Πτυχιακή Εργασία

Ανάπτυξη Διαδικτυακής Εφαρμογής:

Πρόγραμμα Ασφαλιστικής Εταιρείας «InterCar»

Μακρυγιαννάκη Χρυσή

Περπινάκης Πέτρος

2015

Κεφάλαιο 1° - Εισαγωγή στις Διαδικτυακές Εφαρμογές

Εισαγωγή

Με την ολοκληρωτική επικράτηση του Διαδικτύου είναι πλέον αδύνατον να σκεφτούμε τον προσωπικό ηλεκτρονικό υπολογιστή (PC) σαν αποκομμένη οντότητα. Πράγματι είναι πολύ λίγοι πλέον οι υπολογιστές που δεν είναι συνδεδεμένοι σε 24ώρη βάση στο Διαδίκτυο – και αυτοί ακόμα προορίζονται για ειδικούς σκοπούς και συνήθως για μια μοναδική χρήση (π.χ. βιομηχανικό έλεγχο).

Οι προσωπικοί υπολογιστές, τόσο οι οικιακοί όσο και αυτοί που βρίσκονται σε επαγγελματικούς χώρους, χρησιμοποιούνται πλέον κατά κόρον για πρόσβαση στις υπηρεσίες που προσφέρει το Διαδίκτυο. Οι περισσότερες κλασικές εφαρμογές έχουν μετακινηθεί από το παραδοσιακό μοντέλο στο νέο Διαδικτυακό. Ας πάρουμε ένα απλό παράδειγμα:

- Πριν την επικράτηση του Διαδικτύου, το 90% της χρήσης ενός υπολογιστή στο σπίτι ή στο γραφείο ήταν για επεξεργασία κειμένου με προγράμματα όπως το MS Word. Σε περιβάλλον γραφείου, εξέχον λόγο είχε και το MS Excel.
- Με την επικράτηση του Διαδικτύου, δεν έπαψε φυσικά να υπάρχει η ανάγκη για επεξεργασία κειμένου και λογιστικά φύλλα. Όμως τόσο το Word όσο και το Excel παρέχουν άπειρες δυνατότητες που δεν ενδιαφέρουν τον απλό χρήστη γραφείου.
- Μεγάλο μέρος των χρηστών μετακινήθηκε από τις κλασικές εφαρμογές γραφείου στις αντίστοιχες διαδικτυακές: για παράδειγμα στο Google Docs το οποίο δίνει παρόμοιες (αν και απλουστευμένες) δυνατότητες επεξεργασίας κειμένου και λογιστικών φύλλων στο περιβάλλον ενός browser.

Ταυτόχρονα οι διαδικτυακές εφαρμογές προωθούν και το μοντέλο αποθήκευσης των δεδομένων τους στο “Cloud”, ουσιαστικά μεταφέροντας την αποθήκευση από τον αυτοδιαχειριζόμενο υπολογιστή του χρήστη στους υπολογιστές της εταιρίας που παρέχει την εφαρμογή. Υπάρχουν σήμερα αρκετές υπηρεσίες Cloud, τόσο με δωρεάν όσο και με επί πληρωμή πακέτα (π.χ. Dropbox, Google Drive, MS Skydrive, Apple iCloud κλπ) και στο μέλλον είναι σίγουρο ότι θα υπάρξουν ακόμα περισσότερες. Αν και υπάρχει αρκετός κόσμος που αντιμετωπίζει με σκεπτικισμό την αποθήκευση των δεδομένων του στους servers μιας άλλης εταιρίας, το Cloud εξακολουθεί να αποτελεί μια καλή λύση

για αποθήκευση καθώς αν μη τι άλλο ο χρήστης δεν χρειάζεται πλέον να ασχολείται με το backup.

Εκτός όμως από το Cloud, οι περισσότερες εταιρίες σήμερα έχουν επιλέξει να διαθέτουν κάποιο είδος διαδικτυακής παρουσίας, με τη μορφή ιστοσελίδας η οποία μπορεί να είναι από απλά ενημερωτική έως και αρκετά ενεργή (με δυνατότητα online πώλησης προϊόντων ή υπηρεσιών) και άρα αναγκαστικά συνδεδεμένη με τα υπόλοιπα πληροφοριακά συστήματα της εταιρίας για άντληση δεδομένων.

Μια εταιρία που θέλει να αποκτήσει ή να διευρύνει τη Διαδικτυακή της παρουσία σήμερα, έχει ευτυχώς πολλές επιλογές και μάλιστα σε προνομιακές τιμές:

- Μπορεί να αναθέσει το έργο σε μια εταιρεία hosting η οποία θα αναλάβει όλες τις τεχνικές λεπτομέρειες του έργου και το στήσιμο του δικτυακού τόπου.
- Μπορεί να νοικιάσει ένα VPS (Virtual Private Server) και εφ'όσον διαθέτει δικό της IT department να στήσει και να εγκαταστήσει τη δική της διαδικτυακή εφαρμογή.
- Για ακόμα μεγαλύτερες εγκαταστάσεις, μπορεί να αγοράσει και να διαχειριστεί δικό της κανονικό server τον οποίο να έχει στο δικό της χώρο ή να τον τοποθετήσει κατόπιν συμφωνίας στις εγκαταστάσεις κάποιας εταιρίας παροχής Internet (co-location).

Σε κάθε περίπτωση, μια τέτοια υποδομή μπορεί να εξυπηρετήσει πολλαπλές ανάγκες της εταιρίας εκτός από την προφανή της δικτυακής παρουσίας της.

Οι περισσότερες εταιρίες εκτός από επεξεργασία κειμένου και λογιστικά φύλλα χρησιμοποιούν και άλλα πιο εξειδικευμένα προγράμματα που αντιστοιχούν στις εξειδικευμένες υπηρεσίες της εταιρίας. Για παράδειγμα, μια εταιρία παραγωγής αλεύρων και ζωοτροφών διαθέτει κάποιο πρόγραμμα που κρατάει τις συνταγές για κάθε ένα από τα προϊόντα, ενημερώνει για την ανάγκη αγοράς πρώτων υλών, τηρεί τα αποθέματα στις αποθήκες κλπ. Τα προγράμματα αυτού του τύπου, παραδοσιακά εκτελούνται σε ένα ή περισσότερους υπολογιστές της εταιρίας και είναι κλασικές εφαρμογές οι οποίες δεν διαθέτουν κάποια σύνδεση με το Διαδίκτυο (ή διαθέτουν κάποια πολύ στοιχειώδη σύνδεση).

Οι εφαρμογές αυτές συνήθως έχουν γραφτεί είτε κατά παραγγελία από την ίδια την εταιρία είτε αποτελούν προσαρμογή γνωστών προγραμμάτων του εμπορίου και μπορεί να χρησιμοποιούνται με μικρές αλλαγές για πολλά χρόνια.

Όμως δεν μπορούν να εκμεταλλευθούν τα πλεονεκτήματα του Διαδικτύου:

- Η εκτέλεση τους είναι περιορισμένη σε κάποια μόνο μηχανήματα της εταιρίας
- Δεν είναι εύκολη η εργασία σε αυτές από απομακρυσμένα μηχανήματα (work from home)
- Η εκτέλεση τους απαιτεί συνήθως συγκεκριμένο λειτουργικό σύστημα Windows – και σε πολλές περιπτώσεις μάλιστα ίσως απαιτεί και τη χρήση κάποιας παρωχημένης έκδοσης με προβλήματα ασφαλείας
- Όσο παλιώνουν, η ενημέρωση τους γίνεται όλο και πιο δύσκολη.

Μια εταιρία που θέλει να εκσυγχρονισθεί σίγουρα θα επιλέξει να μεταφέρει αυτές τις εφαρμογές τις σε διαδικτυακό περιβάλλον. Έτσι η εκτέλεση τους θα γίνεται μέσω ενός φυλλομετρητή (browser), με δυνατότητα να εξυπηρετήσει κοινά PC, tablets, smartphones και όποια άλλη συσκευή εμφανιστεί μελλοντικά για τη σύνδεση μας στο Διαδίκτυο. Μάλιστα η εταιρία μπορεί να επιλέξει αν η εφαρμογή αυτή θα εκτελείται μόνο στο εσωτερικό της δίκτυο (intranet) ή θα είναι ορατή και μέσω του Διαδικτύου (προφανώς με τη λήψη κατάλληλων μέτρων για την αποφυγή ανεπιθύμητης πρόσβασης).

Δομή μιας Διαδικτυακής Εφαρμογής

Μια διαδικτυακή εφαρμογή αποτελείται από τα παρακάτω κομμάτια:

- Μια **βάση δεδομένων** που αποθηκεύει με την κατάλληλη μορφή τα δεδομένα σε πίνακες. Μια σχεσιακή βάση δεδομένων επιτρέπει την γρήγορη εύρεση και συσχετισμό δεδομένων από πολλαπλούς πίνακες χωρίς να μας απασχολεί προγραμματιστικά ο ακριβής τρόπος λειτουργίας της. Χαρακτηριστικό παράδειγμα βάσης δεδομένων που χρησιμοποιείται σε αυτές τις εφαρμογές είναι η MySQL.
- Το **περιβάλλον χρήστη** που αποτελείται από μια σειρά ιστοσελίδων και φορμών σε γλώσσα HTML. Το περιβάλλον είναι ήδη οικείο στους χρήστες του Διαδικτύου μειώνοντας έτσι το χρόνο και το κόστος εκπαίδευσης.

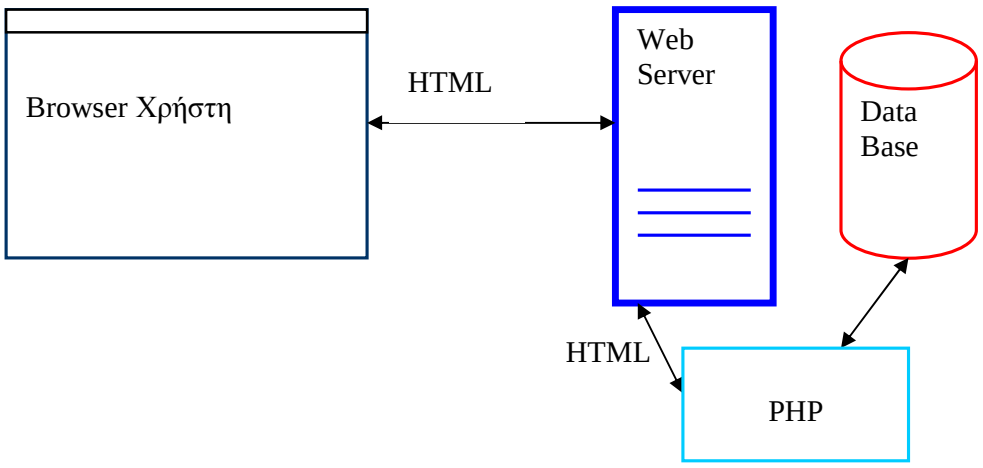
Σε σχέση με μια απλή στατική ιστοσελίδα, μια διαδικτυακή εφαρμογή αποτελείται κατά βάση από *δυναμικές ιστοσελίδες*, που δημιουργούνται τη στιγμή που εκτελείται η εφαρμογή.

Αυτό είναι φυσιολογικό, καθώς τα δεδομένα που φαίνονται στο χρήστη προέρχονται από τη βάση δεδομένων και άρα είναι δυναμικά. Για να λειτουργήσει λοιπόν μια διαδικτυακή εφαρμογή:

- Απαιτείται ένας **εξυπηρετητής διαδικτύου** (web server) με δυνατότητα δυναμικών ιστοσελίδων.
- Η εφαρμογή πρέπει να γραφεί σε κάποια γλώσσα προγραμματισμού που να υποστηρίζεται από τον παραπάνω εξυπηρετητή και όταν εκτελείται να παράγει στην πραγματικότητα τις ιστοσελίδες που φαίνονται στο χρήστη.

Κλασικά παραδείγματα γλωσσών προγραμματισμού για δυναμικές ιστοσελίδες και διασύνδεση με βάσεις δεδομένων: PHP, ASP.

Σχηματικά, θα μπορούσαμε να πούμε ότι μια διαδικτυακή εφαρμογή εκτελείται όπως στο παρακάτω σχήμα:



Γλώσσες όπως η PHP προορίζονται να εκτελούνται στον εξυπηρετητή (Server Side Scripting) και ουσιαστικά γράφουν τις HTML σελίδες που στέλνονται στο browser του χρήστη. Οι φόρμες που εμφανίζονται στο χρήστη μπορεί να απαιτούν και τη χρήση κάποιου τοπικού κώδικα για να λειτουργούν σωστά. Για την περίπτωση αυτή χρησιμοποιείται συνήθως η γλώσσα Javascript που εκτελείται απευθείας στο browser (Client Side Scripting).

Εργαλεία Ανάπτυξης

Στην προηγούμενη ενότητα αναγνωρίσαμε κάποια από τα εργαλεία ανάπτυξης μιας διαδικτυακής εφαρμογής, τα οποία θα χρησιμοποιήσουμε και στη δικιά μας:

- Τη βάση δεδομένων **MySQL**
- Τη γλώσσα προγραμματισμού **PHP**
- Τη γλώσσα δημιουργίας ιστοσελίδων **HTML** καθώς και τη γλώσσα **CSS**
- Τη γλώσσα προγραμματισμού **Javascript**
- Ένα εξυπηρετητή ιστοσελίδων όπως π.χ. ο **Apache Web Server**

Θα αναφερθούμε περιληπτικά στα παραπάνω εργαλεία.

Η Βάση Δεδομένων MySQL

Η βάση δεδομένων MySQL είναι από τις πλέον διαδεδομένες ανοικτού κώδικα βάσεις δεδομένων στον κόσμο. Είναι διαθέσιμη για πρακτικά όλα τα λειτουργικά συστήματα και συνεργάζεται άριστα με την PHP και άλλες Server Side Scripting γλώσσες. Σε συνδυασμό μάλιστα με ένα λειτουργικό σύστημα όπως το Linux και τον Apache Web Server φτιάχνεται ένα σύνολο γνωστό ως LAMP (Linux – Apache – MySQL – PHP). Θα δούμε τον τρόπο εγκατάστασης και ρύθμισης ενός τέτοιου συστήματος στο 2^{ο} κεφάλαιο, καθώς αποτελεί και το σύστημα ανάπτυξης της εφαρμογής μας.

Η MySQL είναι μια κλασική σχεσιακή βάση δεδομένων. Αυτό σημαίνει ότι η χρήση της βασίζεται στη σωστή αναγνώριση οντοτήτων (των αντικειμένων για τα οποία θέλουμε να αποθηκεύσουμε δεδομένα σε πίνακες) και συσχετίσεων (τον τρόπο με τον οποίο οι πίνακες σχετίζονται μεταξύ τους). Όπως είναι προφανές, η MySQL βασίζεται στην γλώσσα βάσεων δεδομένων SQL, αν και δεν υλοποιεί πλήρως το πρότυπο της. Μας παρέχει πάντως τα εργαλεία για να δημιουργήσουμε πίνακες, κλειδιά (primary / foreign keys), ευρετήρια (indexes) και φυσικά να γράψουμε αναζητήσεις.

