(IN fromDate DATETIME)
BEGIN
```

```
 DROP TEMPORARY TABLE IF EXISTS tmpIncome;
 CREATE TEMPORARY TABLE IF NOT EXISTS tmpIncome
 (
   clientID INT, firstName VARCHAR(50), lastName VARCHAR(50), gender BIT, age INT, vehicleCategoryID INT, ccFROM INT, ccTO INT, employeeID INT,
   eFirstName VARCHAR(50), eLastName VARCHAR(50), departmentID INT, departmentName VARCHAR(50), income DOUBLE
 ) ENGINE=MEMORY;

 DROP TEMPORARY TABLE IF EXISTS tmpExpense;
 CREATE TEMPORARY TABLE IF NOT EXISTS tmpExpense
 (
   clientID INT, firstName VARCHAR(50), lastName VARCHAR(50), gender BIT, age INT, vehicleCategoryID INT, ccFROM INT, ccTO INT, employeeID INT,
   eFirstName VARCHAR(50), eLastName VARCHAR(50), departmentID INT, departmentName VARCHAR(50), expense DOUBLE
 ) ENGINE=MEMORY;

 DROP TEMPORARY TABLE IF EXISTS tmpGender;
 CREATE TEMPORARY TABLE IF NOT EXISTS tmpGender(gender BIT, income DOUBLE NULL, expense DOUBLE NULL) ENGINE=MEMORY;

 CALL CalculateMonthlyIncome(fromDate, 'gender');
 CALL CalculateMonthlyExpenses(fromDate, 'gender');

 INSERT INTO tmpGender(gender, income, expense)
 SELECT 0, SUM(tmpIncome.income) AS income, SUM(tmpExpense.expense) AS expense  FROM client
 LEFT JOIN tmpIncome ON cliClientID=tmpIncome.clientID
 LEFT JOIN tmpExpense ON cliClientID=tmpExpense.clientID
 WHERE cliGender=0;

 INSERT INTO tmpGender(gender, income, expense)
 SELECT 1, SUM(tmpIncome.income) AS income, SUM(tmpExpense.expense) AS expense  FROM client
 LEFT JOIN tmpIncome ON cliClientID=tmpIncome.clientID
 LEFT JOIN tmpExpense ON cliClientID=tmpExpense.clientID
 WHERE cliGender=1;

 SELECT * FROM tmpGender;

 DROP TEMPORARY TABLE IF EXISTS tmpIncome;
 DROP TEMPORARY TABLE IF EXISTS tmpExpense;
 DROP TEMPORARY TABLE IF EXISTS tmpGender;

END
$$

DELIMITER ;

/* Balance by client age */
DELIMITER $$

CREATE PROCEDURE CalculateMonthlyBalanceByAge(IN fromDate DATETIME)
BEGIN
```

```
 DROP TEMPORARY TABLE IF EXISTS tmpIncome;
 CREATE TEMPORARY TABLE IF NOT EXISTS tmpIncome
 (
   clientID INT, firstName VARCHAR(50), lastName VARCHAR(50), gender BIT, age INT, vehicleCategoryID INT, ccFROM INT, ccTO INT, employeeID INT,
   eFirstName VARCHAR(50), eLastName VARCHAR(50), departmentID INT, departmentName VARCHAR(50), income DOUBLE
 ) ENGINE=MEMORY;

 DROP TEMPORARY TABLE IF EXISTS tmpExpense;
 CREATE TEMPORARY TABLE IF NOT EXISTS tmpExpense
 (
   clientID INT, firstName VARCHAR(50), lastName VARCHAR(50), gender BIT, age INT, vehicleCategoryID INT, ccFROM INT, ccTO INT, employeeID INT,
   eFirstName VARCHAR(50), eLastName VARCHAR(50), departmentID INT, departmentName VARCHAR(50), expense DOUBLE
 ) ENGINE=MEMORY;

 CALL CalculateMonthlyIncome(fromDate, 'age');
 CALL CalculateMonthlyExpenses(fromDate, 'age');

 CREATE TEMPORARY TABLE IF NOT EXISTS ageRange (fromAge INT, toAge INT, description VARCHAR(50)) ENGINE=MEMORY;
 INSERT INTO ageRange(fromAge, toAge, description) VALUES(0, 19, '< 20');
 INSERT INTO ageRange(fromAge, toAge, description) VALUES(20, 29, '20-29');
 INSERT INTO ageRange(fromAge, toAge, description) VALUES(30, 39, '30-39');
 INSERT INTO ageRange(fromAge, toAge, description) VALUES(40, 49, '40-49');
 INSERT INTO ageRange(fromAge, toAge, description) VALUES(50, 59, '50-59');
 INSERT INTO ageRange(fromAge, toAge, description) VALUES(60, 69, '60-69');
 INSERT INTO ageRange(fromAge, toAge, description) VALUES(70, 79, '70-79');
 INSERT INTO ageRange(fromAge, toAge, description) VALUES(80, 120, '>80');

 SELECT description AS age, SUM(tmpIncome.income) AS income, SUM(tmpExpense.expense) AS expense FROM ageRange
 LEFT JOIN client ON cliAge>=fromAge AND cliAge<=toAge
 LEFT JOIN tmpIncome ON cliClientID=tmpIncome.clientID
 LEFT JOIN tmpExpense ON cliClientID=tmpExpense.clientID
 GROUP BY fromAge
 ORDER BY fromAge;

 DROP TEMPORARY TABLE IF EXISTS tmpIncome;
 DROP TEMPORARY TABLE IF EXISTS tmpExpense;
 DROP TEMPORARY TABLE IF EXISTS ageRange;

END
$$

DELIMITER ;

/* Accidents by client gender */
DELIMITER $$

CREATE PROCEDURE ViewAccidentsByGender(IN fromDate DATETIME)
BEGIN
```

```
 DROP TEMPORARY TABLE IF EXISTS tmpGender;
 CREATE TEMPORARY TABLE IF NOT EXISTS tmpGender(gender BIT, count INT NULL) ENGINE=MEMORY;

 INSERT INTO tmpGender(gender, count)
 SELECT 0, COUNT(cliClientID) AS count
  FROM accident
  JOIN accidentvehicle ON accAccidentID=acvAccidentID
  JOIN vehicle ON acvVehicleID=vehVehicleID
  JOIN client ON vehClientID=cliClientID
  WHERE vehRegisterDate IS NOT NULL AND accDate>=fromDate AND accDate<DATE_ADD(fromDate, INTERVAL 1 MONTH) AND cliGender=0;

 INSERT INTO tmpGender(gender, count)
 SELECT 1, COUNT(cliClientID) AS count
  FROM accident
  JOIN accidentvehicle ON accAccidentID=acvAccidentID
  JOIN vehicle ON acvVehicleID=vehVehicleID
  JOIN client ON vehClientID=cliClientID
  WHERE vehRegisterDate IS NOT NULL AND accDate>=fromDate AND accDate<DATE_ADD(fromDate, INTERVAL 1 MONTH) AND cliGender=1;

