ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ – ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Πρόγραμμα Μεταπτυχιακών Σπουδών
«Πληροφορική»

Μεταπτυχιακή Διατριβή

Τίτλος Διατριβής	**Ανάπτυξη παραθυρικής εφαρμογής .NET Ιατρικών Φακέλων Ασθενών** **Windows Application Development in .NET for Medical Patient Envelopes**
Ονοματεπώνυμο Φοιτητή	**ΤΣΙΤΚΑΝΟΣ ΕΥΑΓΓΕΛΟΣ**
Πατρώνυμο	**ΔΗΜΗΤΡΙΟΣ**
Αριθμός Μητρώου	**ΜΠΠΛ15077**
Επιβλέπων	**Αλέπης Ευθύμιος, Αναπληρωτής Καθηγητής**

Ημερομηνία Παράδοσης **Σεπτέμβριος 2022**

Τριμελής Εξεταστική Επιτροπή

(υπογραφή)	(υπογραφή)	(υπογραφή)
Ευθύμιος Αλέπης Αναπληρωτής Καθηγητής	Μαρία Βίρβου Καθηγήτρια	Κωνσταντίνος Πατσάκης Αναπληρωτής Καθηγητής

Ευχαριστίες

Σε αυτό το σημείο θα ήθελα να ευχαριστήσω όλους εκείνους που συνέβαλλαν και βοήθησαν στην πραγματοποίηση αυτής της διπλωματικής εργασίας.

Θα ήθελα αρχικά να ευχαριστήσω τον επιβλέποντα καθηγητή, κ. Ευθύμιο Αλέπη, που μου έδωσε τη δυνατότητα να ασχοληθώ με ένα τόσο ενδιαφέρον και σύγχρονο θέμα καθώς και την συνολική υποστήριξη του. Επίσης ευχαριστώ τα υπόλοιπα μέλη της τριμελής επιτροπής. Τέλος θα ήθελα να ευχαριστήσω όλους εκείνους που έβαλαν έστω και ένα μικρό λιθαράκι στην περάτωση της διπλωματικής εργασίας και τους γονείς μου για τη συμπαράσταση και την υπομονή τους.

«Ρώτησαν κάποτε ένα μυθικό Βασιλιά:
- Άρχοντα, αν έχανες όλα σου τα αγαθά , ποιο από αυτά τα πέντε θα ήθελες να σου μείνει τελευταίο; Η ελευθερία, η υγεία, το χρυσάφι, η σοφία ή η δόξα;
Και εκείνος αποκρίθηκε:
- Οι πληροφορίες! Δώστε μου πληροφορίες και όλα τα άλλα τα ξαναβρίσκω!»

Ο μύθος αυτός επισημαίνει τη δύναμη της πληροφορίας!

Περιεχόμενα

1 Εισαγωγή

1.1 Περίληψη

Το θέμα της διπλωματικής εργασίας είναι η ανάπτυξη μιας desktop ιατρικής εφαρμογής όπου ο χρήστης γιατρός θα μπορεί να καταχωρεί στοιχεία των ασθενών του και των επισκέψεων τους όπου θα μπορεί να τα επεξεργάζεται και να βγάζει διάφορα συμπεράσματα. Επίσης αναπτύχθηκε μια πολύ απλή εφαρμογή που θα μπορεί ο γιατρός να την έχει στο laptop του και να μπορεί να βρει και να επεξεργαστεί βασικές πληροφορίες των ασθενών χωρίς να έχει την βάση δεδομένων στον υπολογιστή που θα τρέχει η συγκεκριμένη εφαρμογή. Οι τεχνολογίες που χρησιμοποιήθηκαν είναι C# και πιο συγκεκριμένα .NET Framework. Οι εφαρμογές αναπτύχθηκαν με το εργαλείο Visual Studio 2015 και είναι Windows Form Application. Η βάση που χρησιμοποιήσαμε για την αποθήκευση των δεδομένων μας είναι ο SQL Server. Η σύνδεση της βάσης μας με την εφαρμογή έγινε με την τεχνολογία ADO.NET καθώς χρησιμοποιήσαμε και την τεχνολογία WCF για να μπορέσει να επικοινωνήσει η από μακρυσμένη εφαρμογή μας με τον server.