Σε γενικές γραμμές η γλώσσα SQL είναι γλώσσας 4ης γενιάς, που πρακτικά σημαίνει ότι δεν χρειάζεται να ασχοληθούμε με τον τρόπο ανάκτησης ενός αποτελέσματος και αρκεί μόνο να περιγράψουμε από ποιους πίνακες θα γίνει η αναζήτηση. Παραδοσιακά η SQL μας παρέχει δύο είδη γλωσσών:

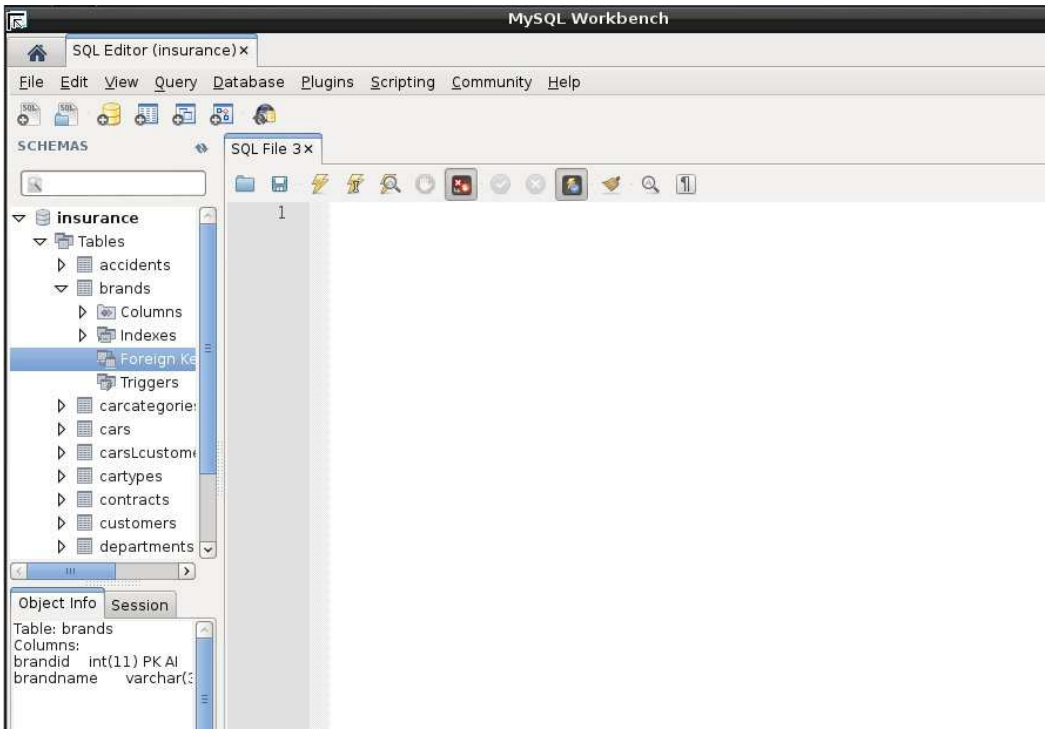
- Τη γλώσσα **DDL – Data Definition Language** για να ορίσουμε πίνακες και σχέσεις πινάκων μεταξύ τους
- Τη γλώσσα **DML – Data Manipulation Language** για να χειριστούμε δεδομένα που είναι αποθηκευμένα στους πίνακες.

Η ίδια η λέξη **SQL** σημαίνει **Structured Query Language** και αναφέρεται στην εξελιγμένη δυνατότητα αναζητήσεων δεδομένων που μας παρέχει. Μέσω της SQL μπορούμε να εκτελέσουμε απλές και σύνθετες αναζητήσεις (queries) αλλά και να προσθέσουμε ή να επεξεργαστούμε εγγραφές σε πίνακες (active queries).

Από μόνη της η MySQL δεν μας παρέχει κάποιο γραφικό εργαλείο διαχείρισης της βάσης δεδομένων. Μπορούμε να χρησιμοποιήσουμε τη γραμμή εντολών και την απλή γλώσσα DDL που παρέχεται για να δημιουργήσουμε τους πίνακες της εφαρμογής μας. Για παράδειγμα, η δημιουργία ενός απλού πίνακα με κωδικό πελάτη, επώνυμο, όνομα και διεύθυνση θα ήταν η παρακάτω:

```
CREATE TABLE `customers` (
  `customerid` int(11) NOT NULL AUTO_INCREMENT,
  `surname` varchar(32) NOT NULL,
  `name` varchar(32) NOT NULL,
  `address` varchar(32) NOT NULL,
  PRIMARY KEY (`customerid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

Το παραπάνω πρέπει να γραφεί στην γραμμή εντολών (κονσόλα) της MySQL, μια διαδικασία που θα εξηγήσουμε περισσότερο στο κεφάλαιο 2. Για τη γραφική απεικόνιση και καταχώριση μπορεί να χρησιμοποιηθεί κάποιο πρόσθετο πρόγραμμα όπως είναι το **MySQL Workbench** που φαίνεται παρακάτω:



Το MySQL Workbench μας επιτρέπει επίσης να οπτικοποιήσουμε τις σχέσεις των πινάκων ώστε να έχουμε μια καλύτερη εποπτεία της δομής της βάσης μας.

Η Γλώσσα Προγραμματισμού PHP

Η PHP είναι μια γλώσσα Server Side Scripting, ιδιαίτερα διαδεδομένη σε εφαρμογές που χρησιμοποιούν βάσεις δεδομένων (ειδικά MySQL). Η PHP (ακρωνύμιο της λέξης PHP Hypertext Processor) χρησιμοποιείται ουσιαστικά για να «γράφει» σελίδες HTML, αφού με τις εντολές τις παράγουμε δυναμικές ιστοσελίδες. Ένα μεγάλο πλεονέκτημα της PHP είναι ότι μπορούμε πολύ εύκολα να αναμίξουμε τις εντολές της μέσα σε κανονικό HTML κώδικα. Για παράδειγμα, ξεκινάμε να γράφουμε μια σελίδα ως απλή HTML και όπου χρειάζεται ενσωματώνουμε τον αντίστοιχο κώδικα PHP.

Σαν ένα γρήγορο παράδειγμα της γλώσσας **PHP**, δείτε την παρακάτω απλή σελίδα:

```
<!DOCTYPE html>
<html>
<head>
<title>PHP Test Script</title>
</head>
<body>
<p>Hello, this is mixed PHP + HTML!</p>
<?php
 echo "<p>Counting...</p>";
 for ($i=0;$i<=5;$i++) {
  echo "<p>$i</p>";
 }
?>
</body>
</html>
```

Όταν το πρόγραμμα αυτό εκτελεστεί στο **Web Server**, θα προκύψει ο παρακάτω **HTML** κώδικας:

```
<!DOCTYPE html>
<html>
<head>
<title>PHP Test Script</title>
</head>
<body>
<p>Hello, this is mixed PHP + HTML!</p>
<p>Counting...</p><p>0</p><p>1</p><p>2</p><p>3</p><p>4</p
><p>5</p></body>
</html>
```

Είναι σημαντικό να κατανοήσουμε ότι η **PHP** δεν εκτελείται στο **browser** του χρήστη αλλά στον εξυπηρετητή και παράγει την ιστοσελίδα που τελικά στέλνεται σε μορφή **HTML** στο χρήστη. Έτσι, αν στην εφαρμογή μας πρέπει να γίνει κάποια ενέργεια από τη στιγμή που έχει πλέον σταλεί η σελίδα στον **browser**, θα πρέπει να γραφεί σε μια γλώσσα **Client Side Scripting** όπως η **Javascript**.

Η γλώσσα **PHP** συνδέεται εύκολα με την **MySQL** και μπορεί να εκτελεί **queries**, να γράφει δεδομένα και να επεξεργάζεται εγγραφές. Λεπτομέρειες αυτής της υλοποίησης θα δούμε στα επόμενα κεφάλαια.

Η Γλώσσα Δημιουργίας Ιστοσελίδων HTML/CSS

Είναι πιστεύουμε περιττό να αναλωθούμε εδώ σε επεξηγήσεις για τη πασίγνωστη γλώσσα HTML. Στην τελευταία της έκδοση (HTML 5) έχουν προστεθεί διάφορες ετικέτες (tags) για να κάνουν ευκολότερη τη ζωή των Web developers.

Ένα πρόβλημα της αρχικής έκδοσης της HTML ήταν η αδυναμία διαχωρισμού της λογικής μορφοποίησης (παράγραφοι, επικεφαλίδες κλπ) από την φυσική μορφοποίηση (είδος γραμμάτων, μέγεθος γραμματοσειράς σε τίτλους και παραγράφους, χρώματα κλπ). Αυτό αντιμετωπίστηκε με τη χρήση του CSS (cascading style sheets) το οποίο έχει φτάσει στην έκδοση 3. Θα χρησιμοποιήσουμε την HTML5 και το CSS3 στην δική μας εργασία, χωρίς ωστόσο να αναφερθούμε εκτενώς σε αυτά.

Η Γλώσσα Προγραμματισμού Javascript

Για οποιεσδήποτε δυναμικές ενέργειες πρέπει να γίνουν στη σελίδα μετά που θα έχει εμφανισθεί στο browser του χρήστη, θα χρησιμοποιήσουμε τη γλώσσα Javascript. Η Javascript είναι το defacto standard στην υλοποίηση Client Side Scripting και υποστηρίζεται από όλους τους γνωστούς browsers.
Ο κύριος τρόπος με τον οποίο ενεργεί η Javascript είναι με αναγνώριση ενός συγκεκριμένου στοιχείου πάνω στη σελίδα (κάποιας ετικέτας με συγκεκριμένο id) και αλλαγή του κώδικα ή του περιεχομένου. Ένα απλό παράδειγμα φαίνεται παρακάτω:

```
<!DOCTYPE html>
<html>
<head>
  <script>
   function sayHello() {
    document.getElementById('p1').innerHTML = 'Hello!'
   }
  </script>
</head>
<body>
  <h1>Javascript Test Page</h1>
  <p id="p1">Javascript</p>
  <button type="button"
onclick="sayHello()">Hello</button>
</body>
</html>
```

Ο Apache Web Server

Για να λειτουργήσει μια διαδικτυακή εφαρμογή πρέπει φυσικά να φιλοξενηθεί σε ένα μηχάνημα το οποίο θα διαθέτει τον κατάλληλο εξυπηρετητή ιστοσελίδων. Για τη δική μας εφαρμογή έχουμε επιλέξει τον Apache Web Server:

- Είναι εξυπηρετητής ΕΛ/ΛΑΚ (ελεύθερο λογισμικό ανοικτού κώδικα)
- Εκτελείται πρακτικά σε όλα τα λειτουργικά συστήματα (τόσο σε Windows όσο και σε όλες τις παραλλαγές του Linux/UNIX)
- Η σταθερότητα και η απόδοση του έχουν αποδειχθεί από την καθημερινή του χρήση στα μεγαλύτερα sites του Internet
- Συνεργάζεται άριστα με την PHP και την MySQL επιτρέποντας μας να δημιουργήσουμε εύκολα και γρήγορα ένα περιβάλλον LAMP.

Λεπτομέρειες για την εγκατάσταση και ρύθμιση του Apache θα δούμε στο δεύτερο κεφάλαιο.

Πλεονεκτήματα Διαδικτυακών Εφαρμογών

Έχοντας παρουσιάσει τα βασικά στοιχεία και τεχνολογίες στα οποία στηρίζονται οι διαδικτυακές εφαρμογές, μπορούμε να εξετάσουμε περιληπτικά τα πλεονεκτήματα που μας παρέχουν σε σχέση με τις κλασικές εφαρμογές:

- Δυνατότητα χρήσης του προγράμματος από οπουδήποτε, με χρήση του browser σε κάθε υπολογιστή της εταιρείας ή και έξω από το εταιρικό περιβάλλον αν αυτό είναι επιθυμητό.
- **Ανεξαρτησία από λειτουργικό σύστημα:** Η εφαρμογή εκτελείται στο server και εν μέρει στο browser του χρήστη αλλά η εκτέλεση της δεν εξαρτάται από κάποιο συγκεκριμένο λειτουργικό η πλατφόρμα. Μπορεί να εκτελείται σε Windows, Linux και σε φορητές συσκευές (tablets κλπ).
- **Ευκολία μεταφοράς:** Αν οι ανάγκες της εταιρίας μεγαλώσουν η εφαρμογή μπορεί πολύ εύκολα να μετακινηθεί σε μεγαλύτερο εξυπηρετητή είτε μέσα στην εταιρία είτε σε μια άλλη εταιρία παροχής υπηρεσιών Internet.
- **Ευκολία χρήσης:** Οι χρήστες τις περισσότερες φορές βρίσκουν πιο οικείο το περιβάλλον μιας διαδικτυακής εφαρμογής καθώς έχουν συνηθίσει να χρησιμοποιούν το browser περισσότερο από κάθε άλλη εφαρμογή.

- **Ευκολία ανάπτυξης:** Τα εργαλεία που χρησιμοποιούνται είναι ανοικτού κώδικα και ελεύθερα (ΕΛ/ΛΑΚ) και υπάρχει πολύ καλή τεκμηρίωση τους στο Internet. Αυτό διευκολύνει την ανάπτυξη και τη μετέπειτα τροποποίηση του προγράμματος όταν χρειαστεί.
- **Χαμηλό κόστος:** Το πρόγραμμα μας βασίζεται σε ελεύθερο λογισμικό και δεν απαιτεί την αγορά ακριβών πακέτων ανάπτυξης ή άλλου λογισμικού για να λειτουργήσει.

Συνοψίζοντας, είναι εμφανές ότι στις μέρες μας οι διαδικτυακές εφαρμογές κερδίζουν έδαφος, και ειδικά σε εταιρικό περιβάλλον αναμένεται να επικρατήσουν καθώς παρουσιάζουν πολλά πλεονεκτήματα σε σχέση με τις κλασικές που αντικαθιστούν.

Κεφάλαιο 2° – Το Περιβάλλον Ανάπτυξης της Εφαρμογής

Εισαγωγή

Για την ανάπτυξη της δικής μας εφαρμογής «Διαδικτυακή Εφαρμογή Εταιρείας Ασφάλισης Οχημάτων» επιλέξαμε να δημιουργήσουμε ένα περιβάλλον ανάπτυξης σε μια εικονική μηχανή. Για το σκοπό αυτό χρησιμοποιήσαμε το περιβάλλον VirtualBox το οποίο διατίθεται δωρεάν. Η ανάπτυξη σε εικονική μηχανή προσφέρει πολλά πλεονεκτήματα:

- Δεν χρειάζεται να κάνουμε εγκαταστάσεις και αλλαγές ρυθμίσεων στο λειτουργικό του κανονικού μας μηχανήματος
- Μας επιτρέπει να τρέξουμε διαφορετικό λειτουργικό σύστημα σε ένα υπολογιστή, ταυτόχρονα με το κανονικά εγκατεστημένο σε αυτόν.
- Στη συγκεκριμένη εφαρμογή, θα μας επιτρέψει να δημιουργήσουμε τον απαραίτητο Web Server μαζί με τη βάση δεδομένων και την PHP που απαιτούνται για να εκτελεστεί η εφαρμογή.
- Αν και εικονικό, το μηχάνημα μπορεί να χρησιμοποιηθεί κανονικά ως εξυπηρετητής τόσο από το μηχάνημα host (αυτό το οποίο το φιλοξενεί) όσο και από άλλα μηχανήματα καθώς μπορεί να έχει – αν το επιθυμούμε – πλήρη πρόσβαση στο δίκτυο.
- Είναι απλό να μεταφέρουμε αργότερα το εικονικό μηχάνημα σε πραγματικό hardware αν το θελήσουμε.

Για το περιβάλλον ανάπτυξης επιλέξαμε να εγκαταστήσουμε το Debian Linux μαζί με τα απαραίτητα προγράμματα που το μετατρέπουν σε LAMP server (Linux – Apache – MySQL – PHP). Στο κεφάλαιο αυτό θα περιγράψουμε εν συντομία την εγκατάσταση και τις ρυθμίσεις που απαιτούνται για την εγκατάσταση και τη δοκιμή του εικονικού μηχανήματος και θα εξετάσουμε τα εργαλεία που χρησιμοποιήσαμε για τη συγγραφή του προγράμματος.

Εγκατάσταση Debian Linux στο VirtualBox

Επιλέξαμε το Debian GNU/Linux ως περιβάλλον ανάπτυξης καθώς είναι μια από τις μεγαλύτερες και καλύτερα υποστηριζόμενες διανομές Linux που υπάρχουν. Το Debian κυκλοφορεί σε κάποιες διαφορετικές εκδοχές οι οποίες διαφέρουν μεταξύ τους κατά βάση ως προς το προεπιλεγμένο γραφικό περιβάλλον που εκτελείται στο τέλος της εγκατάστασης. Για χρήση καθαρά σε περιβάλλον εξυπηρετητή,

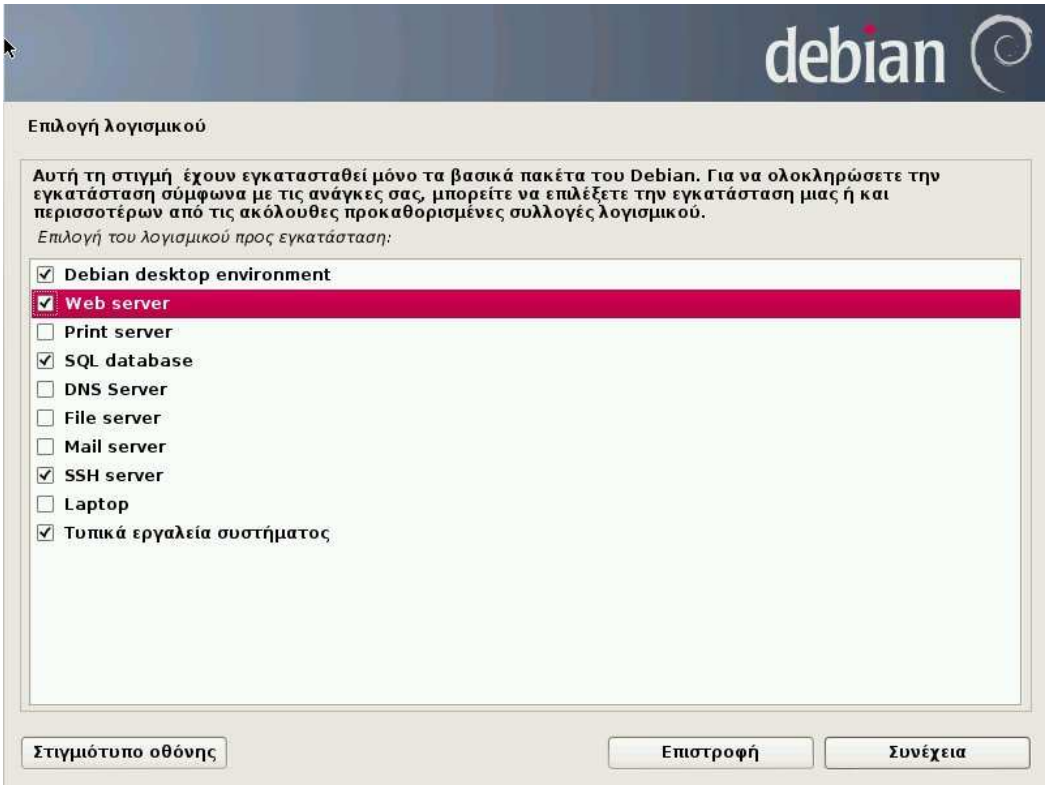
το γραφικό περιβάλλον δεν είναι καν απαραίτητο, επιλέξαμε όμως να εγκαταστήσουμε ένα από τα πλέον μικρά και ελαφριά προκειμένου να το χρησιμοποιήσουμε και κατά τη συγγραφή του προγράμματος.

Χρησιμοποιήσαμε το επίσημο **CD image** του **Debian 7.8** με γραφικό περιβάλλον **LXDE** το οποίο και κατεβάσαμε από την τοποθεσία:

http://cdimage.debian.org/debian-cd/7.8.0/i386/iso-cd/debian-7.8.0-i386-lxde-CD-1.iso

Στο **VirtualBox** δημιουργήσαμε ένα εικονικό μηχάνημα με **256Mb RAM** τα οποία επαρκούν για τη δική μας χρήση. Ρυθμίσαμε την εικονική κάρτα δικτύου σε διάταξη **bridge** ώστε το μηχάνημα μας να έχει κανονική **IP** διεύθυνση. Η βασική εγκατάσταση του λειτουργικού είναι πολύ απλή και δεν θα την περιγράψουμε αναλυτικά εδώ.

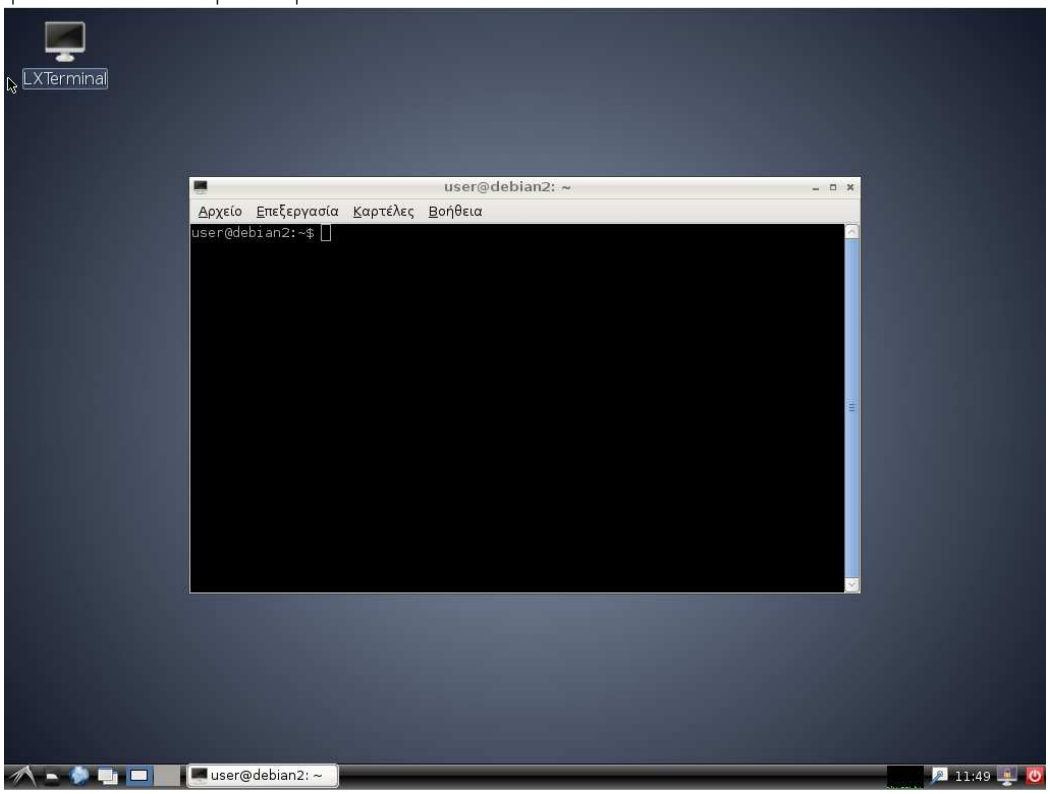
Κατά τη διάρκεια της εγκατάστασης εμφανίζεται ο παρακάτω διάλογος, όπου μεταξύ άλλων επιλέγουμε να εγκαταστήσουμε τη βάση δεδομένων **(SQL database)** και τον εξυπηρετητή ιστοσελίδων **(web server)**:



Κατά τη διάρκεια της εγκατάστασης, ορίζουμε τον κωδικό του διαχειριστή (root) σε 1234 και δημιουργούμε τον απλό χρήστη με όνομα user και κωδικό πάλι το 1234. Φυσικά οι κωδικοί αυτοί πρέπει να είναι διαφορετικοί (και ασφαλείς) σε ένα μηχάνημα παραγωγής.

Με το τέλος της εγκατάστασης θα δούμε το βασικό γραφικό περιβάλλον LXDE, και είμαστε έτοιμοι να προχωρήσουμε σε κάποιες βασικές ρυθμίσεις του συστήματος, να εγκαταστήσουμε τα επιπλέον πακέτα λογισμικού και τέλος να δοκιμάσουμε τη λειτουργία του ως εξυπηρετητή.

Το βασικό γραφικό περιβάλλον είναι αρκετά απλό και φαίνεται στην παρακάτω εικόνα:



Με το τέλος της εγκατάστασης, ρυθμίζουμε τα αποθετήρια λογισμικού αλλάζοντας το περιεχόμενο του αρχείου /etc/apt/sources.list με το παρακάτω:

```
deb http://debian.otenet.gr/debian/ wheezy main
deb-src http://debian.otenet.gr/debian/ wheezy main

deb http://security.debian.org/ wheezy/updates main
deb-src http://security.debian.org/ wheezy/updates main

deb http://debian.otenet.gr/debian/ wheezy-updates main
deb-src http://debian.otenet.gr/debian/ wheezy-updates main
```

Και μπορούμε να ανανεώσουμε το σύστημα μας εκτελώντας ως root τις παρακάτω εντολές:

```
# apt-get update
# apt-get upgrade
```

Με το τέλος της αναβάθμισης, μπορούμε να προχωρήσουμε στην εγκατάσταση επιπλέον λογισμικού και στη ρύθμιση των παραμέτρων.

Ρύθμιση της Βάσης Δεδομένων MySQL

Τη MySQL την έχουμε ήδη εγκαταστήσει, πρέπει όμως να βάλουμε κωδικό στο χρήστη root, να δημιουργήσουμε το χρήστη που θα διαχειρίζεται την βάση δεδομένων του προγράμματος μας και τέλος να δημιουργήσουμε και την ίδια τη βάση δεδομένων του προγράμματος.
Αν παραλείψαμε να εγκαταστήσουμε τη MySQL στα προηγούμενα βήματα, αυτό μπορεί να γίνει με την εντολή:

```
# apt-get install mysql-server-5.5 mysql-client-5.5
```

Αμέσως μετά ρυθμίζουμε το password του διαχειριστή της MySQL (root) σε 1234:

```
# mysqladmin –u root password 1234
```

Μπορούμε τώρα να συνδεθούμε στην MySQL από τη γραμμή εντολών και να δημιουργήσουμε τη βάση δεδομένων με όνομα **insurance** καθώς και τον χρήστη **dbuser (**με κωδικό 1234) που θα έχει δικαιώματα στη βάση αυτή:

```
# mysql –u root –p1234

mysql>create database insurance;
query OK, 1 row affected (0.00 sec)
mysql>grant all privileges on insurance.* to
‘dbuser’@’localhost’ identified by ‘1234’;
query OK, 1 row affected (0.00 sec)
mysql>flush privileges;
query OK, 1 row affected (0.00 sec)
```

Μπορούμε πλέον να χρησιμοποιούμε το χρήστη dbuser για οτιδήποτε σχετίζεται με τη βάση insurance. Εγκαθιστούμε και το πρόγραμμα MySQL workbench που θα μας βοηθήσει στην επεξεργασία της βάσης:

```
# apt-get install mysql-workbench
```

Η ρύθμιση της MySQL έχει ολοκληρωθεί.

Εγκατάσταση και Ρύθμιση PHP

Η PHP αποτελεί το συνδετικό κρίκο μεταξύ της βάσης δεδομένων MySQL και του Apache Web Server. Ο web server είναι ήδη εγκατεστημένος, κάτι που μπορούμε να επιβεβαιώσουμε ανοίγοντας το browser στο εικονικό μηχάνημα και πηγαίνοντας στη διεύθυνση http://127.0.0.1 όπου και θα δούμε τη δοκιμαστική του σελίδα. Για την εγκατάσταση της PHP θα δώσουμε:

```
# apt-get install php5 php5-mysql libapache2-mod-php5
```

Οι ρυθμίσεις του apache βρίσκονται στον κατάλογο /etc/apache2. Για να προσθέσουμε το δικό μας site, φτιάχνουμε το αρχείο **insurance** στον κατάλογο /etc/apache2/sites-available με το παρακάτω περιεχόμενο:

```
<VirtualHost *:80>
        ServerAdmin webmaster@localhost

        DocumentRoot /home/user/www
        <Directory />
                Options FollowSymLinks
                AllowOverride None
        </Directory>
        <Directory /home/user/www/>
                Options Indexes FollowSymLinks MultiViews
                AllowOverride All
                Order allow,deny
                allow from all
        </Directory>
        ErrorLog ${APACHE_LOG_DIR}/error.log
        LogLevel warn
        CustomLog ${APACHE_LOG_DIR}/access.log combined
</VirtualHost>
```

Το site της εφαρμογής μας θα βρίσκεται αποθηκευμένο στον κατάλογο /home/user/www όπου και θα αποθηκεύονται τα αρχεία php που θα δημιουργήσουμε.

Για να ενεργοποιηθεί το site, δημιουργούμε τον κατάλογο και εκτελούμε την εντολή ενεργοποίησης:

```
$ mkdir /home/user/www
$ su –
# a2ensite insurance
# a2dissite default
# service apache2 reload
```

Δοκιμή του Συστήματος Ανάπτυξης

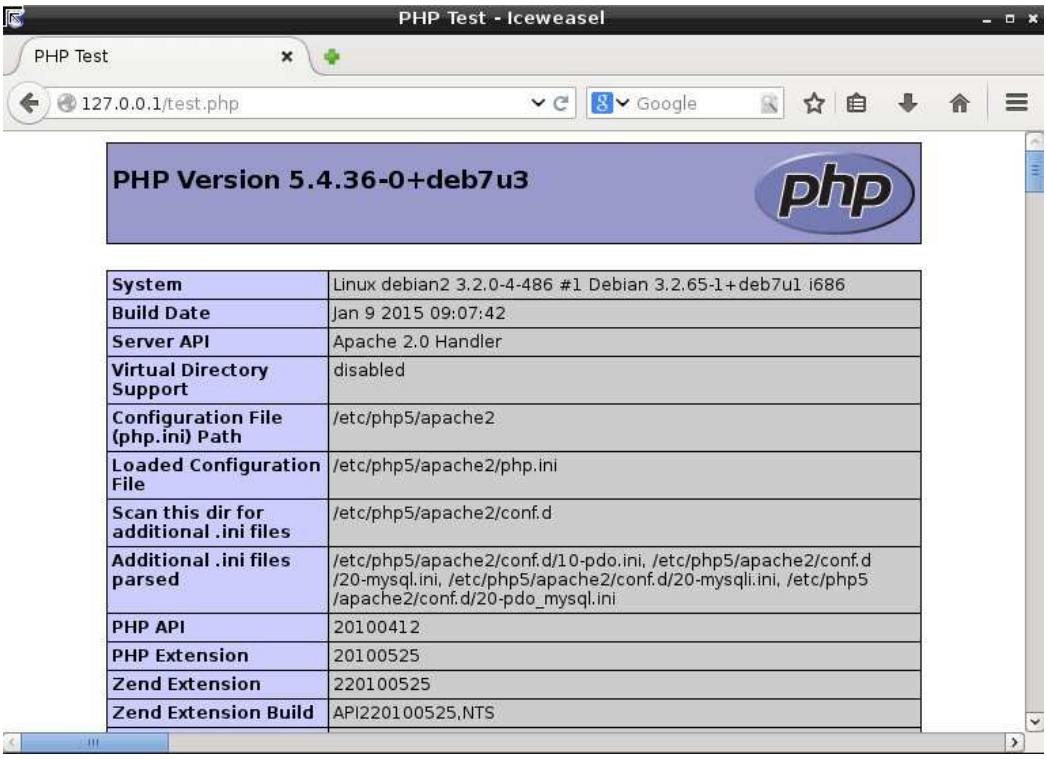
Για να δοκιμάσουμε το σύστημα ανάπτυξης – και ειδικότερα τον Web Server και την ικανότητα του να εκτελεί PHP – αποθηκεύουμε το παρακάτω απλό αρχείο με όνομα **test.php** στον κατάλογο /home/user/www:

```
<html>
  <head>
     <title>PHP Test page</title>
  </head>
  <body>
     <?php phpinfo(); ?>
  </body>
</html>
```

Για να το δοκιμάσουμε, θα πρέπει να συνδεθούμε στον Web Server και να ζητήσουμε αυτή τη σελίδα:

- Μέσα από το εικονικό μηχάνημα, μπορούμε να δούμε τη σελίδα στην τοποθεσία http://127.0.0.1/test.php
- Μέσα από το host μηχάνημα μπορούμε να δούμε τη σελίδα αφού πρώτα βρούμε την IP του εικονικού μηχανήματος. Στο εικονικό μηχάνημα εκτελούμε την εντολή /sbin/ifconfig |grep 'inet addr' και θα δούμε μια διεύθυνση (π.χ. 192.168.0.162). Αυτή τη διεύθυνση μπορούμε να τη χρησιμοποιήσουμε στο host και σε οποιοδήποτε άλλο μηχάνημα είναι στο ίδιο δίκτυο με το host.

Αν τα παραπάνω έχουν γίνει σωστά, θα δούμε την παρακάτω εικόνα:



System	Linux debian2 3.2.0-4-486 #1 Debian 3.2.65-1+deb7u1 i686
Build Date	Jan 9 2015 09:07:42
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php5/apache2
Loaded Configuration File	/etc/php5/apache2/php.ini
Scan this dir for additional .ini files	/etc/php5/apache2/conf.d
Additional .ini files parsed	/etc/php5/apache2/conf.d/10-pdo.ini, /etc/php5/apache2/conf.d /20-mysql.ini, /etc/php5/apache2/conf.d/20-mysqli.ini, /etc/php5 /apache2/conf.d/20-pdo_mysql.ini
PHP API	20100412
PHP Extension	20100525
Zend Extension	220100525
Zend Extension Build	API220100525,NTS

Με αυτές τις ρυθμίσεις είμαστε πλέον έτοιμοι για την ανάπτυξη και τη δοκιμή της εφαρμογής μας.

Κεφάλαιο 3° – Προδιαγραφές Προγράμματος και Σχεδίαση Βάσης Δεδομένων

Εισαγωγή

Στα προηγούμενα κεφάλαια αναγνωρίσαμε τις διαφορές και τα πλεονεκτήματα των διαδικτυακών εφαρμογών σε σχέση με τις κλασικές και είδαμε τα εργαλεία που χρησιμοποιούνται για την ανάπτυξη της. Είδαμε επίσης αναλυτικά τον τρόπο εγκατάστασης του περιβάλλοντος ανάπτυξης, το οποίο ουσιαστικά αποτελεί και τον εξυπηρετητή που θα εκτελεί την εφαρμογή. Είμαστε πλέον έτοιμοι να σχεδιάσουμε και να υλοποιήσουμε τη δική μας διαδικτυακή εφαρμογή.

Η εφαρμογή μας έχει τίτλο «Πρόγραμμα Ασφαλιστικής Εταιρείας Intercar» και θα καλύπτει τις ανάγκες διαχείρισης πελατών μιας τυπικής εταιρείας ασφάλισης οχημάτων. Τυπικά, τέτοια προγράμματα ασχολούνται με την αποθήκευση πληροφοριών για τους πελάτες, τα οχήματα, τα συμβόλαια τους, τα ατυχήματα κλπ. Περιέχουν διαδικασίες εύρεσης και μεταβολής δεδομένων και αυτοματισμούς όπως ανανέωση ληξιπρόθεσμων συμβολαίων και αυτόματη εξαγωγή εισόδων / εξόδων ανά μήνα έτος κλπ.

Για να προχωρήσουμε στη συγγραφή ενός τέτοιου προγράμματος, αρχικά θα δούμε τις διαδικασίες της εικονικής εταιρίας Intercar, και από αυτές θα δούμε ποια στοιχεία αποθηκεύονται (οντότητες), ποια χαρακτηριστικά τους μας ενδιαφέρουν (πεδία) και ποιες σχέσεις υπάρχουν μεταξύ τους. Το παραπάνω θα μας βοηθήσει να σχεδιάσουμε μια βάση δεδομένων η οποία θα αποτελέσει τον κορμό για τη σχεδίαση του υπόλοιπου διαδικτυακού προγράμματος.

Η Εταιρεία Ασφαλίσεως Οχημάτων Intercar

Η Intercar είναι μια τυπική, μεσαίου μεγέθους ασφαλιστική εταιρεία οχημάτων η οποία δραστηριοποιείται σε όλες τις μεγάλες πόλεις της Ελλάδας. Η εταιρεία διαθέτει εσωτερικά διάφορα τμήματα:

- Τμήμα λογιστηρίου
- Τμήμα Ασφαλίσεων
- Τμήμα Προσωπικού και Μισθοδοσίας

Τα τμήματα αυτά δεν υπάρχουν σε κάθε υποκατάστημα, αλλά μπορεί να δημιουργηθούν αν προκύπτει από τις ανάγκες και το μέγεθος ενός υποκαταστήματος. Σε κάθε τέτοιο τμήμα υπάρχει ένας Διευθυντής.

Ένας πελάτης ο οποίος επιθυμεί να ασφαλίσει το όχημα του στην Intercar θα ακολουθήσει σε γενικές γραμμές τα παρακάτω βήματα:

- Θα προσέλθει σε ένα από τα γραφεία της εταιρείας
- Θα προσκομίσει την ταυτότητα του και θα δώσει τα στοιχεία του
- Θα προσκομίσει τις άδειες κυκλοφορίας των οχημάτων του και θα παράσχει πληροφορίες που θα του ζητηθούν σχετικά, όπως π.χ.
  - Αν ήταν ασφαλισμένος πριν σε άλλη εταιρεία
  - Αν είχε ποτέ εμπλακεί σε ατύχημα και αν ναι αν ήταν με δική του υπαιτιότητα
- Θα προσκομίσει το δίπλωμα οδήγησης του

Η εταιρεία αφού συλλέξει τα στοιχεία που χρειάζεται από τα παραπάνω, θα προτείνει στον πελάτη ένα συμβόλαιο ασφάλισης, συνήθως εξάμηνο ή ετήσιο και θα τον ενημερώσει για το κόστος. Ο πελάτης μπορεί ενδεχομένως να διαπραγματευτεί το κόστος ζητώντας να προστεθούν ή να αφαιρεθούν στοιχεία κάλυψης από το συμβόλαιο του.

Το κόστος του συμβολαίου εξαρτάται μεταξύ άλλων από το είδος / μέγεθος του οχήματος, την παλαιότητα του διπλώματος οδήγησης και την μη ανάμιξη του οδηγού σε ατυχήματα με δική του υπαιτιότητα. Η συμφωνία του πελάτη με την εταιρεία οδηγεί στη σύναψη του συμβολαίου και στην είσπραξη από μεριά της εταιρείας των χρημάτων που συμφωνήθηκαν. Το ποσό αυτό εμφανίζεται στα έσοδα της εταιρείας για τον τρέχοντα μήνα.

Σε περίπτωση που ένας πελάτης της εταιρίας εμπλακεί σε ένα ατύχημα:

- Ο πελάτης επισκέπτεται τα γραφεία της εταιρείας και ενημερώνει για το ατύχημα
- Ο υπάλληλος βρίσκει τα στοιχεία του πελάτη και του οχήματος το οποίο εμπλέκεται στο ατύχημα
- Ο υπάλληλος ενημερώνεται για τα στοιχεία του άλλου εμπλεκόμενου στο ατύχημα
- Η επιτροπή εμπειρογνωμόνων της εταιρείας, σύμφωνα και με την καταγραφή της τροχαίας αποφασίζει για το αν η εταιρεία έχει την υποχρέωση να καλύψει τις ζημιές.
- Η επιτροπή εμπειρογνωμόνων αποφασίζει για το ύψος των ζημιών και δίνει την εντολή πληρωμής προς τον πελάτη ή το συνεργείο που θα προβεί στις επισκευές.
- Αν ο πελάτης είναι υπεύθυνος για τις ζημιές, τα ασφάλιστρα του θα ανέβουν ανάλογα.
- Τα χρήματα που τυχόν θα διαθέσει η εταιρεία για το ατύχημα, περνούν στα έξοδα του μήνα

Η παραπάνω είναι φυσικά μια απλουστευμένη περιγραφή των διαδικασιών της **Intercar**. Σε κάθε περίπτωση, μια ρεαλιστική πλήρη απεικόνιση των λειτουργιών, απαιτεί φυσικά τη σε βάθος μελέτη του τρόπου λειτουργίας μιας ασφαλιστικής εταιρείας. Αυτό πρακτικά σημαίνει ότι η συγγραφή ενός πλήρους προγράμματος ασφαλίσεων θα μπορούσε να γίνει μόνο με τη στενή παρακολούθηση της λειτουργίας της εταιρείας και μέσα από συνεντεύξεις με τα στελέχη και τους εργαζόμενους.

Χρησιμοποιώντας ωστόσο το παραπάνω σενάριο, θα μπορέσουμε να αναγνωρίσουμε βασικές οντότητες, πεδία και σχέσεις που υπάρχουν στην **Intercar** και να τα κωδικοποιήσουμε σε μια βάση δεδομένων **SQL**.

Πίνακες, Πεδία και Σχέσεις

Είναι σχετικά απλό να αναγνωρίσουμε βασικές οντότητες χρησιμοποιώντας το προηγούμενο σενάριο. Κάθε οντότητα αποτελεί στο πρόγραμμα μας ένα πίνακα στην βάση MySQL που θα δημιουργήσουμε. Τα δεδομένα που θα αποθηκεύσουμε για κάθε οντότητα θα αποτελέσουν πεδία του πίνακα.

Πίνακας Πελατών (Customers)

Για τον πελάτη, θέλουμε να αποθηκεύσουμε τα παρακάτω στοιχεία (πεδία):

Όνομα Πεδίου	Τύπος	Παρατηρήσεις
CustomerId (Κωδικός)	Ακέραιος	Αυτόματη Αρίθμηση, Πρωτεύον κλειδί πίνακα
Surname (επώνυμο)	Χαρακτήρες	Δεν μπορεί να είναι κενό
Name (όνομα)	Χαρακτήρες	Δεν μπορεί να είναι κενό
Phone (τηλέφωνο)	Χαρακτήρες	Μπορεί να είναι κενό
Birthyear (Ημερομηνία Γέννησης)	Ημερομηνία	Σε μορφή κατάλληλη για MySQL
Licensedate (Ημερομηνία διπλώματος)	Ημερομηνία	Σε μορφή κατάλληλη για MySQL
Cardid (Αρ. ταυτότητας)	Χαρακτήρες	

Είναι πολύ εύκολο να προστεθούν επιπλέον πεδία σε αυτόν τον πίνακα σε περίπτωση που το χρειαστούμε. Για παράδειγμα, θα μπορούσε να προστεθεί η ταχυδρομική διεύθυνση για την αποστολή ειδοποιήσεων στον πελάτη ή ακόμα και ένα **email**. Οι προσθήκες αυτές ωστόσο δεν αλλάζουν τη σχεδίαση της βάσης γιατί δεν στηρίζεται κάποια σχέση πινάκων πάνω σε αυτές.

Πίνακας Οχημάτων (Cars)

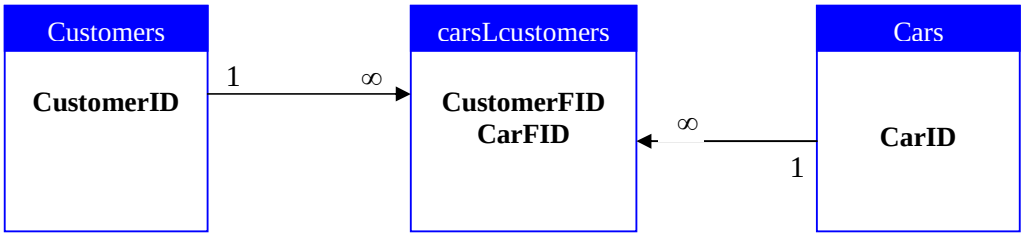
Και τα οχήματα αποτελούν ένα σχετικά απλό πίνακα. Θα πρέπει όμως να αποφύγουμε να αποθηκεύουμε πολλαπλές φορές τα ίδια δεδομένα. Για παράδειγμα, τα οχήματα παράγονται από συγκεκριμένες εταιρίες (μάρκες). Δεν είναι σκόπιμο να αποθηκεύουμε κάθε φορά το όνομα της μάρκας μέσα στον πίνακα των οχημάτων. Αυτό θα οδηγούσε σε λάθη και προβλήματα συνέπειας (π.χ. η μάρκα Ford θα μπορούσε να βρεθεί αποθηκευμένη πολλαπλές φορές ως Ford, ford, fordd) τα οποία ακριβώς λύνει το σχεσιακό μοντέλο. Για το σκοπό αυτό, ο πίνακας των οχημάτων διαθέτει πεδία **δευτερευόντων κλειδιών (foreign keys)** τα οποία αντιστοιχούν στοιχεία όπως η μάρκα και η κατηγορία του οχήματος σε άλλους πίνακες.

Παρατηρήστε ότι αν και ο πίνακας ονομάζεται **cars**, αποθηκεύει και οχήματα τα οποία μπορεί να είναι δίκυκλα (μοτοποδήλατα, μοτοσυκλέτες).

Όνομα Πεδίου	Τύπος	Παρατηρήσεις
Carid (Κωδικός)	Ακέραιος	Αυτόματη Αρίθμηση Πρωτεύον κλειδί
Brandfid (Κωδικός μάρκας)	Ακέραιος	Ακέραιος, Κλειδί από τον πίνακα brands
Modelname (Μοντέλο)	Χαρακτήρες	Δεν μπορεί να είναι κενό
Categoryfid (Κωδικός κατηγ.)	Ακέραιος	Ακέραιος, κλειδί από τον πίνακα Carcategories
Plateno (Πινακίδα κυκλοφ.)	Χαρακτήρες	Δεν μπορεί να είναι κενό
Registrationdate (Ημερ. Άδειας)	Ημερομηνία	Δεν μπορεί να είναι κενό
Owned	Flag (1bit)	Δείχνει αν το όχημα έχει καταχωρηθεί σε κάποιο πελάτη ή όχι

Σε αυτό το σημείο μπορούμε να δούμε την πρώτη σχέση Πελατών ⇔ Οχημάτων. Είναι εμφανές ότι έχουμε πολλούς πελάτες και πολλά οχήματα. Όμως ένας πελάτης μπορεί να έχει παραπάνω από ένα όχημα. Δεν είναι δυνατόν να βάλουμε ένα πεδίο οχήματος στον πελάτη (το οποίο να αποθηκεύει ένα κωδικό οχήματος) γιατί ο πελάτης μπορεί να έχει περισσότερα από ένα.

Για να φτιάξουμε τη σχέση πολλά προς πολλά που έχουμε εδώ, χρειαζόμαστε ένα ακόμα ενδιάμεσο πίνακα, τον οποίο έχουμε ονομάσει **carsLcustomers.** Η σχέση των τριών αυτών πινάκων φαίνεται παρακάτω:



Πίνακας carsLcustomers