 SELECT * FROM tmpGender;

 DROP TEMPORARY TABLE IF EXISTS tmpGender;
END
$$

DELIMITER ;

/* Accidents by client age */
DELIMITER $$

CREATE PROCEDURE ViewAccidentsByAge(IN fromDate DATETIME)
BEGIN

 DROP TEMPORARY TABLE IF EXISTS tmpAccident;
 CREATE TEMPORARY TABLE IF NOT EXISTS tmpAccident (age INT) ENGINE=MEMORY;

 INSERT INTO tmpAccident(age)
 SELECT cliAge
  FROM accident
  JOIN accidentvehicle ON accAccidentID=acvAccidentID
  JOIN vehicle ON acvVehicleID=vehVehicleID
  JOIN client ON vehClientID=cliClientID
  WHERE vehRegisterDate IS NOT NULL AND accDate>=fromDate AND accDate<DATE_ADD(fromDate, INTERVAL 1 MONTH) GROUP BY cliAge;

 CREATE TEMPORARY TABLE IF NOT EXISTS ageRange (fromAge INT, toAge INT, description VARCHAR(50)) ENGINE=MEMORY;
 INSERT INTO ageRange(fromAge, toAge, description) VALUES(0, 19, '< 20');
 INSERT INTO ageRange(fromAge, toAge, description) VALUES(20, 29, '20-29');
 INSERT INTO ageRange(fromAge, toAge, description) VALUES(30, 39, '30-39');
 INSERT INTO ageRange(fromAge, toAge, description) VALUES(40, 49, '40-49');
 INSERT INTO ageRange(fromAge, toAge, description) VALUES(50, 59, '50-59');
```

```
  INSERT INTO ageRange(fromAge, toAge, description) VALUES(60, 69, '60-69');
  INSERT INTO ageRange(fromAge, toAge, description) VALUES(70, 79, '70-79');
  INSERT INTO ageRange(fromAge, toAge, description) VALUES(80, 120, '>80');

  SELECT description AS age, COUNT(tmpAccident.age) AS count FROM ageRange
  LEFT JOIN tmpAccident ON tmpAccident.age >= fromAge AND tmpAccident.age <= toAge
  GROUP BY fromAge
  ORDER BY fromAge;

  DROP TEMPORARY TABLE IF EXISTS tmpAccident;
  DROP TEMPORARY TABLE IF EXISTS ageRange;

END
$$

DELIMITER ;

/* Balance by employees ordered by departments */
DELIMITER $$

CREATE PROCEDURE CalculateMonthlyRangeBalanceByEmployee(IN fromDate DATETIME, IN
toDate DATETIME)
BEGIN

  DECLARE currentDate DATETIME;
  SET currentDate = fromDate;

  DROP TEMPORARY TABLE IF EXISTS tmpIncome;
  CREATE TEMPORARY TABLE IF NOT EXISTS tmpIncome
  (
    clientID INT, firstName VARCHAR(50), lastName VARCHAR(50), gender BIT, age INT,
vehicleCategoryID INT, ccFROM INT, ccTO INT, employeeID INT,
    eFirstName VARCHAR(50), eLastName VARCHAR(50), departmentID INT, departmentName
VARCHAR(50), income DOUBLE
  ) ENGINE=MEMORY;

  DROP TEMPORARY TABLE IF EXISTS tmpExpense;
  CREATE TEMPORARY TABLE IF NOT EXISTS tmpExpense
  (
    clientID INT, firstName VARCHAR(50), lastName VARCHAR(50), gender BIT, age INT,
vehicleCategoryID INT, ccFROM INT, ccTO INT, employeeID INT,
    eFirstName VARCHAR(50), eLastName VARCHAR(50), departmentID INT, departmentName
VARCHAR(50), expense DOUBLE
  ) ENGINE=MEMORY;

  DROP TEMPORARY TABLE IF EXISTS tmpEmployee;
  CREATE TEMPORARY TABLE IF NOT EXISTS tmpEmployee (empID INT, firstName
VARCHAR(50), lastName VARCHAR(50), depID INT, dep VARCHAR(50), income DOUBLE
NULL, expense DOUBLE NULL) ENGINE=MEMORY;

  DROP TEMPORARY TABLE IF EXISTS tmpContracts;
  CREATE TEMPORARY TABLE IF NOT EXISTS tmpContracts (empID INT, contracts INT
NULL) ENGINE=MEMORY;

  WHILE currentDate < toDate DO
```

```
    TRUNCATE TABLE tmpIncome;
    TRUNCATE TABLE tmpExpense;
    CALL CalculateMonthlyIncome(currentDate, 'employeeID');
    CALL CalculateMonthlyExpenses(currentDate, 'employeeID');

    INSERT INTO tmpEmployee(empID, firstName, lastName, depID, dep, income, expense)
    SELECT empEmployeeID AS empID, empFirstName AS firstName, empLastName AS lastName, empDepartmentID AS depID, depName AS dep,  SUM(tmpIncome.income) AS income, SUM(tmpExpense.expense) AS expense FROM employee
    JOIN department ON empDepartmentID=depDepartmentID
    LEFT JOIN tmpIncome ON empEmployeeID=tmpIncome.employeeID
    LEFT JOIN tmpExpense ON empEmployeeID=tmpExpense.employeeID
    GROUP BY empEmployeeID;

    SET currentDate = DATE_ADD(currentDate, INTERVAL 1 MONTH);
  END WHILE;

  INSERT INTO tmpContracts(empID, contracts)
    (SELECT empEmployeeID, EMP.contracts FROM employee LEFT JOIN
    (SELECT insEmployeeID, COUNT(*) AS contracts FROM insurance WHERE insDate>=fromDate AND insDate<=toDate GROUP BY insEmployeeID) EMP ON empEmployeeID=EMP.insEmployeeID);

  SELECT CONCAT(lastName, CONCAT(' ', firstName)) AS empName, depID, dep AS depName, SUM(income) AS income, SUM(expense) AS expense, tmpContracts.contracts AS count FROM tmpEmployee
    LEFT JOIN tmpContracts ON tmpEmployee.empID=tmpContracts.empID
  GROUP BY tmpEmployee.empID
  ORDER BY depID;

  DROP TEMPORARY TABLE IF EXISTS tmpIncome;
  DROP TEMPORARY TABLE IF EXISTS tmpExpense;
  DROP TEMPORARY TABLE IF EXISTS tmpEmployee;
  DROP TEMPORARY TABLE IF EXISTS tmpContracts;
END
$$

DELIMITER ;
```

Αρχείο data.php

```
<?php

        define("PDOUNIQUECONSTRAINTERROR", 1062);
        define("PDOFOREIGNKEYERROR", 23000);

        class NotFoundException extends Exception { }
        class UniqueException extends Exception { }
        class ForeignKeyException extends Exception { }
        class UniqueDirectorException extends Exception { }
        class VehicleCategoryRangeException extends Exception { }

        function getEpochFormat() { return "U"; }
        function getIsoDateFormat() { return "Y-m-d\TH:i:s"; }
        function getMySQLDateTimeFormat() { return "Y-m-d H:i:s"; }
```