1.2 Abstract

The issue of diploma thesis is the development of a desktop medical application where the user's doctor will be able to record patient data and their visits where they can process them and make different conclusions. It also developed a very simple application that the doctor can have on his laptop and he can find and process basic patient information without having the database on the computer running the application. The technologies which used are C # and more specifically the .NET Framework. The applications were developed with Visual Studio 2015 and are Windows Form Application. The database we used to store our data is MS SQL Server. The connection of our database with the application was done with ADO.NET as we also used WCF technology to be able to communicate from our remote application to the server.

1.3 Σκοπός της Εργασίας

Για την δημιουργία της Διπλωματικής Εργασίας χρειάστηκε να αξιοποιηθούν γνώσεις που αποκτήθηκαν κατά την διάρκεια της φοίτησης στο Μεταπτυχιακό Πρόγραμμα Σπουδών «Πληροφορική» του Πανεπιστημίου Πειραιώς. Συγκεκριμένα, η υλοποίηση της Διπλωματικής Εργασίας απαιτεί την κατανόηση σε ικανοποιητικό βαθμό αρκετών μαθημάτων.

Η παρούσα διπλωματική έρχεται να αντιμετωπίσει σημαντικά ζητήματα υλοποίησης ηλεκτρονικού ιατρικού φακέλου ασθενών. Αντικείμενο της διπλωματικής εργασίας αποτελεί ο σχεδιασμός & υλοποίηση ολοκληρωμένου συστήματος ηλεκτρονικού φάκελου υγείας ασθενούς για κατανεμημένα ιατρικά πληροφοριακά συστήματα ιατρείων. Κύριο μέλημα της εργασίας ήταν να εντοπιστούν καταρχήν τα τεχνολογικά ζητήματα εφαρμογής ενός τέτοιου συστήματος και να προταθούν τεχνολογικές προσεγγίσεις για την αντιμετώπιση τους. Μεγάλη βαρύτητα δόθηκε στην αρχιτεκτονική πρόταση υλοποίησης του ενιαίου συστήματος, αξιοποιώντας τις κρατούσες τεχνολογικές προσεγγίσεις περί ανάπτυξης συστημάτων πολλαπλών επιπέδων (multi-tier), βασισμένων σε τεχνολογίες wcf με την αξιοποίηση τεχνολογιών της Microsoft και συγκερκιμένα του Visual Studio. Επιπλέον, για την αντιμετώπιση των αναγκών ομογενοποίησης και ενιαιοποίησης των δεδομένων του ηλεκτρονικού ιατρικού φακέλου ασθενών, θα σχεδιαστεί ένα σύστημα βάσης δεδομένων με δομές και οντότητες σχεδιασμένες για την εξυπηρέτηση των αναγκών του φακέλου υγείας. Στην συγκεκριμένη διπλωματική χρησιμοποιήσαμε τον SQL Server της Microsoft για να υλοποιήσουμε την βάση δεδομένων. Σημαντικά είναι και τα ζητήματα διαλειτουργικότητας με τρίτα συστήματα οπότε μελετήθηκαν και προτείνονται μηχανισμοί διαλειτουργικότητας που εξυπηρετούν τις ανάγκες ανταλλαγής δεδομένων μεταξύ των συστημάτων. Τέλος, επειδή είναι σημαντικά και τα επιχειρησιακά ζητήματα λειτουργίας ενός τέτοιου συστήματος, η εργασία προσεγγίζει το συγκεκριμένο θέμα με την πρόταση δημιουργίας ενός κεντρικού server που θα αποθηκεύονται τα δεδομένα του ηλεκτρονικού ιατρικού φακέλου των ασθενών. Μέσω αυτού του πιλοτικού συστήματος αναδείχτηκαν τα οφέλη εφαρμογής ενός τέτοιου συστήματος και εντοπίστηκαν πεδία περαιτέρω έρευνας και ανάλυσης που είναι αναγκαία για την εφαρμογή του συστήματος σε ευρεία κλίμακα.

2 Ανασκόπηση Πεδίου

Διανύουμε την Τρίτη μεγάλη επανάσταση του ανθρώπινου Γένους μετά τη Γεωργική Επανάσταση και την Βιομηχανική, την Επανάσταση της Υψηλής Τεχνολογίας και των Ηλεκτρονικών Υπολογιστών. Μετά τη Γεωργική Επανάσταση χρειάστηκε να περάσουν 5000 χρόνια για να γίνει η Βιομηχανική Επανάσταση και μόλις δύο περίπου αιώνες, για να πραγματώσει το ανθρώπινο γένος την Τρίτη και μεγαλύτερη Επανάσταση της Υψηλής Τεχνολογίας και των Η.Υ. και μέσα σε έναν αιώνα να εξελιχτεί ραγδαία.