Ο πίνακας αυτός χρησιμοποιείται μόνο για να υλοποιήσει τη σχέση πολλά προς πολλά που απαιτείται για τα οχήματα και τους πελάτες και δεν περιέχει άλλα πεδία εκτός από τους αντίστοιχους κωδικούς ως ξένα κλειδιά (foreign keys).

Όνομα Πεδίου	Τύπος	Παρατηρήσεις
CarFID (Κωδικός οχήματος)	Ακέραιος	Ξένο κλειδί από πίνακα cars
CustomerFID (Κωδικός πελάτη)	Ακέραιος	Ξένο Κλειδί από πίνακα customers.

Ο πίνακας των οχημάτων περιέχει ακόμα δύο κλειδιά που προέρχονται από τους πίνακες carcategories και brands που θα δούμε τώρα.

Πίνακας Κατασκευαστών (Brands)

Ο πίνακας αυτός περιέχει μόνο τις ονομασίες των κατασκευαστών οχημάτων για λόγους συνέπειας στην αποθήκευση. Μπορούν να προστεθούν και άλλα πεδία που αφορούν τον κατασκευαστή (π.χ. χώρα προέλευσης) αλλά αυτά δεν επηρεάζουν τη βάση καθώς δεν αποτελούν πεδία σχέσεων με άλλους πίνακες.

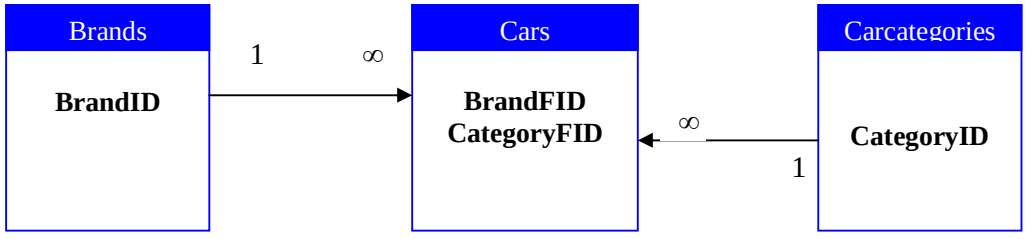
Όνομα Πεδίου	Τύπος	Παρατηρήσεις
BrandID (Κωδικός κατασκευαστή)	Ακέραιος	Αυτόματη αρίθμηση Πρωτεύον κλειδί
Brandname (Όνομα κατασκευαστή)	Χαρακτήρες	Δεν μπορεί να είναι κενό

Πίνακας Κατηγοριών Οχημάτων (Carcategories)

Ο πίνακας αυτός αντιπροσωπεύει τις κατηγορίες των οχημάτων με τις οποίες επίσης γίνεται και ο υπολογισμός των ασφαλίστρων. Περιέχει τα ελάχιστα και μέγιστα κυβικά εκατοστά για κάθε κατηγορία, μια περιγραφή και ένα κλειδί προς τον πίνακα cartypes που περιέχει γενικότερες κατηγορίες οχημάτων (π.χ. επιβατικά ΙΧ, φορτηγά, δίκυκλα κλπ).

Όνομα Πεδίου	Τύπος	Παρατηρήσεις
categoryid (Κωδικός κατηγορίας)	Ακέραιος	Αυτόματη αρίθμηση Πρωτεύον κλειδί
typefid (Κωδικός τύπου οχήματος)	Ακέραιος	Ξένο κλειδί από πίνακα cartypes
Categoryname (Όνομα κατηγορίας)	Χαρακτήρες	Δεν μπορεί να είναι κενό
Mincc (Ελάχιστα κυβικά εκατοστά)	Ακέραιος	
Maxcc (Μέγιστα κυβικά εκαταστά)	Ακέραιος	
Asfalistra (Ποσό ασφαλίστρων)	Ακέραιος	Χρησιμοποιείται ως αναφορά για τον αρχικό υπολογισμό ετήσιων ασφαλίστρων της κατηγορίας

Οι σχέσεις των πινάκων Brands και Carcategories με τον πίνακα Cars φαίνεται παρακάτω:

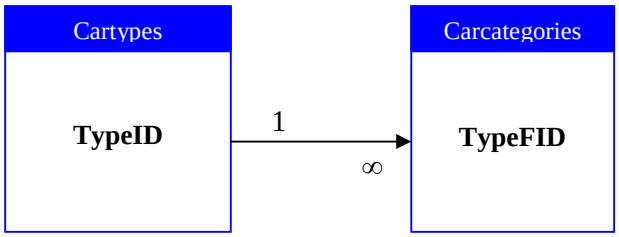


Ο πίνακας **Cartypes** είναι εξαιρετικά απλός και φαίνεται παρακάτω:

Πίνακας Τύπων Οχημάτων (Cartypes)

Όνομα Πεδίου	Τύπος	Παρατηρήσεις
TypeID (Κωδικός τύπου οχήματος)	Ακέραιος	Αυτόματη αρίθμηση Πρωτεύον κλειδί
Typename (Περιγραφή τύπου)	Χαρακτήρες	Δεν μπορεί να είναι κενό

Για παράδειγμα, ο πίνακας **cartypes** μπορεί να περιέχει τιμές όπως «Επιβατικό», «Μοτοσυκλέτα», «Φορτηγό ΔΧ», «Φορτηγό ΙΧ». Η σχέση του με τον πίνακα **Carcategories** φαίνεται παρακάτω:



Πίνακας Συμβολαίων (Contracts)

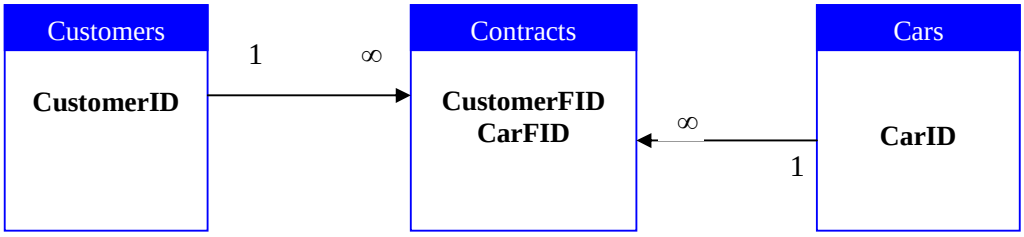
Ο πίνακας συμβολαίων περιέχει τα συμβόλαια του κάθε πελάτη με την εταιρία:

- Για κάθε όχημα του πελάτη υπάρχει ένα συμβόλαιο
- Το συμβόλαιο έχει μια ημερομηνία έναρξης και μια λήξης μετά την οποία μπορεί να ανανεωθεί
- Το κόστος του συμβολαίου προκύπτει από την κατηγορία οχήματος αλλά μπορεί να μεταβληθεί προς τα πάνω ή προς τα κάτω ανάλογα με την παλαιότητα του διπλώματος, τα ατυχήματα του πελάτη κλπ.

Όνομα Πεδίου	Τύπος	Παρατηρήσεις
ContractID (Κωδικός συμβολάιου)	Ακέραιος	Αυτόματη αρίθμηση Πρωτεύον κλειδί
CustomerFID (Κωδικός πελάτη)	Ακέραιος	Ξένο κλειδί από πίνακα πελατών
CarFID (Κωδικός οχήματος)	Ακέραιος	Ξένο κλειδί από πίνακα οχημάτων
Stardate (Ημερομηνία έναρξης)	Ημερομηνία	Δεν μπορεί να είναι κενή
Enddate (Ημερομηνία λήξης)	Ημερομηνία	Δεν μπορεί να είναι κενή
Cost (Κόστος	Πραγματικός	

συμβολαίου)		

Είναι εμφανές ότι υπάρχουν σχέσεις του πίνακα συμβολαίων με τους πίνακες πελατών και οχημάτων:



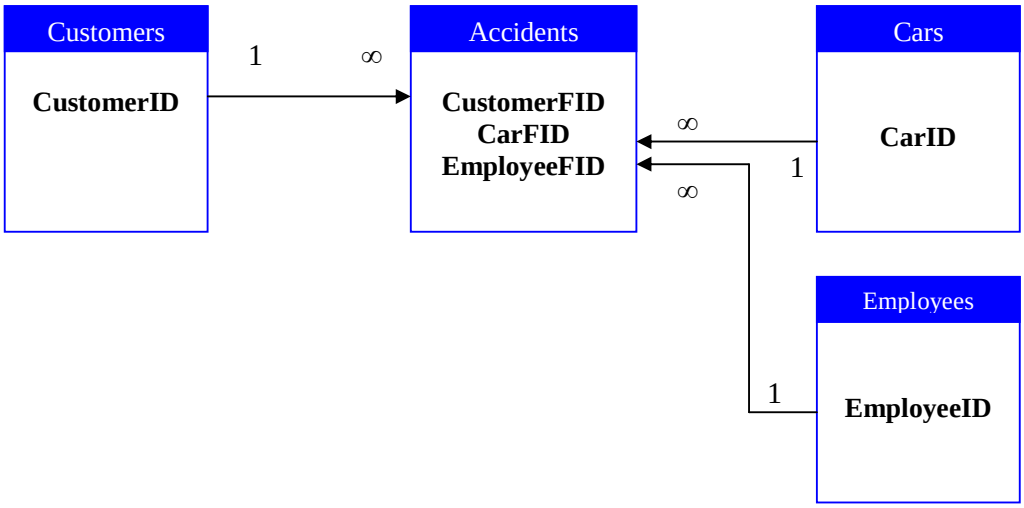
Σημειώνουμε εδώ ότι η συσχέτιση θα μπορούσε να γίνει και μόνο μέσω του οχήματος, καθώς ήδη υπάρχει σχέση οχήματος και πελάτη και όπως είναι φτιαγμένο στη βάση, ένα όχημα μπορεί να ανήκει σε ένα μόνο πελάτη.

Πίνακας Ατυχημάτων (accidents)

Ο πίνακας περιέχει λίστα των ατυχημάτων με αναφορές στον πελάτη (customerfid) και το όχημα (carfid) και πληροφορίες όπως την ημερομηνία και το ποσό που πρέπει να πληρωθεί. Ένα ατύχημα το διαχειρίζεται ένας συγκεκριμένος υπάλληλος, και υπάρχει μια σχέση προς τον αντίστοιχο πίνακα employees.

Όνομα Πεδίου	Τύπος	Παρατηρήσεις
accidentID (Κωδικός ατυχήματος)	Ακέραιος	Αυτόματη αρίθμηση Πρωτεύον κλειδί
CustomerFID (Κωδικός πελάτη)	Ακέραιος	Ξένο κλειδί από πίνακα πελατών
CarFID (Κωδικός οχήματος)	Ακέραιος	Ξένο κλειδί από πίνακα οχημάτων
employeeFID (Κωδικός υπαλλήλου)	Ακέραιος	Ξένο κλειδί από πίνακα υπαλλήλων
accidentdate (Ημερομ. ατυχήματος)	Ημερομηνία	Δεν μπορεί να είναι κενή
paymentdue (Κόστος εταιρείας)	Πραγματικός	
Notes (Σημειώσεις)	Κείμενο	Ελεύθερο πεδίο σημειώσεων

Οι σχέσεις του πίνακα φαίνονται παρακάτω:



Πίνακας Υπαλλήλων (Employees)

Ο πίνακας περιέχει τα γενικά στοιχεία των υπαλλήλων και είναι αρκετά αντίστοιχος με τον πίνακα πελατών.

Όνομα Πεδίου	Τύπος	Παρατηρήσεις
employeeID (Κωδικός υπαλλήλου)	Ακέραιος	Αυτόματη αρίθμηση Πρωτεύον κλειδί
departmentFID (Κωδικός τμήματος)	Ακέραιος	Ξένο κλειδί από πίνακα τμημάτων
employeesurname (Επώνυμο)	Χαρακτήρες	Δεν μπορεί να είναι κενό
employeename (Όνομα)	Χαρακτήρες	Δεν μπορεί να είναι κενό
employeephone (Τηλέφωνο)	Χαρακτήρες	
employeeaddress (Διεύθυνση)	Χαρακτήρες	

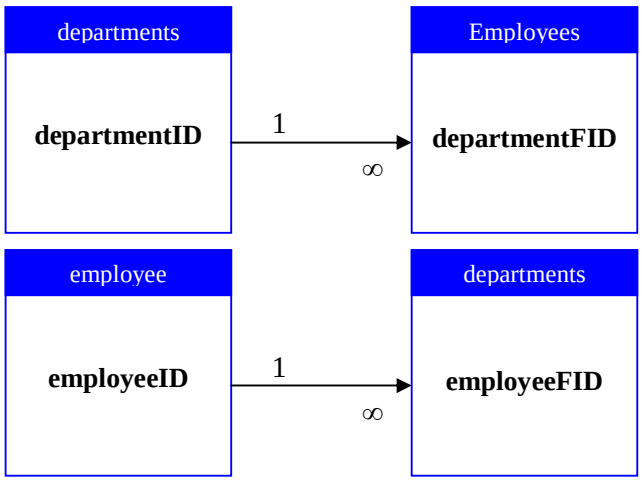
Κάθε υπάλληλος αντιστοιχίζεται σε ένα τουλάχιστον τμήμα:

Πίνακας Τμημάτων (Departments)

Στον πίνακα των τμημάτων, υπάρχει ξανά κλειδί από τον πίνακα των υπαλλήλων. Ο υπάλληλος που είναι αντιστοιχισμένος εδώ αποτελεί το διευθυντή του τμήματος. Είναι δυνατόν να γίνει καταχώριση ενός τμήματος χωρίς ύπαρξη διευθυντή και να γίνει η σχετική ανάθεση αργότερα.

Όνομα Πεδίου	Τύπος	Παρατηρήσεις
departmentID (Κωδικός τμήματος)	Ακέραιος	Αυτόματη αρίθμηση Πρωτεύον κλειδί
employeeFID (Κωδικός τμήματος)	Ακέραιος	Ξένο κλειδί από πίνακα υπαλλήλων
departmentname (Όνομα τμήματος)	Χαρακτήρες	Δεν μπορεί να είναι κενό
departmentlocation (Τοποθεσία)	Χαρακτήρες	Δεν μπορεί να είναι κενό

Με βάση τα παραπάνω, έχουμε τις σχέσεις:



Οι Πίνακες Εισόδων / Εξόδων (Income / Expenses)

Οι πίνακες αυτές χρησιμοποιούνται για την τήρηση των λογιστικών της εταιρείας:

- Κάθε φορά που η εταιρεία πρέπει να πληρώσει αποζημίωση ως αποτέλεσμα ενός ατυχήματος πελάτη της, αυτό γράφεται στον πίνακα εξόδων. Στην εγγραφή αναφέρεται η ημερομηνία πληρωμής, ο κωδικός ατυχήματος και το ποσό.
- Κάθε φορά που ένας πελάτης δημιουργεί ή ανανεώνει ένα συμβόλαιο και το πληρώνει, γίνεται αντίστοιχη εγγραφή στον πίνακα εσόδων. Στην εγγραφή αναφέρεται η ημερομηνία πληρωμής, ο κωδικός συμβολαίου και το ποσό.

Στους πίνακες αυτούς το ποσό αναγράφεται για λόγους απόδοσης κατά τους υπολογισμούς στατιστικών (έσοδα / έξοδα μήνα κλπ). Τα αντίστοιχα ποσά όμως υπάρχουν και στους πίνακες των ατυχημάτων και συμβολαίων αντίστοιχα.

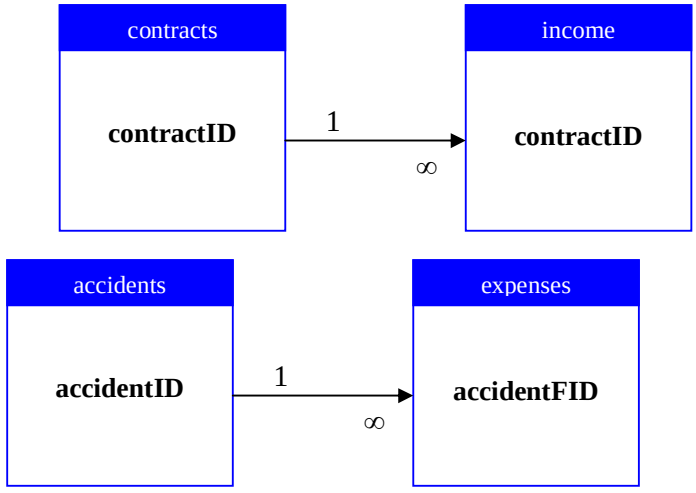
Πίνακας Εισόδων (Income)

Όνομα Πεδίου	Τύπος	Παρατηρήσεις
incomeID (Κωδικός πληρωμής)	Ακέραιος	Αυτόματη αρίθμηση Πρωτεύον κλειδί
contractFID (Κωδικός συμβολαίου)	Ακέραιος	Ξένο κλειδί από πίνακα συμβολαίων
paymentdate (Ημερομ. Πληρωμής)	Ημερομηνία	Δεν μπορεί να είναι κενό
amount (Ποσό)	Πραγματικός	

Πίνακας Εξόδων (Expenses)

Όνομα Πεδίου	Τύπος	Παρατηρήσεις
expenseID (Κωδικός πληρωμής)	Ακέραιος	Αυτόματη αρίθμηση Πρωτεύον κλειδί
accidentFID (Κωδικός ατυχήματος)	Ακέραιος	Ξένο κλειδί από πίνακα ατυχημάτων
paymentdate (Ημερομ. πληρωμής)	Ημερομηνία	Δεν μπορεί να είναι κενό
amount (Ποσό)	Πραγματικός	

Ισχύουν οι παρακάτω σχέσεις:



Σε αυτό το σημείο έχουμε ολοκληρώσει την αρχική αναγνώριση των πινάκων και των σχέσεων και είμαστε έτοιμοι να ξεκινήσουμε την υλοποίηση τους στη MySQL. Ταυτόχρονα θα γράφουμε και τον αντίστοιχο κώδικα PHP που θα δημιουργεί τις φόρμες για να περάσουμε δεδομένα στους πίνακες.

Η αρχική αυτή σχεδίαση μπορεί να μεταβληθεί κατά τη διάρκεια της υλοποίησης της αν ανακαλύψουμε ότι δεν μας καλύπτει. Συνηθισμένα προβλήματα κατά την υλοποίηση μιας βάσης δεδομένων περιλαμβάνουν την λανθασμένη ή ελλιπή αναγνώριση σχέσεων καθώς και την παρουσία παραδοχών λειτουργίας από μέρους μας που δεν ισχύουν.

Για το σκοπό αυτό είναι σημαντικό το πρόγραμμα να γράφεται και να ελέγχεται τμηματικά ώστε να είναι δυνατή η αλλαγή στη σχεδίαση του, αν απαιτηθεί, με το μικρότερο δυνατό αντίκτυπο στα κομμάτια που έχουν ήδη φτιαχθεί.

Στο επόμενο κεφάλαιο θα εξετάσουμε μια πιθανή υλοποίηση αυτής της σχεδίασης.

Κεφάλαιο 4° – Υλοποίηση της Εφαρμογής

Εισαγωγή

Σε ένα πρόγραμμα που κατά βάση υλοποιεί μια βάση δεδομένων, μπορούμε να θεωρήσουμε ότι η σωστή σχεδίαση της βάσης αποτελεί το 50% του προγράμματος. Το υπόλοιπο φυσικά αποτελείται από:

- Τα κομμάτια του κώδικα που δημιουργούν το περιβάλλον της εφαρμογής
- Τις φόρμες που θα χρησιμοποιεί ο χρήστης για να εισάγει δεδομένα στη βάση
- Τις φόρμες που υλοποιούν αναζητήσεις και σύνθετες λειτουργίες
- Τον κώδικα που υλοποιεί λειτουργίες στη βάση δεδομένων που δεν καλύπτονται από αναζητήσεις SQL αλλά απαιτούν τη χρήση μιας διαδικασιακής γλώσσας, κυρίως για την εκτέλεση υπολογισμών και την ανανέωση πινάκων που δεν έχουν μεταξύ τους συσχετίσης.

Στο κεφάλαιο αυτό θα δούμε ενδεικτικά τον τρόπο υλοποίησης κάποιων από αυτά τα τμήματα του προγράμματος. Θα ξεκινήσουμε όμως με την δημιουργία των ίδιων των πινάκων της βάσης MySQL.

Δημιουργία Πινάκων στη MySQL

Για να δημιουργήσουμε τους πίνακες στη βάση MySQL, χρησιμοποιήσαμε το ομώνυμο πρόγραμμα της γραμμής εντολών:

```
$ mysql –u dbuser –p1234 insurance
```

Ήδη στο προηγούμενο κεφάλαιο είδαμε πως με το ίδιο πρόγραμμα κατασκευάσαμε τη βάση και δώσαμε τα δικαιώματα στο χρήστη **dbuser** ο οποίος θα χρησιμοποιηθεί και από το πρόγραμμα μας σε PHP. Για να δημιουργήσουμε τους πίνακες, χρησιμοποιήσαμε την εντολή CREATE TABLE:

```
mysql> create table carsLcustomers (
       customerfid int(11) not null,
       carfid int(11) not null,
       foreign key (customerfid) references customers (customerid),
       foreign key (carfid) references cars (carid),
       primary key clc (customerfid, carfid))
       Engine=InnoDB default charset=utf8;
```

Στο παράδειγμα βλέπουμε τη δημιουργία του πίνακα carsLcustomers που συνδέει τα οχήματα με τους πελάτες. Πρόκειται για το συνδετικό πίνακα της σχέσης πολλά-προς-

πολλά. Ο πίνακας περιέχει δύο πεδία που αναφέρονται σε άλλους πίνακες (ξένα κλειδιά), το customerfid που αναφέρεται στον πίνακα customers και στο πεδίο customerid και το carfid που αναφέρεται στον πίνακα cars και στο πεδίο carid. Η σύμβαση που χρησιμοποιείται εδώ είναι όταν αν ένα πεδίο είναι ξένο κλειδί χρησιμοποιείται η παραλλαγή του κανονικού του ονόματος μαζί με το γράμμα 'f': customerid → customerfid, carid→carfid κ.ο.κ.
Ο πίνακας περιέχει ένα σύνθετο πρωτεύον κλειδί που αποτελείται από το συνδυασμό των πεδίων customerfid και carfid. Τα δύο αυτά πεδία μπορούν το καθένα να περιέχει επαναλαμβανόμενα την ίδια τιμή, ο συνδυασμός τους όμως θα είναι μοναδικός.

Ας δούμε ένα ακόμα παράδειγμα πίνακα, τον **cars:**

```
create table cars(
  carid int(11) not null auto_increment,
  brandfid int(11) not null,
  modelname varchar(32) not null,
  categoryfid int(11) not null,
  plateno varchar(12) not null,
  registrationdate date not null,
  primary key (carid),
  foreign key (brandfid) references brands (brandid),
  foreign key (categoryfid) references carcategories (categoryid))
  Engine=InnoDB default charset=utf8;
```

Μερικές ακόμα παρατηρήσεις:

- Χρησιμοποιούμε πάντοτε για κωδικοποίηση χαρακτήρων στον πίνακα το πρότυπο utf8. Έτσι δεν έχουμε κανένα πρόβλημα στην αποθήκευση ελληνικών χαρακτήρων. Το πρόγραμμα – ιστοσελίδα που θα φτιάξουμε χρησιμοποιεί επίσης utf8 από προεπιλογή.
- Η MySQL μας παρέχει διάφορες μηχανές αποθήκευσης (Engines). Έχουμε επιλέξει την InnoDB καθώς μας επιτρέπει να ορίσουμε σχέσεις μεταξύ πινάκων με τη βοήθεια των εντολών foreign key που φαίνονται παραπάνω. Η MySQL επιβάλλει τη χρήση περιορισμών που θέτουμε και δεν επιτρέπει σε πίνακες να περιέχουν ασυνεπή δεδομένα.
- Για τα πεδία που αποτελούν πρωτεύοντα κλειδιά στους περισσότερους πίνακες, χρησιμοποιούμε ως είδος πεδίου τον ακέραιο και την δυνατότητα αυτόματης αρίθμησης (auto_increment) ώστε να μη χρειάζεται να παράγουμε εμείς τιμές.

Με παρόμοιο τρόπο δημιουργήθηκαν όλοι οι πίνακες. Μπορούμε να δούμε τα ονόματα όλων με την εντολή:

```
mysql> show tables;
```

Μπορούμε να δούμε την εντολή δημιουργίας κάθε πίνακα με την εντολή:

```
mysql> show create table όνομα_πίνακα;
```

Γενική Δομή του Προγράμματος

Το πρόγραμμα υλοποιείται ως μια σειρά από ιστοσελίδες – κατά βάση φόρμες εισαγωγής δεδομένων – που αποθηκεύονται ως αρχεία PHP στον κατάλογο /home/user/www του server. Γενικά μπορούμε να διακρίνουμε τα παρακάτω είδη αρχείων PHP:

- *Ιστοσελίδες εισόδου*. Είναι ιστοσελίδες τύπου «μενού» λειτουργίων, που περιέχουν κατά βάση περιγραφή και πλήκτρα (buttons) που οδηγούν σε άλλες ιστοσελίδες. Το όνομα αρχείου τους ξεκινά με index.
- Ι*στοσελίδες προσθήκης εγγραφών στη βάση*. Πρόκειται για φόρμες που χρησιμοποιούνται για να εισάγουν δεδομένα σε κάποιο πίνακα ή σε κάποιο συνδυασμό πινάκων. Οι φόρμες αυτές εμφανίζονται στο χρήστη αρχικά κενές και πάντοτε περιέχουν το πλήκτρο «καταχώριση» με το οποίο επιβεβαιώνεται η καταχώριση δεδομένων. Το όνομα αρχείου στις περισσότερες από αυτές τις ιστοσελίδες ξεκινά με τη λέξη add.
- *Ιστοσελίδες επεξεργασίας περιεχομένου εγγραφής*. Πρόκειται για φόρμες, πολλές φορές αντίστοιχες με τις προηγούμενες αλλά εδώ τα πεδία είναι προσυμπληρωμένα με τιμές που προέρχονται από τη βάση δεδομένων και ο χρήστης καλείται να κάνει αλλαγές. Ενδέχεται κάποια από τα πεδία να είναι κλειδωμένα και να διατίθενται μόνο για ανάγνωση. Τα δεδομένα στη φόρμα έχουν προκύψει συνήθως ως αναζήτηση από το χειριστή. Το όνομα αρχείου στις περισσότερες από αυτές τις ιστοσελίδες ξεκινά με τη λέξη edit.
- *Σελίδες καταχώρισης δεδομένων*. Το όνομα αρχείου σε αυτές ξεκινά με τη λέξη update-record. Οι ιστοσελίδες αυτές δείχνουν μόνο ένα απλό μήνυμα, συνήθως «Η καταχώριση έγινε» αλλά η εκτέλεση τους είναι αυτή που προσθέτει ή αλλάζει τα δεδομένα στη βάση. Υπάρχει μια τέτοια σελίδα για κάθε φόρμα που προσθέτει / μεταβάλλει εγγραφές.
- *Σελίδες επιλογής μία-από-πολλές*. Όταν η αναζήτηση χρειάζεται περισσότερα κριτήρια, χρησιμοποιείται μια τέτοια φόρμα. Για παράδειγμα, ένας πελάτης μπορεί να έχει περισσότερα από ένα συμβόλαια. Όταν τον

επιλέξουμε, εμφανίζεται μια φόρμα που δείχνει όλα τα συμβόλαια του και μας επιτρέπει να επιλέξουμε ένα συγκεκριμένο για να το επεξεργαστούμε.

- *Αρχεία **PHP*** που δεν αποτελούν αυτόνομες ιστοσελίδες: Είναι το αρχείο functions.php που περιέχει συνδετικό κώδικα και τα αρχεία module που περιέχουν τμήματα ιστοσελίδων μενού.

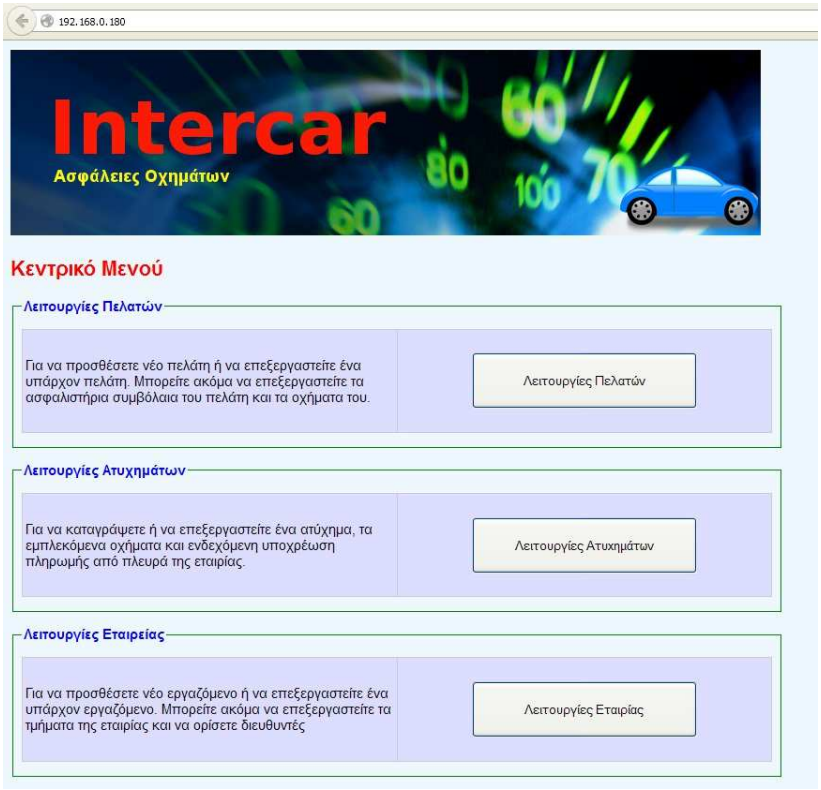
Εκτός από τα αρχεία PHP, θα βρούμε ακόμα:

- *Αρχεία **js*** με συναρτήσεις της Javascript.
- *Αρχεία **css*** για τη μορφοποίηση των ιστοσελίδων.
- *Τον κατάλογο **jquery*** που περιέχει αρχεία Javascript της βιβλιοθήκη jqueryui που χρησιμοποιούμε για το ημερολόγιο από το οποίο γίνεται η επιλογή των ημερομηνιών σε φόρμες με αντίστοιχα πεδία (και το οποίο θα δούμε παρακάτω).

Θα εξετάσουμε ενδεικτικά τον τρόπο λειτουργίας κάθε είδους από αυτές τις σελίδες.

Ιστοσελίδες Εισόδου

Οι σελίδες αυτές είναι κατά βάση οι απλούστερες του προγράμματος, αφού υλοποιούν απλώς περιγραφή των λειτουργιών και περιέχουν δεσμούς (με μορφή buttons) που μας οδηγούν απευθείας σε λειτουργίες ή σε άλλες σελίδες εισόδου. Για παράδειγμα, η κεντρική σελίδα εισόδου του προγράμματος είναι η παρακάτω:



Ένα μέρος της ιστοσελίδας, index.php είναι το παρακάτω:

```
<!DOCTYPE HTML>
<html>
  <head>
    <link rel="stylesheet" type="text/css"
href="style.css"/>
    <meta charset="utf-8">
    <title>Ασφάλειες</title>
  </head>

  <body>
    <?php include('dbinfo.inc.php'); ?>
    <header><img src="images/intercar.jpg"></header>
    <div id="container">
      <h2>Κεντρικό Μενού</h2>
      <form action="index-customers.php" method="post">
         <fieldset>
         <legend>Λειτουργίες Πελατών</legend>
         <table class="narrow"><tr><td>
         <p>Για να προσθέσετε νέο πελάτη ή να
επεξεργαστείτε ένα υπάρχον πελάτη.
 Μπορείτε ακόμα να επεξεργαστείτε τα ασφαλιστήρια
συμβόλαια του πελάτη και τα οχ
ήματα του.<p></td>
         <td class="center">
         <p><button class="fat" type="Submit"
```

```
name="action">Λειτουργίες Πελατών<
/button></p></td></tr>
         </table>
         </fieldset>
      </form>
      <form action="index-accidents.php" method="post">
         <fieldset>
            <legend>Λειτουργίες Ατυχημάτων</legend>…
```

Η ιστοσελίδα υλοποιείται σαν μια σειρά από φόρμες που ενεργοποιούν άλλες ιστοσελίδες εισόδου (**index-customers.php, index-accidents.php** κλπ). Σε κάθε φόρμα υπάρχει συμπληρωμένη η ιδιότητα **action** που δείχνει και τη φόρμα προορισμού. Το πλήκτρο είναι πάντοτε τύπου **submit** και απλά ενεργοποιεί την επόμενη φόρμα.

Αν για παράδειγμα μετακινηθούμε στις υπηρεσίες πελατών, θα δούμε την παρακάτω φόρμα (παραλείποντας το **header image**):

Εδώ η φόρμα πλέον μας παραπέμπει σε λειτουργίες του προγράμματος και όχι σε άλλη φόρμα εισόδου. Για να μπορούμε να προσθέτουμε και να αφαιρούμε λειτουργίες με ευκολία, αυτές οι φόρμες είναι σε **modular** μορφή:

```
<body>
    <?php include('dbinfo.inc.php'); ?>
    <?php include('functions.php'); ?>
    <header><img src="images/intercar.jpg"></header>
    <div id="container">
```

```
      <h2>Λειτουργίες Πελατών</h2>
      <p>Επεξεργασία στοιχείων πελατών και οχημάτων τους</p>
      <?php include('module-add-customer.php'); ?>
      <?php include('module-add-car.php'); ?>
    </div>
   <p><a href="index.php">Κεντρικό Μενού</a></p>
  </body>
```

Βλέποντας το body παρατηρούμε ότι τα κύρια κομμάτια βρίσκονται σε δύο χωριστά αρχεία:

- module-add-customer.php για τις λειτουργίες πελατών
- module-add-car.php για τις λειτουργίες οχημάτων

Ας δούμε λίγο το module-add-customer το οποίο ενεργοποιεί λειτουργίες προσθήκης αλλά και επεξεργασίας πελατών:

```
<form id="addcustomer" method="get">
        <fieldset>
        <legend>Δημιουργία / Επεξεργασία Πελάτη</legend>
        <table class="narrow"><tr><td>
        <label>Καρτέλα Πελάτη:</label></td>
        <td class="center">
         <input type="button"
                class="wider"
           onclick="submitForm('addcustomer','add-customer.php')"
           value="Νέος Πελάτης"></td></tr>
        <tr><td>
        <label>Εύρεση:</label>
        <select class="wide right" name="customerid-edit">
          <?php iterateAllCustomers(); ?>
        </select></td><td class="center">
        <input
            type="button"
            class="wider"
         onclick="submitForm('addcustomer','edit-customer.php')"
         value="Επεξεργασία"></td></tr>
```

Επικεντρώνοντας για λίγο στη φόρμα, θα δούμε ότι περιέχει buttons τα οποία ανοίγουν κάθε φορά τη σωστή φόρμα μέσω javascript. Έχουμε γράψει τη συνάρτηση submitForm που είναι η παρακάτω:

```
function submitForm(formid, action)
    {
        document.getElementById(formid).action = action;
        document.getElementById(formid).submit();
    }
```

Η συνάρτηση παίρνει δύο παραμέτρους, την φόρμα εκκίνησης και τη φόρμα που θέλουμε να ενεργοποιείται με το πλήκτρο. Ουσιαστικά, η συνάρτηση ξαναγράφει το action της φόρμας εκκίνησης ώστε το πλήκτρο να ανοίγει κάθε φορά την επιθυμητή φόρμα προορισμού.

Έτσι, όταν πιέζουμε το πλήκτρο «Νέος Πελάτης» η φόρμα addcustomer ξαναγράφεται ως:

```
<form id="addcustomer" action="add-customer.php" method="get">
```

Ενώ με το πλήκτρο «Επεξεργασία Πελάτη» η φόρμα γράφεται:

```
<form id="addcustomer" action="edit-customer.php" method="get">
```

Οι περισσότερες φόρμες του προγράμματος πρέπει να διαβάζουν δεδομένα από τη βάση. Ακόμα και οι απλές φόρμες όπως η προηγούμενη χρειάζεται να διαβάσει δεδομένα για να γεμίσει π.χ. το drop down box της εύρεσης πελατών. Πρέπει να δούμε πως γίνεται η σύνδεση της βάσης δεδομένων με την PHP και πως υλοποιούνται τα συγκεκριμένα drop down.

Σύνδεση PHP με MySQL

Η σύνδεση PHP/MySQL υλοποιείται στο παρακάτω αρχείο το οποίο συμπεριλαμβάνεται σε κάθε φόρμα που χρειάζεται να αντλήσει (ή να εγγράψει) δεδομένα από τη βάση:

Αρχείο dbinfo.inc.php

```
<?php
  $username="dbuser";
  $password="1234";
  $database="insurance";
  $handle=mysql_connect("localhost", $username, $password) or
die('Cannot connect to database!');
  mysql_select_db($database)     or     die('Cannot     select
database!');
  mysql_set_charset('utf8',$handle);
?>
```

Χρησιμοποιείται η συνάρτηση mysql_connect για να ξεκινήσει η επικοινωνία PHP/MySQL. Τα στοιχεία που δίνονται είναι:

- Το όνομα του εξυπηρετητή της βάσης δεδομένων
- Το όνομα χρήστη της βάσης, dbuser
- Ο κωδικός, 1234

Με το άνοιγμα της επικοινωνίας, επιλέγεται η βάση insurance και καθορίζεται ότι η επικοινωνία θα γίνεται σε σύνολο χαρακτήρων utf8.

Λειτουργία Drop Down Box

Τα drop down boxes εμφανίζονται σε πολλά σημεία του προγράμματος, είτε για εύρεση μιας εγγραφής και περαιτέρω

επεξεργασίας της είτε και για εισαγωγή / μεταβολή δεδομένων σε μια φόρμα. Θα δούμε πως υλοποιούνται και στις δύο περιπτώσεις.

Drop Down Box για Εύρεση Δεδομένων

Χρησιμοποιώντας ως παράδειγμα το drop down box της εύρεσης πελάτη στη φόρμα που εξετάζαμε προηγουμένως, ο κώδικας που θα δούμε είναι ο παρακάτω:

Αρχείο: module-add-customer.php

```
<label>Εύρεση:</label>
<select class="wide right" name="customerid-edit">
 <?php iterateAllCustomers(); ?>
</select>
```

Ξεκινάμε με την κανονική <select> της HTML, αλλά το τμήμα που αναλαμβάνει να γεμίσει με δεδομένα το drop down box γράφεται σε PHP. Ο κώδικας για όλα αυτά τα τμήματα βρίσκεται στο αρχείο **functions.php** και για το συγκεκριμένο select είναι:

Αρχείο: functions.php

```
function iterateAllCustomers() {
 $query="select customerid, surname, name from customers
order by surname;";
  $res=mysql_query($query);
  if ($res) {
    while ($row=mysql_fetch_array($res)) {
         echo '<option
value="'.$row['customerid'].'">'.$row['surname'].' '.$row
['name'].'</option>';

    }
  return 0; }
 else {
   return -1;
 }
```

Μπορούμε εδώ να δούμε τα βήματα με τα οποία γίνεται μια αναζήτηση στη βάση μέσω της PHP:

- Διαμορφώνουμε την αναζήτηση μας ως SQL query σε ένα string, $query.
- Στέλνουμε το query στη βάση με την εντολή mysql_query($query) που μας επιστρέφει ένα αποτέλεσμα στη μεταβλητή $res.
- Αν το αποτέλεσμα δεν είναι μηδενικό, μπορούμε να διαβάσουμε τις εγγραφές που βρέθηκαν στη βάση χρησιμοποιώντας την mysql_fetch_array.

- Αν γνωρίζουμε ότι η αναζήτηση μας θα επιφέρει περισσότερες από μια εγγραφές, καλούμε την **mysql_fetch_array** σε βρόχο. Όσο η τιμή που επιστρέφει δεν είναι μηδενική, γνωρίζουμε ότι υπάρχουν επιπλέον εγγραφές.
- Η μεταβλητή **$row** που επιστρέφει η εντολή περιέχει τα αποτελέσματα της αναζήτησης και αρκεί να ζητήσουμε το αντίστοιχο πεδίο. Π.χ. το **$row['customerid']** περιέχει τον κωδικό πελάτη.

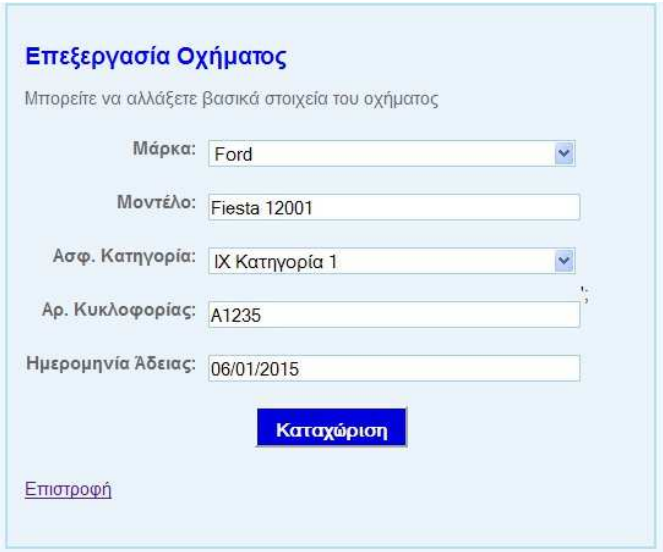
Η συνάρτηση ουσιαστικά γράφει το σώμα της ετικέτας **<select>** χρησιμοποιώντας τα δεδομένα, π.χ:

```
<option value="3">Καραγιώργος Ελευθέριος</option>
```

Σαν τιμή **(value)**, τις περισσότερες περιπτώσεις χρησιμοποιείται το αντίστοιχο κλειδί του πίνακα ενώ στο χρήστη πάντα εμφανίζονται πληροφορίες που να μπορεί να αντιληφθεί εύκολα (ονοματεπώνυμο, αριθμός πινακίδας αυτοκινήτου κλπ).

Drop Down Box για Εισαγωγή Δεδομένων

Για εισαγωγή δεδομένων, το **drop down box** δεν διαφέρει γενικά από αυτό που δείξαμε προηγουμένως για εύρεση. Αν όμως βρίσκεται σε μια φόρμα επεξεργασίας δεδομένων, ενδέχεται να πρέπει να ξεκινήσει δείχνοντας μια συγκεκριμένη τιμή (αυτή που αντιστοιχεί στην εγγραφή που φαίνεται στη φόρμα). Για παράδειγμα, ας δούμε τη φόρμα επεξεργασίας δεδομένων ενός συγκεκριμένου οχήματος:



Ανοίγοντας τη φόρμα για το όχημα «**A1235**» θα πρέπει τα δύο **drop down boxes** να δείχνουν τα σωστά στοιχεία που

αντιστοιχούν στην εγγραφή του συγκεκριμένου αυτοκινήτου. Παίρνοντας για παράδειγμα τον κώδικα που αντιστοιχεί στη μάρκα:

Αρχείο: functions.php

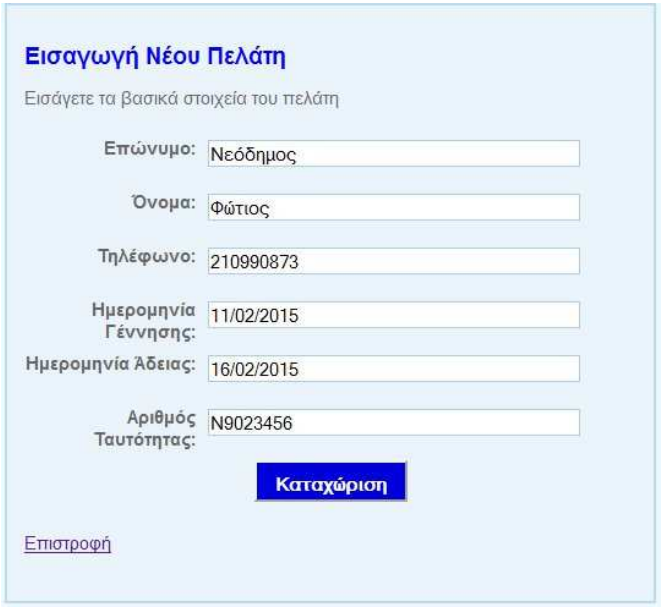
```
function iterateAllBrandsWithSelection($selection) {
  $query="select brandid, brandname from brands order by
brandname;";
  $res=mysql_query($query);
  if ($res) {
    while ($row=mysql_fetch_array($res)) {
         if ($row['brandid']==$selection) {
            echo '<option value="'.$row['brandid'].'"
selected="selected">'.$row
['brandname'].'</option>';
         } else {
            echo '<option
value="'.$row['brandid'].'">'.$row['brandname'].'</opt
ion>';
         }
    }
  }
}
```

Για κάθε τέτοια περίπτωση έχουμε μια παραλλαγή της συνάρτησης που μας δίνει τη δυνατότητα να επιλέξουμε ποια εγγραφή θα φαίνεται κατά την εκκίνηση.

- Η συνάρτηση παίρνει ως παράμετρο την τιμή του κλειδιού της εγγραφής που θέλουμε να φαίνεται ως προεπιλογή
- Καθώς δημιουργείται το drop down box ελέγχεται η τιμή του κλειδιού με αυτή που διαβάζεται από τη βάση. Αν οι τιμές είναι ίσες, προστίθεται η ιδιότητα «selected» στο συγκεκριμένο option.

Ιστοσελίδες Προσθήκης Εγγραφών στη Βάση

Θα εξετάσουμε μια σελίδα προσθήκης νέας εγγραφής στη βάση δεδομένων. Για παράδειγμα, ας πάρουμε τη σελίδα προσθήκης νέου πελάτη.



Η σελίδα περιέχει δεδομένα σε μορφή πεδίων κειμένου και ημερομηνιών. Για την εισαγωγή των ημερομηνιών θα χρησιμοποιήσουμε το **datepicker control** της βιβλιοθήκης **jqueryui. Το** βασικό κομμάτι κώδικα της φόρμας είναι:

Αρχείο: add-customer.php

```
<form action="update-record-new-customer.php" method="post">
<h1>Εισαγωγή Νέου Πελάτη</h1>
<p>Εισάγετε τα βασικά στοιχεία του πελάτη</p>
<p><label>Επώνυμο:</label> <input type="text" name="surname"></p>
<p><label>Όνομα:</label> <input type="text" name="name"></p>
<p><label>Τηλέφωνο:</label> <input type="text" name="phone"></p>
<p><label>Ημερομηνία Γέννησης:</label> <input type="text"
id="datepicker" readonly="readonly" name="birthyear"></p>
<p><label>Ημερομηνία Άδειας:</label> <input type="text"
id="datepicker2" readonly="readonly"  name="licensedate"></p>
<p><label>Αριθμός Ταυτότητας:</label> <input type="text"
name="cardid"></p>
<p><button type="Submit">Καταχώριση</button></p>
```

Τα υπόλοιπα πεδία εισόδου υλοποιούνται ως απλά **<input>.** Το πλήκτρο είναι τύπου **submit** και καλεί μια σελίδα καταχώρισης δεδομένων, **update-record-new-customer.php** που θα δούμε παρακάτω.

Το πεδίο ημερομηνίας χρησιμοποιεί το **datepicker control** με τον παρακάτω τρόπο:

Αρχείο: add-customer.php

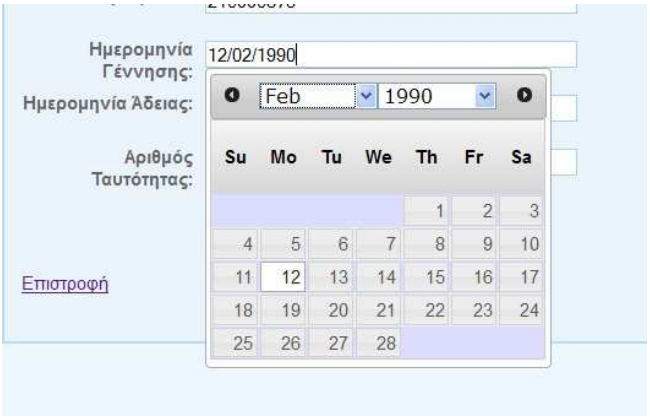
```
<script src="jquery/jquery-1.9.1.js"></script>
<script src="jquery/ui/jquery-ui.js"></script>
<script>
  $(function() {
   $("#datepicker" ).datepicker({ changeMonth: true,
changeYear: true });
```

```
   $( "#datepicker" ).datepicker( "option", "dateFormat",
"dd/mm/yy");
   $( "#datepicker2" ).datepicker({ changeMonth: true,
changeYear: true });
   $( "#datepicker2" ).datepicker( "option", "dateFormat",
"dd/mm/yy");
  });
</script>
```

Το datepicker control από τη συλλογή jqueryui, ρυθμίζεται ώστε:

- Να επιτρέπει αλλαγή τόσο μήνα όσο και έτους
- Να εμφανίζει την ημερομηνία με μορφή HH/MM/ΕΕ αντί για το συνηθισμένο αμερικάνικο σύστημα MM/HH/ΕΕ.

Τα αντίστοιχα πεδία που το χρησιμοποιούν τίθενται σε κατάσταση read only ώστε ο χρήστης να μη μπορεί να γράψει ημερομηνία χειροκίνητα. Όταν ο χρήστης επιλέξει με το ποντίκι το πεδίο εμφανίζεται το datepicker control και με την επιλογή ημερομηνίας αυτή γράφεται στο πεδίο με τη σωστή μορφή:



Για να καταχωρισθούν τα δεδομένα, θα πρέπει να εκτελεστεί η σελίδα καταχώρισης, update-record-new-customer.php

Σελίδες Καταχώρισης Δεδομένων

Ας δούμε ως παράδειγμα την σελίδα update-record-new-customer.php που αποθηκεύει τα δεδομένα πελάτη από την προηγούμενη φόρμα:

Αρχείο: update-record-new-customer.php

```
<?php
include("dbinfo.inc.php");
$surname=mysql_real_escape_string($_POST['surname']);
```

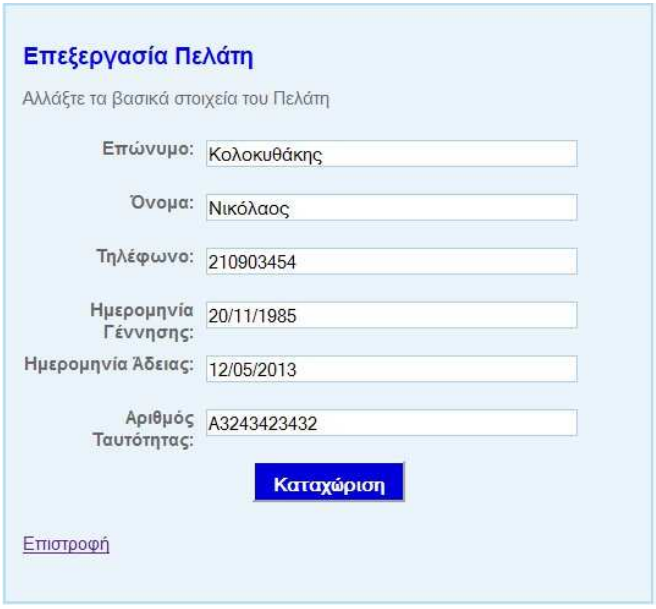
```
$name=mysql_real_escape_string($_POST['name']);
$phone=mysql_real_escape_string($_POST['phone']);
$birthyear=mysql_real_escape_string($_POST['birthyear']);
$licensedate=mysql_real_escape_string($_POST['licensedate']);
$cardid=mysql_real_escape_string($_POST['cardid']);
$query="insert into customers (surname, name, phone,
birthyear, licensedate, cardid)
values('$surname','$name','$phone',str_to_date('$birthyear','%
d/%m/%Y'),str_to_date('$licensedate','%d/%m/%Y'),'$cardid');";
if (!mysql_query($query))
 {
  die('Error: '.mysql_error());
 } else {
 echo "<p>Η καταχώριση έγινε</p>";
}
mysql_close();
?>
```

Με την πίεση του πλήκτρου **submit** στη φόρμα **add-customer.php**, τα δεδομένα μεταφέρονται μέσω της μεθόδου **POST** στη σελίδα προορισμού, **update-record-new-customer.php**.

- Χρησιμοποιούμε τη συνάρτηση **mysql_real_escape_string** για να εξασφαλίσουμε ότι τα δεδομένα είναι ασφαλή για χρήση σε **SQL query**. Η συνάρτηση καθαρίζει τα δεδομένα από προβλήματα και εγγυάται ότι τα δεδομένα δεν περιέχουν εντολές της **SQL** που θα μπορούσαν να εκτελεστούν.
- Δημιουργούμε ξανά μια εντολή **SQL query**, αλλά αυτή τη φορά για εγγραφή δεδομένων.
- Την εκτελούμε καλώντας την **mysql_query**.
- Αν η εκτέλεση είναι επιτυχής, δίνουμε το αντίστοιχο μήνυμα.
- Για να γράψουμε ημερομηνίες στη βάση, πρέπει να χρησιμοποιήσουμε τη συνάρτηση **str_to_date** για να μετατρέψουμε από τη μορφή που έχουμε δώσει στη φόρμα **(HH/MM/EE)** σε MM/HH/EE που δέχεται η **MySQL**.

Ιστοσελίδες Επεξεργασίας Εγγραφών στη Βάση

Οι ιστοσελίδες αυτές είναι αρκετά όμοιες με τις ιστοσελίδες νέων εγγραφών αλλά ξεκινάνε πάντα δείχνοντας κάποια ήδη υπάρχουσα εγγραφή. Για παράδειγμα, η φόρμα επεξεργασίας πελάτη, έχοντας επιλέξει ένα πελάτη από την αναζήτηση, δείχνει τα στοιχεία του και μας επιτρέπει να τα αλλάξουμε:



Η φόρμα ξεκινάει διαβάζοντας τα δεδομένα της σχετικής εγγραφής έχοντας πάρει το κλειδί (customerid) από τη φόρμα εύρεσης. Ανάλογα με τη φόρμα, μπορεί να έρχεται με μέθοδο POST ή με GET.

Αρχείο: edit-customer.php