```
// Department methods


function GetDepartments()
{
	global $db;

	$results = array();

	try
	{
		$db->beginTransaction();

		$stmt = $db->query("SELECT * FROM department ORDER BY depName");
		$rows = $stmt->fetchAll(PDO::FETCH_OBJ);

		for($i=0; $i<count($rows); $i++)
		{
			$result = new Department();
			$result->ID = $rows[$i]->depDepartmentID;
			$result->Name = $rows[$i]->depName;
			$result->Location = $rows[$i]->depLocation;
			$results[] = $result;
		}

		$db->commit();
	}
	catch(PDOException $ex)
	{
		$db->rollBack();
		echo $ex->getMessage();
	}

	return $results;
}

function GetDepartment($id)
{
	global $db;

	$result = null;

	try
	{
		$db->beginTransaction();

		$stmt = $db->query("SELECT * FROM department WHERE
depDepartmentID=".$id);
		$rows = $stmt->fetchAll(PDO::FETCH_OBJ);

		if(count($rows) == 1)
		{
			$result = new Department();
```

```
                        $result->ID = $rows[0]->depDepartmentID;
                        $result->Name = $rows[0]->depName;
                        $result->Location = $rows[0]->depLocation;
                }

                $db->commit();
        }
        catch(PDOException $ex)
        {
                $db->rollBack();
                throw $ex;
        }

        if($result == null)
                throw new NotFoundException();

        return $result;
}

function DeleteDepartments($ids)
{
        global $db;

        try
        {
                $db->beginTransaction();
                $stmt = $db->query("DELETE FROM department WHERE
depDepartmentID IN (".implode(",", $ids).")");
                $db->commit();
        }
        catch(PDOException $ex)
        {
                $db->rollBack();
                if ($e->errorInfo[1] == PDOUNIQUECONSTRAINTERROR)
                        throw new ForeignKeyException();
        }
}

function SaveDepartment($department)
{
        global $db;

        $entityID = $department->ID;

        try
        {
                $db->beginTransaction();

                // update or insert department
                $sql = $department->ID > 0 ?
                "UPDATE department SET".
                " depName='".$department->Name."',".
                " depLocation='".$department->Location."'".
           " WHERE depDepartmentID=".$department->ID :
```

```
                "INSERT INTO department (depName, depLocation) VALUES(".
                "'".$department->Name."',".
                "'".$department->Location."')";

                $db->exec($sql);

                if($entityID  <= 0)
                        $entityID = $db->lastInsertId();

                $db->commit();
        }
        catch(PDOException $ex)
        {
                $db->rollBack();

                if ($ex->errorInfo[1] == PDOUNIQUECONSTRAINTERROR)
                        throw new UniqueException();

                throw $ex;
        }

        return $entityID;
}

function GetDepartmentsLookup()
{
        global $db;

        $results = array();

        try
        {
                $db->beginTransaction();

                $stmt = $db->query("SELECT depDepartmentID, depName FROM
department ORDER BY depName");
                $rows = $stmt->fetchAll(PDO::FETCH_OBJ);

                for($i=0; $i<count($rows); $i++)
                {
                        $result = new LookupItem();
                        $result->ID = $rows[$i]->depDepartmentID;
                        $result->Name = $rows[$i]->depName;
                        $results[] = $result;
                }

                $db->commit();
        }
        catch(PDOException $ex)
        {
                $db->rollBack();
                echo $ex->getMessage();
        }

        return $results;
```

```
	}


	// Employee methods


	function GetEmployees()
	{
		global $db;

		$results = array();

		try
		{
			$db->beginTransaction();

			$stmt = $db->query("SELECT * FROM employee ".
					"JOIN department ON
empDepartmentID=depDepartmentID ORDER BY empLastName");
			$rows = $stmt->fetchAll(PDO::FETCH_OBJ);

			for($i=0; $i<count($rows); $i++)
			{
				$result = new Employee();
				$result->ID = $rows[$i]->empEmployeeID;
				$result->FirstName = $rows[$i]->empFirstName;
				$result->LastName = $rows[$i]->empLastName;
				$result->Address = $rows[$i]->empAddress;
				$result->Phone = $rows[$i]->empPhone;
				$result->IsDirector = intval($rows[$i]->empIsDirector) == 1;
				$result->DepartmentID = $rows[$i]->empDepartmentID;
				$result->DepartmentName = $rows[$i]->depName;
				$result->Username = $rows[$i]->empUsername;
				$results[] = $result;
			}

			$db->commit();
		}
		catch(PDOException $ex)
		{
			$db->rollBack();
			echo $ex->getMessage();
		}

		return $results;
	}

	function GetEmployee($id)
	{
		global $db;

		$result = null;

		try
		{
```

```
                $db->beginTransaction();

                $stmt = $db->query("SELECT * FROM employee ".
                                                "JOIN department ON
empDepartmentID=depDepartmentID WHERE empEmployeeID=".$id);
                $rows = $stmt->fetchAll(PDO::FETCH_OBJ);

                if(count($rows) == 1)
                {
                        $result = new Employee();
                        $result->ID = $rows[0]->empEmployeeID;
                        $result->FirstName = $rows[0]->empFirstName;
                        $result->LastName = $rows[0]->empLastName;
                        $result->Address = $rows[0]->empAddress;
                        $result->Phone = $rows[0]->empPhone;
                        $result->IsDirector = intval($rows[0]->empIsDirector) == 1;
                        $result->DepartmentID = $rows[0]->empDepartmentID;
                        $result->DepartmentName = $rows[0]->depName;
                        $result->Username = $rows[0]->empUsername;
                        $result->Password = $rows[0]->empPassword;
                }

                $db->commit();
        }
        catch(PDOException $ex)
        {
                $db->rollBack();
                throw $ex;
        }

        if($result == null)
                throw new NotFoundException();

        return $result;
    }

    function DeleteEmployees($ids)
    {
        global $db;

        try
        {
                $db->beginTransaction();

                $stmt = $db->query("SELECT COUNT(*) AS cnt FROM employee WHERE
empEmployeeID IN (".implode(",", $ids).") AND empIsDirector=1");
                $rows = $stmt->fetchAll(PDO::FETCH_OBJ);
                if(count($rows) > 0 && $rows[0]->cnt > 0)
                        throw new UniqueDirectorException();

                $stmt = $db->query("DELETE FROM employee WHERE empEmployeeID
IN (".implode(",", $ids).")");
                $db->commit();
        }
        catch(PDOException $ex)
```