Έτσι σήμερα η πληροφορία και η γνώση διαδίδονται με ταχύτατους ρυθμούς σε όλη την υφήλιο μέσω των ηλεκτρονικών υπολογιστών.

Μεγάλη η δύναμη της πληροφορίας. Η γνώση, η είδηση για πρόσωπα, ζώα, άτομα, γεγονότα, για τον κόσμο των επιστημών, του κόσμου όλου, είναι το ισχυρότερο όπλο που κράτησε ο άνθρωπος στα χέρια του.

Σήμερα η ποσότητα των πληροφοριών είναι τεράστια, η ροή τους γίνεται πολύ γρήγορα. Αυτό οφείλεται στην Τεχνολογία, στην Επιστήμη της Πληροφορικής και ειδικά στους Ηλεκτρονικούς Υπολογιστές. Ας δώσουμε τους ορισμούς.

1) Όταν λέμε ***Τεχνολογία*** εννοούμε την πρακτική εφαρμογή των τεχνών και των Επιστημών και ιδιαίτερα της μηχανικής προόδου στη Βιομηχανία και το Εμπόριο. Εννοούμε το σύνολο των επιτευγμάτων του ανθρώπου στον τεχνικό τομέα, καθώς και τη γραμμική ανάλυση των λέξεων και των τύπων.
2) Όταν λέμε ***Πληροφορική*** εννοούμε το σύνολο των μεθόδων για τη συλλογή, διαλογή, απομνημόνευση, μετάδοση και χρησιμοποίηση πληροφοριών που επεξεργάζονται αυτόματα, με την βοήθεια προγραμμάτων σε ηλεκτρονικούς υπολογιστές. Είναι η επιστήμη της πληροφόρησης.
3) Όταν λέμε ***Ηλεκτρονικό*** εννοούμε τον σχετικό με τα ηλεκτρόνια, αυτόν που λειτουργεί με τους νόμους της Ηλεκτρονικής.
4) Όταν λέμε ***Ηλεκτρονική*** εννοούμε τον κλάδο της ηλεκτρολογίας που μελετά τη λειτουργία και την εφαρμογή κυκλωμάτων, σε συσχετισμό με ηλεκτρικά ή μαγνητικά σώματα που εισάγονται σ ’αυτά.
5) Όταν λέμε ***Υπολογιστή*** εννοούμε τη μηχανή που λειτουργεί με συγκεκριμένο πρόγραμμα, εκτελεί με ταχύτητα και ακρίβεια υπολογισμούς, και αποθηκεύει στοιχεία στην μνήμη, τα οποία μπορούν να ανακληθούν. Η αποτελεσματικότητα της εξαρτάται από τη δημιουργικότητα και γνώση του χρήστη.

Όποιος έχει ‘’σπουδάσει’’ την ηλεκτρονική και την Επιστήμη της Πληροφορικής και γνωρίζει την λειτουργία των ηλεκτρονικών υπολογιστών, έχει μεγάλη δύναμη στα χέρια του. Πως θα την χρησιμοποιήσει αυτήν την δύναμη; Για να καταστρέψει ή να προάγει τη ζωή;

Φυσικά, θα πρέπει να τη χρησιμοποιήσει για την προαγωγή της ζωής, τη βελτίωση της, προς όφελος όλων των ανθρώπων.

Αυτό θα πρέπει να υπηρετήσει και η Ιατρική Πληροφορική ως κλάδος της Πληροφορικής.

6) Όταν λέμε ***Ιατρική*** εννοούμε την επιστήμη που έχει ως αντικείμενο τη διατήρηση της υγείας και την καταπολέμηση των ασθενειών του ανθρώπου.
7) Όταν λέμε ***Ιατρικός*** εννοούμε τον σχετικό με την ιατρεία και τους γιατρούς.

Ιατρική Πληροφορική είναι ο κλάδος που περιλαμβάνει όλες τις εφαρμογές της πληροφορικής στην ιατρική και περιλαμβάνει και πληροφοριακές εφαρμογές στην νοσηλευτική και την φαρμακολογία.

Σύμφωνα με την Εθνική Ιατρική Βιβλιοθήκη των Η.Π.Α. η Πληροφοριακή Ιατρική ορίζεται ως ο διεπιστημονικός κλάδος που ασχολείται με την μελέτη, τον σχεδιασμό, την ανάπτυξη, την υιοθέτηση και την εφαρμογή της πληροφορικής στον τομέα των υπηρεσιών υγειονομικής περίθαλψης, διαχείρισης και σχεδιασμού.