```
$customerid=mysql_real_escape_string($_GET['customerid-
edit']);
  $query="select surname, name, phone,
date_format(birthyear, '%d/%m/%Y') as birthyear1,
date_format(licensedate, '%d/%m/%Y') as licensedate1,
cardid from customers where customerid='$customerid';";
  $res=mysql_query($query);
  if ($res) {
    $row=mysql_fetch_array($res);
  } else {
    echo "Error:".mysql_error(); }
```

Τα δεδομένα ημερομηνίας διαβάζονται από τη βάση απευθείας σε μορφή εμφάνισης χρησιμοποιώντας τη συνάρτηση date_format. Για τα πεδία χρησιμοποιείται PHP ώστε να εμφανιστούν απευθείας οι τιμές που διαβάσαμε, π.χ.:

```
<?php
 echo '<p><label>Επώνυμο:</label> <input type="text"
name="surname" value="'.$row['surname'].'"></p>';
 echo '<p><label>Όνομα:</label> <input type="text"
name="name" value="'.$row['name'].'"></p>'; …
```

Για τα πεδία ημερομηνιών, πρέπει οι αντίστοιχες τιμές να δοθούν στο datepicker control από πριν:

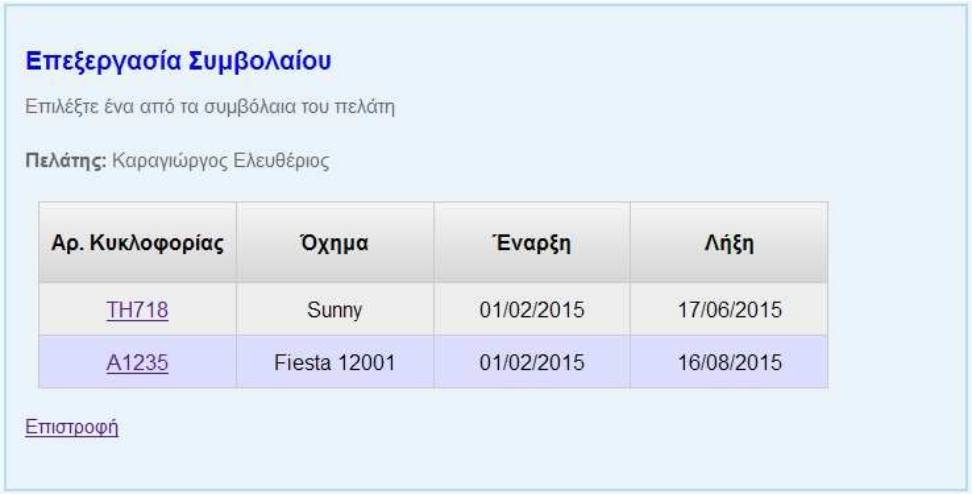
```
$(function() {
  $( "#datepicker" ).datepicker({ changeMonth: true,
changeYear: true });
  $( "#datepicker" ).datepicker( "option", "dateFormat",
"dd/mm/yy");
  $("#datepicker").datepicker( "setDate" ,"<?php echo
$row['birthyear1'];?>");
```

Αυτός είναι και ο λόγος που η ανάκτηση των δεδομένων από τη βάση γίνεται στην αρχή της σελίδας, πριν ακόμα και το αρχικό **<html> tag**.

Η ιστοσελίδα καταχώρισης εγγραφής είναι παρόμοια με αυτή που είδαμε προηγουμένως. Η μόνη πρακτική αλλαγή είναι η χρήση της εντολής **UPDATE** της **MySQL** αντί για την **INSERT** που προορίζεται για νέα εγγραφή.

Σελίδες Επιλογής Μία-Από-Πολλές

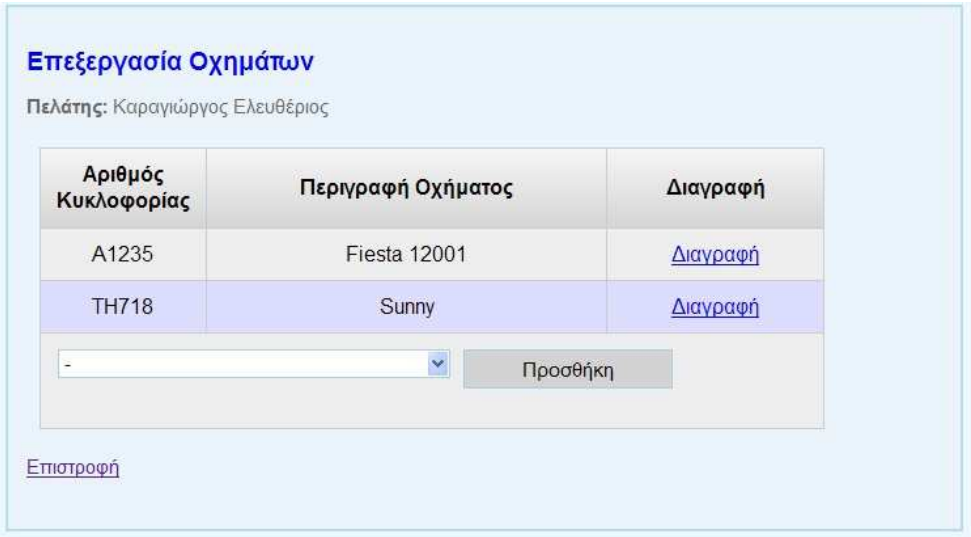
Μια τέτοια σελίδα χρησιμοποιείται για να επιλέξουμε ένα από τα πολλά συμβόλαια ενός πελάτη (αν έχει περισσότερα από ένα οχήματα):



Δεν διαφέρει ιδιαίτερα από τις προηγούμενες σελίδες, εκτός από ότι χρησιμοποιεί **links** και τη μέθοδο **GET** για να μας μεταφέρει απευθείας σε μια σελίδα επεξεργασίας του αντίστοιχου συμβολαίου.

Μια παραλλαγή είναι ακόμα η σελίδα όπου μπορούμε να αντιστοιχίσουμε οχήματα σε ένα πελάτη. Η σελίδα αυτή χρησιμοποιεί **javascript** για να προσθέτει κάθε φορά

γραμμές στον πίνακα. Η καταχώριση στη βάση δεδομένων γίνεται άμεσα με την προσθήκη της γραμμής.



Το combo box δείχνει μόνο τα οχήματα που δεν έχουν ακόμα καταχωρισθεί σε πελάτη. Για το σκοπό αυτό χρησιμοποιείται το πεδίο «owned» του πίνακα cars και το οποίο ενημερώνεται μετά από κάθε αλλαγή κατάστασης.

Κεφάλαιο 5° – Περιορισμοί του Προγράμματος – Βελτιώσεις – Συμπεράσματα

Εισαγωγή

Στο κεφάλαιο αυτό θα αναφερθούμε στα συμπεράσματα και την εμπειρία που αποκομίσαμε κατά τη διάρκεια αυτής της εργασίας και ιδιαίτερα κατά την προσπάθεια σχεδίασης και συγγραφής του κώδικα του προγράμματος. Οι δυσκολίες που συναντήσαμε οφείλονται κατά βάση:

- Στις διάφορες παραδοχές που πρέπει να γίνουν στη βάση δεδομένων, το οποίο έχει ως αποτέλεσμα να μην μπορεί να καλύψει όλα τα σενάρια χρήσης.
- Οι αλλαγές σχεδίασης που έγιναν στη βάση κατά τη διάρκεια της συγγραφής οδήγησαν πολλές φορές στον ανασχεδιασμό του υπόλοιπου κώδικα
- Η PHP αν και σχετικά εύκολη σαν γλώσσα, οδηγεί πολλές φορές σε επαναλαμβανόμενο κώδικα.
- Ο συνδυασμός HTML, Javascript και PHP σε ένα αρχείο κάνει τον κώδικα πολλές φορές σχετικά δυσανάγνωστο, ακόμα και αν η λειτουργία του είναι σχετικά απλή.

Περιορισμοί του Προγράμματος

Το πρόγραμμα στην τελική του μορφή μπορεί να βελτιωθεί με διάφορους τρόπους:

Βελτίωση της μορφής

Η σχεδίαση του προγράμματος μπορεί αισθητικά να βελτιωθεί πάρα πολύ χρησιμοποιώντας είτε κάποια έτοιμα CSS templates είτε σχεδιάζοντας από την αρχή ένα προσαρμοσμένο σύγχρονο πρότυπο σύμφωνα με τη φιλοσοφία του Web2.
Αν και τα διαδικτυακά προγράμματα όπως το συγκεκριμένο χρησιμοποιούν κατά βάση φόρμες για την είσοδο δεδομένων, είναι επίσης δυνατόν να χρησιμοποιηθούν κάποια έτοιμα frameworks για τη σχεδίαση τους, και τα οποία παρέχουν καλύτερη λειτουργικότητα και απαιτούν λιγότερο κώδικα από μέρους μας.

Στο πρόγραμμα μας χρησιμοποιήσαμε το datepicker control από τη σειρά JQueryUI. Θεωρήσαμε τη χρήση του επιβεβλημένη, καθώς η HTML δεν παρέχει κάποιο έτοιμο widget για την εισαγωγή ημερομηνιών σε φόρμες. Η βάση

δεδομένων MySQL αναμένει πάντοτε τις ημερομηνίες με συγκεκριμένη μορφή και η χρήση του datepicker εξασφαλίζει συνέπεια στην είσοδο.

Ωστόσο δεν συμβαίνει το ίδιο και με άλλα πεδία στις φόρμες, όπου σημαντική βελτίωση θα ήταν να χρησιμοποιηθούν και άλλα widgets από το JQueryUI ή αντίστοιχο framework. Μεταξύ άλλων, τα συγκεκριμένα widgets προσφέρουν και επαλήθευση τύπου δεδομένων κάτι που δεν καλύπτεται από τα τυποποιημένα HTML controls. Συνολικά, μπορούμε να θεωρήσουμε ότι ένα ολοκληρωμένο τέτοιο πρόγραμμα για εμπορική χρήση θα πρέπει να έχει επαλήθευση όλων των δυνατών εισόδων και σε καμιά περίπτωση να μην επιτρέπει την απόπειρα εγγραφής άκυρων ή λανθασμένων δεδομένων στη βάση.

Βελτιώσεις στις Αναφορές

Για να ολοκληρωθούν οι αναφορές δεδομένων από τη βάση, θα πρέπει να γίνει μια έρευνα αγοράς σε σχέση με τα στατιστικά που απαιτούνται σε εφαρμογές τέτοιου τύπου. Αν η εφαρμογή γράφεται για συγκεκριμένο πελάτη, θα πρέπει να γίνει συζήτηση με αυτόν προκειμένου να σχεδιαστούν οι τελικές αναφορές.

Η υλοποίηση των αναφορών δεν είναι δύσκολη, εξαρτάται όμως από την ορθότητα και την αποτελεσματικότητα υλοποίησης της βάσης δεδομένων. Ανάλογα με τις παραδοχές που έχουν γίνει στη σχεδίαση της βάσης, κάποιες αναφορές μπορεί να μην είναι δυνατόν να γραφούν, να έχουν άλλους περιορισμούς ή να εκτελούνται πολύ αργά (αν πρέπει να συνδυάσουν δεδομένα από πίνακες που δεν είναι συνδεδεμένοι μεταξύ τους ήδη).

Αν το πρόγραμμα σχεδιάζεται για πώληση στο ελεύθερο εμπόριο, θα πρέπει οι λειτουργίες του να συγκριθούν με άλλα έτοιμα πακέτα και τουλάχιστον να εξισωθούν με αυτά. Ιδανικά, θα πρέπει να σχεδιαστεί ένας οδηγός αναφορών με τέτοιο τρόπο ώστε ο πελάτης να μπορεί να προσαρμόζει τα δεδομένα που θα βλέπει, χρησιμοποιώντας κάποιο απλό Report Wizard.

Συμπληρωματικά, θα πρέπει να είναι εύκολη η προσθήκη πιο εξελιγμένων αναφορών από την εταιρεία που υποστηρίζει το πρόγραμμα.

Βελτιώσεις στους Υπολογισμούς

Οι υπολογισμοί των ασφαλίστρων μπορούν να γίνουν με διάφορους τρόπους και καθώς οι συντελεστές αλλάζουν

συνέχεια, είναι σημαντικό να δίνεται δυνατότητα στον πελάτη να τους αλλάζει εύκολα.

Όπως και με τη δυνατότητα των αναφορών, ένα εμπορικό πρόγραμμα θα πρέπει να διαθέτει την δυνατότητα παραμετροποίησης των υπολογισμών μέσω ενός εύκολου οδηγού αλλά και την δυνατότητα αναβάθμισης μέσω ενημερώσεων από την εταιρεία υποστήριξης.

Συμπεράσματα

Κατά τη συγγραφή αυτής της εργασίας είχαμε την ευκαιρία να δουλέψουμε και να συνδυάσουμε αρκετές από τις σύγχρονες τεχνολογίες που χρησιμοποιούνται σήμερα για την ανάπτυξη τόσο ιστοσελίδων όσο και διαδικτυακών εφαρμογών.

Το σημαντικότερο για εμάς είναι ότι είδαμε πως μπορούν αυτές οι τεχνολογίες να συνεργαστούν μεταξύ τους για να προκύψει ένα αρμονικό αποτέλεσμα:

- Η **HTML5** για τη συγγραφή των ιστοσελίδων
- Το **CSS3** για τη ρύθμιση της μορφής και το διαχωρισμό της φυσικής από τη λογική μορφοποίηση.
- Την γλώσσα **PHP** που επιτρέπει τη συγγραφή δυναμικών ιστοσελίδων, ουσιαστικά γράφοντας τον **HTML** κώδικα που στέλνεται από τον εξυπηρετητή.
- Τη βάση δεδομένων **MySQL** που θεωρείται το **defacto standard** για την δημιουργία δυναμικών σελίδων.
- Τη γλώσσα **Javascript** για οποιοδήποτε **scripting** απαιτείται πλέον στην πλευρά του **browser**.

Ταυτόχρονα είχαμε την ευκαιρία να γνωρίσουμε πως λειτουργεί και πως στήνεται ένας **LAMP server**, αφού χρησιμοποιήσαμε τη γνωστή διανομή **Debian** για να φτιάξουμε το περιβάλλον ανάπτυξης και δοκιμής της εφαρμογής μας.

Συνολικά, αποκομίσαμε μια σημαντική εμπειρία κατανόησης των παραπάνω τεχνολογιών, της συνεργασίας μεταξύ τους αλλά και των δυσκολιών που παρουσιάζουν στη καθημερινή εφαρμογή τους.