```
		{
			$db->rollBack();
			if ($e->errorInfo[1] == PDOUNIQUECONSTRAINTERROR)
				throw new ForeignKeyException();
		}
		catch(Exception $ex)
		{
			$db->rollBack();
			throw $ex;
		}
	}

	function SaveEmployee($employee)
	{
		global $db;

		$entityID = $employee->ID;

		try
		{
			$db->beginTransaction();

			$stmt = $db->query("SELECT empEmployeeID FROM employee WHERE empEmployeeID!=".$employee->ID." AND empDepartmentID=".$employee->DepartmentID." AND empIsDirector=1");
			$rows = $stmt->fetchAll(PDO::FETCH_OBJ);

			// department has already a director, remove him from director
			if($employee->IsDirector && count($rows) > 0)
				$db->exec("UPDATE employee SET empIsDirector=0 WHERE empEmployeeID=".$rows[0]->empEmployeeID);
			// department has not a director, so this employee should be a director
			else if(!$employee->IsDirector && count($rows) == 0)
				throw new UniqueDirectorException();

			// update or insert employee
			$sql = $employee->ID > 0 ?
			"UPDATE employee SET".
			" empFirstName='".$employee->FirstName."',".
			" empLastName='".$employee->LastName."',".
			" empAddress='".$employee->Address."',".
			" empPhone='".$employee->Phone."',".
			" empUsername='".$employee->Username."',".
			" empPassword='".$employee->Password."',".
			" empDepartmentID=".$employee->DepartmentID.",".
			" empIsDirector=".($employee->IsDirector ? 1 : 0).
		  " WHERE empEmployeeID=".$employee->ID :

			"INSERT INTO employee (empFirstName, empLastName, empAddress, empPhone, empUsername, empPassword, empDepartmentID, empIsDirector) VALUES(".
			"'".$employee->FirstName."',".
			"'".$employee->LastName."',".
			"'".$employee->Address."',".
			"'".$employee->Phone."',".
			"'".$employee->Username."',".
```

```
                """.$employee->Password."',".
                $employee->DepartmentID.",".
                ($employee->IsDirector ? 1 : 0).")";

                $db->exec($sql);

                if($entityID  <= 0)
                        $entityID = $db->lastInsertId();

                $db->commit();
        }
        catch(PDOException $ex)
        {
                $db->rollBack();

                if ($ex->errorInfo[1] == PDOUNIQUECONSTRAINTERROR)
                        throw new UniqueException();

                throw $ex;
        }
        catch(Exception $ex)
        {
                $db->rollBack();
                throw $ex;
        }

        return $entityID;
}

function ValidateEmployee($username, $password)
{
        global $db;

        $employeeID = null;

        try
        {
                $db->beginTransaction();

                $stmt = $db->query("SELECT empEmployeeID FROM employee WHERE
empUsername='".$username."' AND empPassword='".$password."'");
                $rows = $stmt->fetchAll(PDO::FETCH_OBJ);

                if(count($rows) == 1)
                        $employeeID    = $rows[0]->empEmployeeID;

                $db->commit();
        }
        catch(PDOException $ex)
        {
                $db->rollBack();
                throw $ex;
        }

        return $employeeID;
```

```
	}


	// VehicleCategory methods


	function GetVehicleCategories()
	{
		global $db;

		$results = array();

		try
		{
			$db->beginTransaction();

			$stmt = $db->query("SELECT * FROM vehiclecategory ORDER BY vecCCFrom, vecCCTo");
			$rows = $stmt->fetchAll(PDO::FETCH_OBJ);

			for($i=0; $i<count($rows); $i++)
			{
				$result = new VehicleCategory();
				$result->ID = $rows[$i]->vecVehicleCategoryID;
				$result->CCFrom = $rows[$i]->vecCCFrom;
				$result->CCTo = $rows[$i]->vecCCTo;
				$result->IsCommercial = $rows[$i]->vecIsCommercial;
				$result->SemesterCost = $rows[$i]->vecSemesterCost;
				$results[] = $result;
			}

			$db->commit();
		}
		catch(PDOException $ex)
		{
			$db->rollBack();
			echo $ex->getMessage();
		}

		return $results;
	}

	function GetVehicleCategory($id)
	{
		global $db;

		$result = null;

		try
		{
			$db->beginTransaction();

			$stmt = $db->query("SELECT * FROM vehiclecategory WHERE vecVehicleCategoryID=".$id);
			$rows = $stmt->fetchAll(PDO::FETCH_OBJ);
```

```
                if(count($rows) == 1)
                {
                        $result = new VehicleCategory();
                        $result->ID = $rows[0]->vecVehicleCategoryID;
                        $result->CCFrom = $rows[0]->vecCCFrom;
                        $result->CCTo = $rows[0]->vecCCTo;
                        $result->IsCommercial = $rows[0]->vecIsCommercial;
                        $result->SemesterCost = $rows[0]->vecSemesterCost;
                }

                $db->commit();
        }
        catch(PDOException $ex)
        {
                $db->rollBack();
                throw $ex;
        }

        if($result == null)
                throw new NotFoundException();

        return $result;
}

function DeleteVehicleCategories($ids)
{
        global $db;

        try
        {
                $db->beginTransaction();
                $stmt = $db->query("DELETE FROM vehiclecategory WHERE
vecVehicleCategoryID IN (".implode(",", $ids).")");
                $db->commit();
        }
        catch(PDOException $ex)
        {
                $db->rollBack();
                if ($e->errorInfo[1] == PDOUNIQUECONSTRAINTERROR)
                        throw new ForeignKeyException();
        }
}

function SaveVehicleCategory($vehicleCategory)
{
        global $db;

        $entityID = $vehicleCategory->ID;

        if($vehicleCategory->CCFrom >= $vehicleCategory->CCTo)
                throw new VehicleCategoryRangeException();

        try
        {
```

```
                $db->beginTransaction();

                $sql = "SELECT vecVehicleCategoryID FROM vehiclecategory WHERE
vecVehicleCategoryID!=".$vehicleCategory->ID." AND ".

        "((vecCCFRom<=".$vehicleCategory->CCFrom." AND vecCCTo>".$vehicleCategory-
>CCFrom.") OR (vecCCFRom<=".$vehicleCategory->CCTo." AND vecCCTo>".$vehicleCategory-
>CCTo.") OR ".