Ο όρος Πληροφορική Ιατρική (Medical Informatics) πολλοί θεωρούν ότι αποτελεί υποσύνολο του όρου Πληροφορική της Υγείας (Health Informatics). Άλλοι τον χρησιμοποιούν παράλληλα ως συνώνυμο.

2.1 Η εξέλιξη της Πληροφορικής στην Υγεία

Η πρώτη εταιρία που σχετίζει τους Ηλεκτρονικούς Υπολογιστές με την Ιατρική είναι η Γερμανική Εταιρία για την Ιατρική Τεκμηρίωση, Επιστήμη των Υπολογιστών και Στατιστική (German Society for Medical Documentation, Computer Science and Statistics), που ιδρύεται το 1949 από τον Gustav Wagner.

Στο συνέδριο της Αμερικάνικης Εταιρείας Κλινικής Παθολογίας το 1952 ο Arthur E.Rappoport παρουσίασε την εμπειρία του από την χρήση της συσκευής McBee, για την εισαγωγή διατρητών καρτών για την καταγραφή εργαστηριακών δεδομένων (Porth; Lubke; 1996; Park, et al., 2013)

Το 1950 οι Ledley και Lusted δημοσίευσαν στο περιοδικό Science μια εργασία με τίτλο ”Reasoning Foundations of Medical Diagnosis”. Σε αυτήν εξέφρασαν την ελπίδα ότι με την αξιοποίηση των ηλεκτρονικών υπολογιστών και της πληροφορικής μεγάλο μέρος της δουλειάς των γιατρών θα αυτοματοποιηθεί και έτσι θα μπορούσαν να αποφευχθούν πολλά ανθρώπινα λάθη.

Ο όρος Πληροφορική Ιατρική εμφανίζεται στη Γαλλία το 1960 και το 1964 ιδρύεται το αντίστοιχο πανεπιστημιακό τμήμα MEDLARS (Medical Literature Analysis and Retrieval System) αρχίζει την λειτουργία του. Είναι μια δημόσια βιβλιογραφική βάση δεδομένων βιοϊατρικής πληροφορίας. Η βάση περιλαμβάνει βιβλιογραφικές πληροφορίες για άρθρα από περιοδικά που καλύπτουν την ιατρική, νοσηλευτική, φαρμακευτική, οδοντιατρική, κτηνιατρική και υγειονομική περίθαλψη.

Το 1969 αναπτύσσεται το ARPANET, που αποτελεί πρόδρομο του INTERNET, και το MEDLARS γίνεται διαθέσιμο και μέσω του διαδικτύου, με την MEDLARS online, το 1971.

Το πρώτο σύστημα ιατρικής υποστήριξης χρησιμοποιείται στο Νοσοκομείο Latter-Day Saints στη Γιούτα των Η.Π.Α. Το λογισμικό ονομάζονταν HELP (Health Evaluation through Logical Processing) και χρησιμοποιείται το 1967.

Το 1970 κάνει την εμφάνιση του το MYCIN. Χρησιμοποιείται, για να προσδιορίσει τα βακτηρίδια που προκαλούν λοιμώξεις.

Από κει και πέρα η χρήση της Πληροφορικής στην Υγεία γενικεύεται και θεωρείται αυτονόητη.

2.2 Εφαρμογές της Ιατρικής Πληροφορικής

Η Ιατρική Πληροφορική προσφέρει πολύτιμες υπηρεσίες με τον αυτοματισμό και τους ηλεκτρονικούς υπολογιστές σε όλους τους θεράποντες της υγείας, προσφέροντας τους τις απαραίτητες πληροφορίες, για να βελτιώσουν την προσφορά τους στον ασθενή.

Μία βασική εφαρμογή της Ιατρικής Πληροφορικής είναι ο *Ηλεκτρονικός Ιατρικός Φάκελος Ασθενούς*.

Όταν λέμε φάκελο εννοούμε τη χάρτινη θήκη και επιστολή ή έγγραφο και το σύνολο των εγγράφων που αναφέρονται σε ορισμένο θέμα.

Μέχρι πριν λίγα χρόνια οι ιατρικές πληροφορίες που αφορούσαν τους ασθενείς οργανώνονται και αρχειοθετούνταν χειρόγραφα σε χάρτινες καρτέλες. Σήμερα όμως ο όγκος των πληροφοριών είναι μεγάλος. Έτσι καθίσταται αναγκαία η εύρεση άλλου τρόπου αρχειοθέτησης και διατήρησης των ιατρικών πληροφοριών. Αυτό προσφέρει ο Ηλεκτρονικός Ιατρικός Φάκελος Ασθενούς.