"(vecCCFRom>=".$vehicleCategory->CCFrom." AND vecCCTo<".$vehicleCategory->CCTo."))";
                $stmt = $db->query($sql);
                $rows = $stmt->fetchAll(PDO::FETCH_OBJ);


                // category range violates other vehicle categories range
                if(count($rows) > 0)
                        throw new VehicleCategoryRangeException();

                // update or insert vehiclecategory
                $sql = $vehicleCategory->ID > 0 ?
                "UPDATE vehiclecategory SET".
                " vecCCFrom=".$vehicleCategory->CCFrom.",".
                " vecCCTo=".$vehicleCategory->CCTo.",".
                " vecIsCommercial=".($vehicleCategory->IsCommercial ? 1 : 0).",".
                " vecSemesterCost=".$vehicleCategory->SemesterCost.
            " WHERE vecVehicleCategoryID=".$vehicleCategory->ID :

                "INSERT INTO vehiclecategory (vecCCFrom, vecCCTo, vecIsCommercial,
vecSemesterCost) VALUES(".
                $vehicleCategory->CCFrom.",".
                $vehicleCategory->CCTo.",".
                ($vehicleCategory->IsCommercial ? 1 : 0).",".
                $vehicleCategory->SemesterCost.")";

                $db->exec($sql);

                if($entityID  <= 0)
                        $entityID = $db->lastInsertId();

                // update vehicles that belong to this category (may have to be assigned to a
new category)
                $sql = "UPDATE vehicle LEFT JOIN vehiclecategory ON
vecCCFrom<=vehCC AND vecCCTo>vehCC SET vehVehicleCategoryID=vecVehicleCategoryID;";
                $db->exec($sql);

                $db->commit();
        }
        catch(PDOException $ex)
        {
                $db->rollBack();

                if ($ex->errorInfo[1] == PDOUNIQUECONSTRAINTERROR)
                        throw new UniqueException();

                throw $ex;
```

```
		}
		catch(Exception $ex)
		{
			$db->rollBack();
			throw $ex;
		}

		return $entityID;
	}


	// Client methods


	function GetClients()
	{
		global $db;

		$results = array();

		try
		{
			$db->beginTransaction();

			$stmt = $db->query("SELECT * FROM client ORDER BY cliLastName, cliFirstName");
			$rows = $stmt->fetchAll(PDO::FETCH_OBJ);

			for($i=0; $i<count($rows); $i++)
			{
				$result = new Client();
				$result->ID = $rows[$i]->cliClientID;
				$result->FirstName = $rows[$i]->cliFirstName;
				$result->LastName = $rows[$i]->cliLastName;
				$result->Gender = intval($rows[$i]->cliGender) == 1;
				$result->Phone = $rows[$i]->cliPhone;
				$result->Age = $rows[$i]->cliAge;
				$result->LicenceDate = DateTime::createFromFormat(getMySQLDateTimeFormat(), $rows[$i]->cliLicenceDate)->format(getIsoDateFormat());
				$result->IDNumber = $rows[$i]->cliIDNumber;
				$results[] = $result;
			}

			$db->commit();
		}
		catch(PDOException $ex)
		{
			$db->rollBack();
			echo $ex->getMessage();
		}

		return $results;
	}
```

```
	function GetClient($id)
	{
		global $db;

		$result = null;

		try
		{
			$db->beginTransaction();

			$stmt = $db->query("SELECT * FROM client WHERE cliClientID=".$id);
			$rows = $stmt->fetchAll(PDO::FETCH_OBJ);

			if(count($rows) == 1)
			{
				$result = new Client();
				$result->ID = $rows[0]->cliClientID;
				$result->FirstName = $rows[0]->cliFirstName;
				$result->LastName = $rows[0]->cliLastName;
				$result->Gender = intval($rows[0]->cliGender) == 1;
				$result->Phone = $rows[0]->cliPhone;
				$result->Age = $rows[0]->cliAge;
				$result->LicenceDate = DateTime::createFromFormat(getMySQLDateTimeFormat(), $rows[0]->cliLicenceDate)->format(getIsoDateFormat());
				$result->IDNumber = $rows[0]->cliIDNumber;
			}

			$db->commit();
		}
		catch(PDOException $ex)
		{
			$db->rollBack();
			throw $ex;
		}

		if($result == null)
			throw new NotFoundException();

		return $result;
	}

	function DeleteClients($ids)
	{
		global $db;

		try
		{
			$db->beginTransaction();

			$stmt = $db->query("DELETE FROM client WHERE cliClientID IN (".implode(",", $ids).")");
			$db->commit();
		}
		catch(PDOException $ex)
```

```
		{
			$db->rollBack();
			if ($e->errorInfo[1] == PDOUNIQUECONSTRAINTERROR)
				throw new ForeignKeyException();
		}
		catch(Exception $ex)
		{
			$db->rollBack();
			throw $ex;
		}
	}

	function SaveClient($client)
	{
		global $db;

		$entityID = $client->ID;

		try
		{
			$db->beginTransaction();

			// update or insert client
			$sql = $client->ID > 0 ?
			"UPDATE client SET".
			" cliFirstName='".$client->FirstName."',".
			" cliLastName='".$client->LastName."',".
			" cliGender=".($client->Gender ? 1 : 0).",".
			" cliAge=".$client->Age.",".
			" cliPhone='".$client->Phone."',".
			"                                    cliLicenceDate='".$client->LicenceDate-
>format(getMySQLDateTimeFormat())."',".
			" cliIDNumber='".$client->IDNumber."'".
		  " WHERE cliClientID=".$client->ID :

			"INSERT INTO client (cliFirstName, cliLastName, cliGender, cliAge,
cliPhone, cliLicenceDate, cliIDNumber) VALUES(".
			"'".$client->FirstName."',".
			"'".$client->LastName."',".
			($client->Gender ? 1 : 0).",".
			$client->Age.",".
			"'".$client->Phone."',".
			"'".$client->LicenceDate->format(getMySQLDateTimeFormat())."',".
			"'".$client->IDNumber."')";

			$db->exec($sql);

			if($entityID  <= 0)
				$entityID = $db->lastInsertId();

			$db->commit();
		}
		catch(PDOException $ex)
		{
			$db->rollBack();
```

```
                if ($ex->errorInfo[1] == PDOUNIQUECONSTRAINTERROR)
                        throw new UniqueException();

                throw $ex;
        }
        catch(Exception $ex)
        {
                $db->rollBack();
                throw $ex;
        }

        return $entityID;
}

function GetClientsLookup()
{
        global $db;

        $results = array();

        try
        {
                $db->beginTransaction();

                $stmt = $db->query("SELECT cliClientID, cliFirstName, cliLastName
FROM client ORDER BY cliLastName, cliFirstName");
                $rows = $stmt->fetchAll(PDO::FETCH_OBJ);

                for($i=0; $i<count($rows); $i++)
                {
                        $result = new LookupItem();
                        $result->ID = $rows[$i]->cliClientID;
                        $result->Name = $rows[$i]->cliLastName." " .$rows[$i]-
>cliFirstName;
                        $results[] = $result;
                }

                $db->commit();
        }
        catch(PDOException $ex)
        {
                $db->rollBack();
                echo $ex->getMessage();
        }

        return $results;
}


// Vehicle methods


function GetVehicles()
{
```

```
		global $db;

		$results = array();

		try
		{
			$db->beginTransaction();