Τα πλεονεκτήματα και τα μειονεκτήματα του Ηλεκτρονικού Ιατρικού Φακέλου Ασθενούς μας τα παρουσιάζουν (καθαρά) τρία πρόσωπα που κάνουν χρήση του Η.Ι.Φ.Α. μέσα από τις συνεντεύξεις τους.

1) Συνέντευξη με τον γιατρό καρδιολόγο Σπυρίδωνα Σαβιτσάνο, συμβεβλημένο με τον ΕΟΠΠΥ.

Ερώτηση: Ποια θεωρείτε ότι είναι τα πλεονεκτήματα και ποια τα μειονεκτήματα του Ηλεκτρονικού Ιατρικού Φακέλου Ασθενούς;

Απάντηση: Τα πλεονεκτήματα είναι πάρα πολλά.

- Ο γιατρός γνωρίζει ακριβώς τα προβλήματα του ασθενή. Είναι πιο κοντά στον ασθενή.

- Γνωρίζει την φαρμακευτική αγωγή, καθώς και την εξέλιξη της νόσου. Έχει γνώση της αποτελεσματικότητας της Φαρμακευτικής Αγωγής.

- Μεγάλο πλεονέκτημα είναι ότι καταχωρούνται στο φάκελο οι εξετάσεις και δεν χάνονται, γιατί είναι σύνηθες φαινόμενο οι ασθενείς να τις χάνουν.

- Ενώ στην αρχή μπορεί να χρειάζεται περισσότερος χρόνος, για να δημιουργήσεις και να οργανώσεις τον ηλεκτρονικό φάκελου του ασθενή, στη συνέχεια τα βρίσκεις έτοιμα και κερδίζεις χρόνο από αυτό.

- Ο φάκελος του ασθενή έχει όλα τα στοιχεία του ασθενή. Όνομα, επώνυμο, διεύθυνση, ΑΜΚΑ, τηλέφωνα όχι μόνο του ασθενή αλλά και συγγενικών προσώπων. Οπότε όταν χρειαστεί, ο γιατρός ενημερώνει αμέσως τους συγγενής για άμεση αντιμετώπιση της ασθένειας.

- Πολλοί ασθενείς ξεχνούν τα στοιχεία τους, την φαρμακευτική αγωγή που λαμβάνουν, τη διεύθυνση τους, τα τηλέφωνα των παιδιών τους. Εγώ με ένα κλικ τα βρίσκω αμέσως στο φάκελο του.

- Έχω την δυνατότητα να κρατάω τις εξετάσεις του ασθενή, που πολλές φορές τις χάνει ή νομίζει ότι δεν χρειάζεται να τις φέρει μαζί του για σύγκριση με τις νέες. Έτσι κρατάω αρχείο ψηφιακά των στεφανογραφιών, των triplex, και όταν τις χρειαστούν οι ασθενείς σε άλλη περίπτωση, όπως εισαγωγή τους σε νοσοκομείο, τους τις δίνω.

- Το σημαντικό για τον γιατρό είναι ότι προστατεύει και τον ίδιο από ψευδείς καταγγελίες. Αν προκύψει τέτοιο θέμα ανοίγει τον φάκελο του και αμέσως έχει την εικόνα του ασθενή. Και προλαβαίνει και τις παραλείψεις.

- Μπορώ να κάνω επαναληπτικό, να διορθώνω και να μεταφέρω και αλλού με ένα απλό copy paste στα αρχεία.

- Μπορώ να εκτυπώσω όλες τις εξετάσεις του ασθενούς.

- Για την προστασία των προσωπικών δεδομένων των ασθενών, δουλεύω χωρίς σύνδεση στο Internet, για να μην πέσω θύμα hackers και υποκλαπούν.

- Κάνω δύο buck-up κάθε μέρα και τέσσερα ανά εξάμηνο.

2) Συνέντευξη με την κ. Ιωάννα Κέκου, γραμματέας σε ιατρείο

- *Ερώτηση:* Ποια θεωρείτε ότι είναι τα πλεονεκτήματα και ποια τα μειονεκτήματα του Ηλεκτρονικού Ιατρικού Φακέλου Ασθενούς;

Απάντηση: Πολλά τα πλεονεκτήματα του Η.Ι.Φ.Α. σε σχέση με την παλιά χειρόγραφη καρτέλα ή μόνο πλεονεκτήματα έχει.