			$stmt = $db->query("SELECT * FROM vehicle JOIN client ON vehClientID=cliClientID ORDER BY vehPlate");
			$rows = $stmt->fetchAll(PDO::FETCH_OBJ);

			for($i=0; $i<count($rows); $i++)
			{
				$result = new Vehicle();
				$result->ID = $rows[$i]->vehVehicleID;
				$result->Plate = $rows[$i]->vehPlate;
				$result->RegisterDate = !is_null($rows[$i]->vehRegisterDate) ? DateTime::createFromFormat(getMySQLDateTimeFormat(), $rows[$i]->vehRegisterDate)->format(getIsoDateFormat()) : null;
				$result->CC = $rows[$i]->vehCC;
				$result->ClientID = $rows[$i]->vehClientID;
				$result->ClientName = $rows[$i]->cliLastName." ".$rows[$i]->cliFirstName;
				$results[] = $result;
			}

			$db->commit();
		}
		catch(PDOException $ex)
		{
			$db->rollBack();
			echo $ex->getMessage();
		}

		return $results;
	}

	function GetVehicle($id)
	{
		global $db;

		$result = null;

		try
		{
			$db->beginTransaction();

			$stmt = $db->query("SELECT * FROM vehicle JOIN client ON vehClientID=cliClientID WHERE vehVehicleID=".$id);
			$rows = $stmt->fetchAll(PDO::FETCH_OBJ);

			if(count($rows) == 1)
			{
				$result = new Vehicle();
```

```
                                $result->ID = $rows[0]->vehVehicleID;
                                $result->Plate = $rows[0]->vehPlate;
                                $result->RegisterDate = !is_null($rows[0]->vehRegisterDate) ? DateTime::createFromFormat(getMySQLDateTimeFormat(), $rows[0]->vehRegisterDate)->format(getIsoDateFormat()) : null;
                                $result->CC = $rows[0]->vehCC;
                                $result->ClientID = $rows[0]->vehClientID;
                                $result->ClientName = $rows[0]->cliLastName." ".$rows[0]->cliFirstName;
                        }

                        $db->commit();
                }
                catch(PDOException $ex)
                {
                        $db->rollBack();
                        throw $ex;
                }

                if($result == null)
                        throw new NotFoundException();

                return $result;
        }

        function DeleteVehicles($ids)
        {
                global $db;

                try
                {
                        $db->beginTransaction();

                        $stmt = $db->query("DELETE FROM vehicle WHERE vehVehicleID IN (".implode(",", $ids).")");
                        $db->commit();
                }
                catch(PDOException $ex)
                {
                        $db->rollBack();
                        if ($e->errorInfo[1] == PDOUNIQUECONSTRAINTERROR)
                                throw new ForeignKeyException();
                }
                catch(Exception $ex)
                {
                        $db->rollBack();
                        throw $ex;
                }
        }

        function SaveVehicle($vehicle)
        {
                global $db;

                $entityID = $vehicle->ID;
```

```
		try
		{
			$db->beginTransaction();

			// find matching vehiclecategory
			$stmt = $db->query("SELECT vecVehicleCategoryID FROM
vehiclecategory WHERE vecCCFrom<=".$vehicle->CC." AND vecCCTo>".$vehicle->CC);
			$rows = $stmt->fetchAll(PDO::FETCH_OBJ);
			if(count($rows) == 0)
				throw new VehicleCategoryRangeException();
			$vehicleCategoryID = $rows[0]->vecVehicleCategoryID;

			// update or insert vehicle
			$sql = $vehicle->ID > 0 ?
			"UPDATE vehicle SET".
			" vehPlate='".$vehicle->Plate."',".
			"	vehRegisterDate=".(!empty($vehicle->RegisterDate) ? ("'".$vehicle-
>RegisterDate->format(getMySQLDateTimeFormat())."'") : "NULL").",".
			" vehClientID=".$vehicle->ClientID.",".
			" vehCC=".$vehicle->CC.",".
			" vehVehicleCategoryID=".$vehicleCategoryID.
		  " WHERE vehVehicleID=".$vehicle->ID :

			"INSERT INTO vehicle (vehPlate, vehRegisterDate, vehClientID, vehCC,
vehVehicleCategoryID) VALUES(".
			"'".$vehicle->Plate."',".
			(!empty($vehicle->RegisterDate) ? ("'".$vehicle->RegisterDate-
>format(getMySQLDateTimeFormat())."'") : "NULL").",".
			$vehicle->ClientID.",".
			$vehicle->CC.",".
			$vehicleCategoryID.")";

			$db->exec($sql);

			if($entityID  <= 0)
				$entityID = $db->lastInsertId();

			$db->commit();
		}
		catch(PDOException $ex)
		{
			$db->rollBack();

			if ($ex->errorInfo[1] == PDOUNIQUECONSTRAINTERROR)
				throw new UniqueException();

			throw $ex;
		}
		catch(Exception $ex)
		{
			$db->rollBack();
			throw $ex;
		}
```

```
		return $entityID;
	}

	function GetVehiclesLookup($onlyRegistered)
	{
		global $db;

		$results = array();

		try
		{
			$db->beginTransaction();

			$where = $onlyRegistered ? "WHERE vehRegisterDate IS NOT NULL" : "";

			$stmt = $db->query("SELECT vehVehicleID, vehPlate, cliFirstName, cliLastName FROM vehicle JOIN client ON vehClientID=cliClientID ".$where." ORDER BY vehPlate, cliLastName, cliFirstName");
			$rows = $stmt->fetchAll(PDO::FETCH_OBJ);

			for($i=0; $i<count($rows); $i++)
			{
				$result = new LookupItem();
				$result->ID = $rows[$i]->vehVehicleID;
				$result->Name = $rows[$i]->vehPlate." [".$rows[$i]->cliLastName." " .$rows[$i]->cliFirstName."]";
				$results[] = $result;
			}

			$db->commit();
		}
		catch(PDOException $ex)
		{
			$db->rollBack();
			echo $ex->getMessage();
		}

		return $results;
	}



	// Insurance methods


	function GetInsurances($employeeID)
	{
		global $db;

		$results = array();

		try
		{
			$db->beginTransaction();
```

```
                $stmt = $db->query("SELECT * FROM insurance JOIN vehicle ON insVehicleID=vehVehicleID JOIN client ON vehClientID=cliClientID WHERE insEmployeeID=".$employeeID." ORDER BY insDate DESC");
                $rows = $stmt->fetchAll(PDO::FETCH_OBJ);

                for($i=0; $i<count($rows); $i++)
                {
                        $result = new Insurance();
                        $result->ID = $rows[$i]->insInsuranceID;
                        $result->Code = $rows[$i]->insCode;
                        $result->Date = DateTime::createFromFormat(getMySQLDateTimeFormat(), $rows[$i]->insDate)->format(getIsoDateFormat());
                        $result->Cost = $rows[$i]->insCost;
                        $result->VehicleID = $rows[$i]->insVehicleID;
                        $result->VehicleName = $rows[$i]->vehPlate." [".$rows[$i]->cliLastName." " .$rows[$i]->cliFirstName."]";
                        $results[] = $result;
                }