- Μεγάλο πλεονέκτημα είναι η εξοικονόμηση χρόνου. Πατάω το όνομα του ασθενή στον υπολογιστή και αμέσως βλέπω όλη την εικόνα του ασθενή. Ενώ πριν έπρεπε να ψάξω να βρω την καρτέλα.

- Οι παλιές καρτέλες έπιαναν χώρο. Έπρεπε να είναι ταξινομημένες, για να κερδίσω χρόνο, έχοντας εύκολη πρόσβαση σε αυτές. Έπρεπε μετά από κάθε χρήση τους να τις τοποθετήσω στη θέση τους. Χάσιμο χρόνου άσκοπα.

- Δεν συμπληρώνω χειρόγραφα. Αυτό είχε παλιά τον κίνδυνο να αλλοιωθούν από την πολυκαιρία. Ή να χαθούν, να καταστραφούν σε περίπτωση φωτιάς, γιατί δεν υπήρχαν αντίγραφα των καρτελών εκτός του ιατρείου.

- Μειονέκτημα αποτελεί η πιθανή παραβίαση των προσωπικών δεδομένων. Ο γιατρός όμως έχει διασφαλίσει ή πρέπει να διασφαλίζει την προστασία τους.

- Με το πάτημα του πλήκτρου βρίσκω τα πάντα για τον ασθενή. Τα φάρμακα που παίρνει, τι εξετάσεις έχει κάνει, τι εκκρεμότητες έχει. Πριν αυτές τις πληροφορίες έπρεπε να τις δίνει ο ασθενής. Αυτό δεν είναι πολλές φορές εύκολο, γιατί οι περισσότεροι ασθενείς ξεχνούν και δεν θυμόνται τι φάρμακα παίρνουν, τί εξετάσεις έκαναν.

3) Συνέντευξη με τον κ. Ιωάννη Δούκα, τεχνικό πληροφορικών συστημάτων του κλάδου της Υγείας (γιατροί, φαρμακεία, νοσοκομεία).

- *Ερώτηση:* Ποια θεωρείτε ότι είναι τα πλεονεκτήματα και ποια τα μειονεκτήματα του Ηλεκτρονικού Ιατρικού Φακέλου Ασθενούς;

Απάντηση: Πολλά τα πλεονεκτήματα του Ηλεκτρονικού Ιατρικού Φακέλου.

- Πρώτο και σημαντικό είναι η αξιοπιστία. Ο άνθρωπος κάνει λάθη, η μηχανή δεν κάνει.

- Προσφέρει καλύτερη οργάνωση και δομή. Σωστή ενημέρωση κάθε ενδιαφερόμενου από την κορυφή μέχρι τον τελευταίο, των πληροφοριών που χρειάζεται για τον ασθενή.

- Ταχύτητα στο να έχουν ανά πάσα στιγμή από μακριά ή κοντά πρόσβαση στην καρτέλα του ασθενή, στα στοιχεία του, στη φαρμακευτική αγωγή, διαφορετικά τμήματα και υπηρεσίες σε ίδια στοιχεία.

- Παρέχει ασφάλεια. Προϋπόθεση η σωστή συντήρηση του Πληροφοριακού Συστήματος. Να γίνεται back-up.

Η πολιτεία αναγνωρίζοντας ότι η προαγωγή της Δημόσιας Υγείας μπορεί να γίνει και μέσω του Ηλεκτρονικού Ιατρικού Φακέλου Ασθενούς, κατέθεσε τον Ιούλιο του 2017 νομοσχέδιο για την υποχρεωτική χρήση του Η.Ι.Φ.Α. από τις αρχές του 2018.

Νομοσχέδιο του Υπουργείου Υγείας «Μεταρρύθμιση της Πρωτοβάθμιας Φροντίδας Υγείας, επείγουσες ρυθμίσεις αρμοδιότητας του Υπουργείου Υγείας άλλες διατάξεις».

Σύμφωνα με το εν λόγω νομοσχέδιο όλοι οι πολίτες θα έχουν Ηλεκτρονικό Ιατρικό Φάκελο ο οποίος θα περιέχει συνοπτικό ιατρικό ιστορικό. Ο φάκελος θα καταρτίζεται από τον οικογενειακό γιατρό που θα έχει την ευθηνή ενημέρωσης του. Τα προσωπικά δεδομένα θα τηρούνται με ευθηνή του Υπουργείου Υγείας. Ενώ πρόσβαση σε αυτόν θα έχει ο οικογενειακός γιατρός, ο πολίτης και οι μονάδες του ΕΣΥ, εφόσον απαιτείται νοσοκομειακή περίθαλψη.

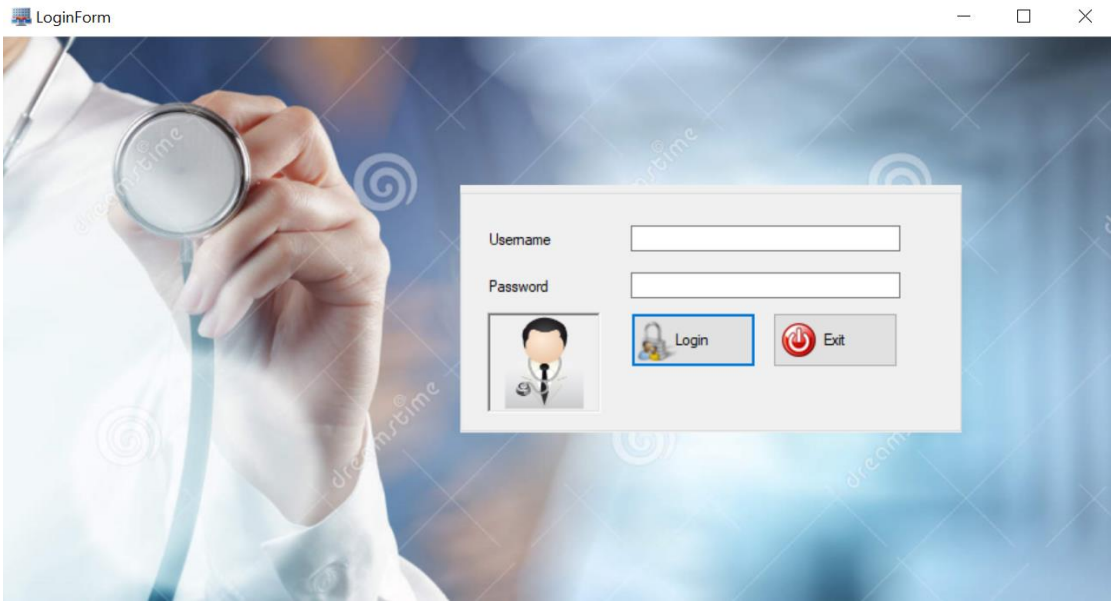
Σύμφωνα με τον υπουργό Υγείας στο φάκελο θα καταχωρείται υλικό που θα αφορά σε παρεμβάσεις πρόσληψης, τους προ συμπτωματικούς ελέγχους, την κλινική και εργαστηριακή παρακολούθηση, τη φαρμακευτική, τη νοσηλεία, την αποκατάσταση ή οποιαδήποτε άλλη υγειονομική φροντίδα παρέχεται στον ασθενή.

Ο ηλεκτρονικός φάκελος συνιστά (κατά την αιτιολογική έκθεση του Υπουργείου Υγείας για το νομοσχέδιο) το ηλεκτρονικό εργαλείο που θα διασφαλίσει τη συνέχεια στη φροντίδα, ενώ παράλληλα αποτελεί προαπαιτούμενο της ορθής και αποτελεσματικής εφαρμογής της παραπομπής.

(Από τον Ημερήσιο Τύπο)

3 Εγχειρίδιο Χρήσης

Η αρχική φόρμα της εφαρμογής ζητάει από τον χρήστη να βάλει Username και Password για να κάνει ταυτοποίηση των στοιχείων του ώστε να του επιτρέψει την είσοδο του στο κύριο μέρος της εφαρμογής.