                $db->commit();
        }
        catch(PDOException $ex)
        {
                $db->rollBack();
                echo $ex->getMessage();
        }

        return $results;
    }

    function GetInsurance($id)
    {
        global $db;

        $result = null;

        try
        {
                $db->beginTransaction();

                $stmt = $db->query("SELECT * FROM insurance JOIN vehicle ON insVehicleID=vehVehicleID JOIN client ON vehClientID=cliClientID WHERE insInsuranceID=".$id);
                $rows = $stmt->fetchAll(PDO::FETCH_OBJ);

                if(count($rows) == 1)
                {
                        $result = new Insurance();
                        $result->ID = $rows[0]->insInsuranceID;
                        $result->Code = $rows[0]->insCode;
                        $result->Date = DateTime::createFromFormat(getMySQLDateTimeFormat(), $rows[0]->insDate)->format(getIsoDateFormat());
                        $result->Cost = $rows[0]->insCost;
```

```
                                $result->VehicleID = $rows[0]->insVehicleID;
                                $result->VehicleName = $rows[0]->vehPlate." [".$rows[0]->cliLastName." " .$rows[0]->cliFirstName."]";
                        }

                        $db->commit();
                }
                catch(PDOException $ex)
                {
                        $db->rollBack();
                        throw $ex;
                }

                if($result == null)
                        throw new NotFoundException();

                return $result;
        }

        function DeleteInsurances($ids)
        {
                global $db;

                try
                {
                        $db->beginTransaction();

                        $stmt = $db->query("DELETE FROM insurance WHERE insInsuranceID IN (".implode(",", $ids).")");
                        $db->commit();
                }
                catch(PDOException $ex)
                {
                        $db->rollBack();
                        if ($e->errorInfo[1] == PDOUNIQUECONSTRAINTERROR)
                                throw new ForeignKeyException();
                }
                catch(Exception $ex)
                {
                        $db->rollBack();
                        throw $ex;
                }
        }

        function SaveInsurance($insurance)
        {
                global $db;

                $entityID = $insurance->ID;

                try
                {
                        $db->beginTransaction();

                        // update or insert vehicle
```

```
                $sql = $insurance->ID > 0 ?
                "UPDATE insurance SET".
                " insCode='".$insurance->Code."',".
                " insDate='".$insurance->Date->format(getMySQLDateTimeFormat())."',".
                " insCost=".$insurance->Cost.",".
                " insVehicleID=".$insurance->VehicleID.",".
                " insEmployeeID=".$insurance->EmployeeID.
            " WHERE insInsuranceID=".$insurance->ID :

                "INSERT INTO insurance (insCode, insDate, insCost, insVehicleID,
insEmployeeID) VALUES(".
                "'".$insurance->Code."',".
                "'".$insurance->Date->format(getMySQLDateTimeFormat())."',".
                $insurance->Cost.",".
                $insurance->VehicleID.",".
                $insurance->EmployeeID.")";

                $db->exec($sql);

                if($entityID  <= 0)
                        $entityID = $db->lastInsertId();

                $db->commit();
        }
        catch(PDOException $ex)
        {
                $db->rollBack();

                if ($ex->errorInfo[1] == PDOUNIQUECONSTRAINTERROR)
                        throw new UniqueException();

                throw $ex;
        }
        catch(Exception $ex)
        {
                $db->rollBack();
                throw $ex;
        }

        return $entityID;
}


// Accident methods


function GetAccidents($employeeID)
{
        global $db;

        $results = array();

        try
        {
                $db->beginTransaction();
```

```
                $stmt = $db->query("SELECT * FROM accident".
            " JOIN accidentvehicle ON accAccidentID=acvAccidentID".
            " JOIN vehicle ON acvVehicleID=vehVehicleID".
            " JOIN client ON vehClientID=cliClientID".
            " WHERE accEmployeeID=".$employeeID.
        " ORDER BY accDate DESC, accAccidentID DESC, vehPlate, cliLastName, cliFirstName");
                $rows = $stmt->fetchAll(PDO::FETCH_OBJ);

                $lastAccident = null;

                for($i=0; $i<count($rows); $i++)
                {
                        if($lastAccident == null || $lastAccident->ID != $rows[$i]->accAccidentID)
                        {
                                $lastAccident = new Accident();
                                $lastAccident->ID = $rows[$i]->accAccidentID;
                                $lastAccident->Date = DateTime::createFromFormat(getMySQLDateTimeFormat(), $rows[$i]->accDate)->format(getIsoDateFormat());
                                $lastAccident->Location = $rows[$i]->accLocation;
                                $results[] = $lastAccident;
                                $lastAccident->Vehicles = array();
                        }

                        $vehicle = new AccidentVehicle();
                        $vehicle->VehicleID = $rows[$i]->acvVehicleID;
                        $vehicle->VehicleName = $rows[$i]->vehPlate." [".$rows[$i]->cliLastName." " .$rows[$i]->cliFirstName."]";
                        $vehicle->IsResponsible = intval($rows[$i]->acvIsResponsible) == 1;
                        $vehicle->DamageCost = $rows[$i]->acvDamageCost;
                        $lastAccident->Vehicles[] = $vehicle;
                }

                $db->commit();
        }
        catch(PDOException $ex)
        {
                $db->rollBack();
                echo $ex->getMessage();
        }

        return $results;
}

function GetAccident($id)
{
        global $db;

        $result = null;

        try
        {
                $db->beginTransaction();
```

```
                $stmt = $db->query("SELECT * FROM accident".
                " JOIN accidentvehicle ON accAccidentID=acvAccidentID".
                " JOIN vehicle ON acvVehicleID=vehVehicleID".
                " JOIN client ON vehClientID=cliClientID".
                " WHERE accAccidentID=".$id);
                $rows = $stmt->fetchAll(PDO::FETCH_OBJ);

                if(count($rows) > 0)
                {
                    $result = new Accident();
                    $result->ID = $rows[0]->accAccidentID;
                    $result->Date = DateTime::createFromFormat(getMySQLDateTimeFormat(), $rows[0]->accDate)->format(getIsoDateFormat());
                    $result->Location = $rows[0]->accLocation;
                    $result->Vehicles = array();

                    for($i=0; $i<count($rows); $i++)
                    {
                        $vehicle = new AccidentVehicle();
                        $vehicle->VehicleID = $rows[$i]->acvVehicleID;
                        $vehicle->VehicleName = $rows[$i]->vehPlate." [".$rows[$i]->cliLastName." " .$rows[$i]->cliFirstName."]";
                        $vehicle->IsResponsible = intval($rows[$i]->acvIsResponsible) == 1;
                        $vehicle->DamageCost = $rows[$i]->acvDamageCost;
                        $result->Vehicles[] = $vehicle;
                    }
                }

                $db->commit();
            }
            catch(PDOException $ex)
            {
                $db->rollBack();
                throw $ex;
            }

            if($result == null)
                throw new NotFoundException();

            return $result;
        }

        function DeleteAccidents($ids)
        {
            global $db;

            try
            {
                $db->beginTransaction();

                $stmt = $db->query("DELETE FROM accidentvehicle WHERE acvAccidentID IN (".implode(",", $ids).")");
```

```
                $stmt = $db->query("DELETE FROM accident WHERE accAccidentID IN
(".implode(",", $ids).")");

                $db->commit();
            }
            catch(PDOException $ex)
            {
                $db->rollBack();
                if ($e->errorInfo[1] == PDOUNIQUECONSTRAINTERROR)
                    throw new ForeignKeyException();
            }
            catch(Exception $ex)
            {
                $db->rollBack();
                throw $ex;
            }
        }

        function SaveAccident($accident)
        {
            global $db;

            $entityID = $accident->ID;

            try
            {
                $db->beginTransaction();

                // update or insert vehicle
                $sql = $accident->ID > 0 ?
                "UPDATE accident SET".
                " accDate='".$accident->Date->format(getMySQLDateTimeFormat())."',".
                " accLocation='".$accident->Location."'".
                " WHERE accAccidentID=".$accident->ID :

                "INSERT INTO accident (accDate, accLocation, accEmployeeID)
VALUES(".
                "'".$accident->Date->format(getMySQLDateTimeFormat())."',".
                "'".$accident->Location."',".
                $accident->EmployeeID.")";

                $db->exec($sql);

                if($entityID  <= 0)
                    $entityID = $db->lastInsertId();

                // delete old accident-vehicles
                $db->exec("DELETE FROM accidentvehicle WHERE
acvAccidentID=".$entityID);

                // add new accident-vehicles
                for($i=0; $i<count($accident->Vehicles); $i++)
                {
                    $sql = "INSERT INTO accidentvehicle (acvAccidentID,
acvVehicleID, acvDamageCost, acvIsResponsible) VALUES(".
```

```
                                        $entityID.",".
                                        $accident->Vehicles[$i]->VehicleID.",".
                                        $accident->Vehicles[$i]->DamageCost.",".
                                        ($accident->Vehicles[$i]->IsResponsible ? 1 : 0).")";

                                $db->exec($sql);
                        }

                        $db->commit();
                }
                catch(PDOException $ex)
                {
                        $db->rollBack();

                        if ($ex->errorInfo[1] == PDOUNIQUECONSTRAINTERROR)
                                throw new UniqueException();

                        throw $ex;
                }
                catch(Exception $ex)
                {
                        $db->rollBack();
                        throw $ex;
                }

                return $entityID;
}


// Balance methods


// $groupBy = 0 (by client)
// $groupBy = 1 (by category)
// $groupBy = 2 (by gender)
// $groupBy = 3 (by age)
function GetBalance($fromDate, $groupBy)
{
        global $db;

        $results = array();

        try
        {
                $db->beginTransaction();

                $storedProcedure = "CalculateMonthlyBalanceByClient";
                if($groupBy == 1)
                        $storedProcedure = "CalculateMonthlyBalanceByVehicleCategory";
                else if($groupBy == 2)
                        $storedProcedure = "CalculateMonthlyBalanceByGender";
                else if($groupBy == 3)
                        $storedProcedure = "CalculateMonthlyBalanceByAge";
```

```
                        $db->setAttribute(PDO::ATTR_EMULATE_PREPARES, true);
                        $stmt = $db->prepare("CALL ".$storedProcedure."(:fromDate)");
                        $stmt->execute(array(':fromDate' => $fromDate-
>format(getMySQLDateTimeFormat())));

                        $results = array();

                        while ($row = $stmt->fetch())
                        {
                                $result = $groupBy == 0 ? new BalanceClient() : ($groupBy == 1 ?
new BalanceCategory() : ($groupBy == 2 ? new BalanceGender() : new BalanceAge()));
                $result->Income = is_null($row['income']) ? null : floatval($row['income']);
                $result->Expense = is_null($row['expense']) ? null : floatval($row['expense']);

                                if($groupBy == 0)
                                        $result->Client = $row['client'];
                                else if($groupBy == 1)
                                        $result->Category = $row['category'];
                                else if($groupBy == 2)
                                        $result->Gender = $row['gender'];
                                else if($groupBy == 3)
                                        $result->Age = $row['age'];

                                $results[] = $result;
                        }

                        $db->commit();
                }
                catch(PDOException $ex)
                {
                        $db->rollBack();
                        echo $ex->getMessage();
                }

                return $results;
        }

        function GetBalanceByEmployee($fromDate, $toDate)
        {
                global $db;

                $results = array();

                try
                {
                        $db->beginTransaction();

                        $db->setAttribute(PDO::ATTR_EMULATE_PREPARES, true);
                        $stmt=$db->prepare("CALL
CalculateMonthlyRangeBalanceByEmployee(:fromDate, :toDate)");
                        $stmt-
>execute(array(':fromDate'=>$fromDate>format(getMySQLDateTimeFormat()), ':toDate' => $toDate
->format(getMySQLDateTimeFormat())));

                        $results = array();
```

```
                while ($row = $stmt->fetch())
                {
                        $result = new BalanceEmployee();
                        $result->Income=is_null($row['income'])? null :
floatval($row['income']);
                        $result->Expense = is_null($row['expense']) ? null :
floatval($row['expense']);
                        $result->Employee = $row['empName'];
                        $result->DepartmentID = $row['depID'];
                        $result->DepartmentName = $row['depName'];
                        $result->Count = $row['count'];

                        $results[] = $result;
                }

                $db->commit();
        }
        catch(PDOException $ex)
        {
                $db->rollBack();
                echo $ex->getMessage();
        }

        return $results;
}


// Accident report methods


// $groupBy = 0 (by gender)
// $groupBy = 1 (by age)
function GetAccidentReport($fromDate, $groupBy)
{
        global $db;

        $results = array();

        try
        {
                $db->beginTransaction();

                $storedProcedure = "ViewAccidentsByGender";
                if($groupBy == 1)
                        $storedProcedure = "ViewAccidentsByAge";

                $db->setAttribute(PDO::ATTR_EMULATE_PREPARES, true);
                $stmt = $db->prepare("CALL ".$storedProcedure."(:fromDate)");
                $stmt->execute(array(':fromDate' => $fromDate-
>format(getMySQLDateTimeFormat())));

                $results = array();

                while ($row = $stmt->fetch())
```

```
                        {
                                $result = $groupBy == 0 ? new AccidentReportGender() : new
AccidentReportAge();
                                $result->Count = is_null($row['count']) ? null : intval($row['count']);

                                if($groupBy == 0)
                                        $result->Gender = $row['gender'];
                                else if($groupBy == 1)
                                        $result->Age = $row['age'];

                                $results[] = $result;
                        }

                        $db->commit();
                }
                catch(PDOException $ex)
                {
                        $db->rollBack();
                        echo $ex->getMessage();
                }

                return $results;
        }
?>
```