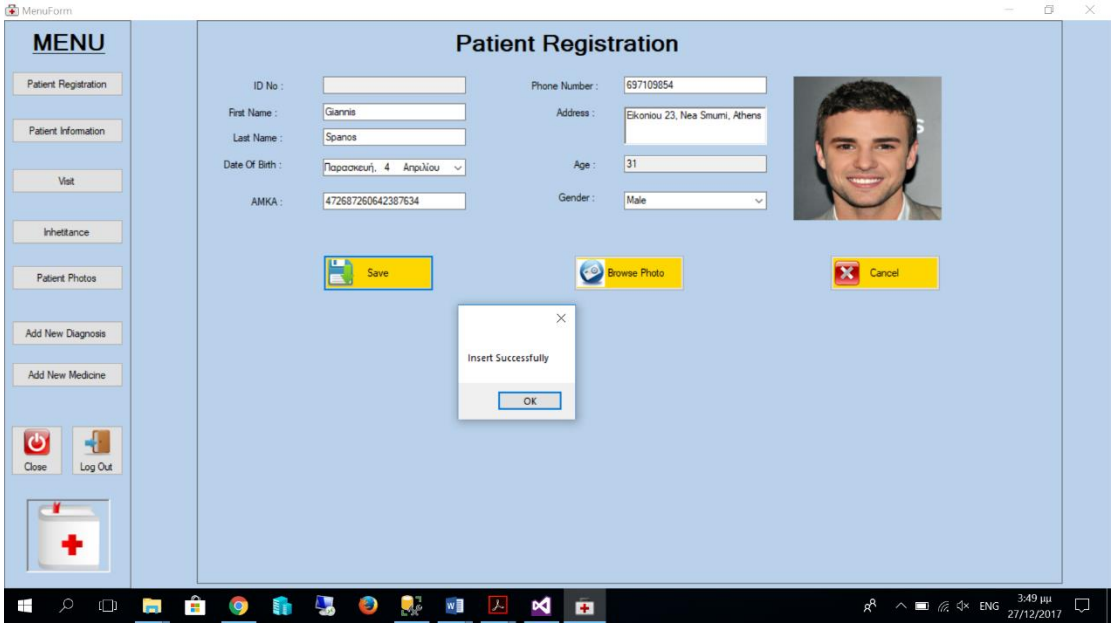


Εικόνα 1. Εμφάνιση φόρμας Login

Στην περίπτωση που ο χρήστης βάλει λάθος στοιχεία του εμφανίζεται μήνυμα ‘’Invalid Login’’. Ενώ στην αντίθετη περίπτωση βγαίνει μήνυμα “Login Successfull” και μεταφερόμαστε στο κύρια φόρμα της εφαρμογής.

Εικόνα 2. Αρχική φόρμα εφαρμογής

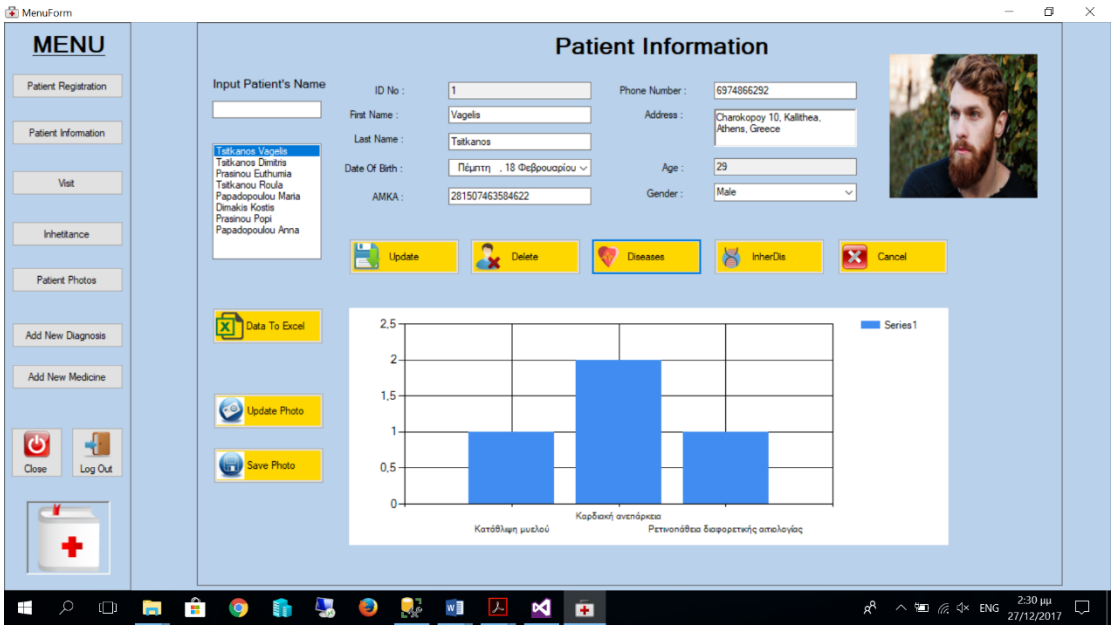
Η πρώτη επιλογή είναι εγγραφή ασθενούς στο σύστημα μας.



Εικόνα 3. Εγγραφή ασθενούς

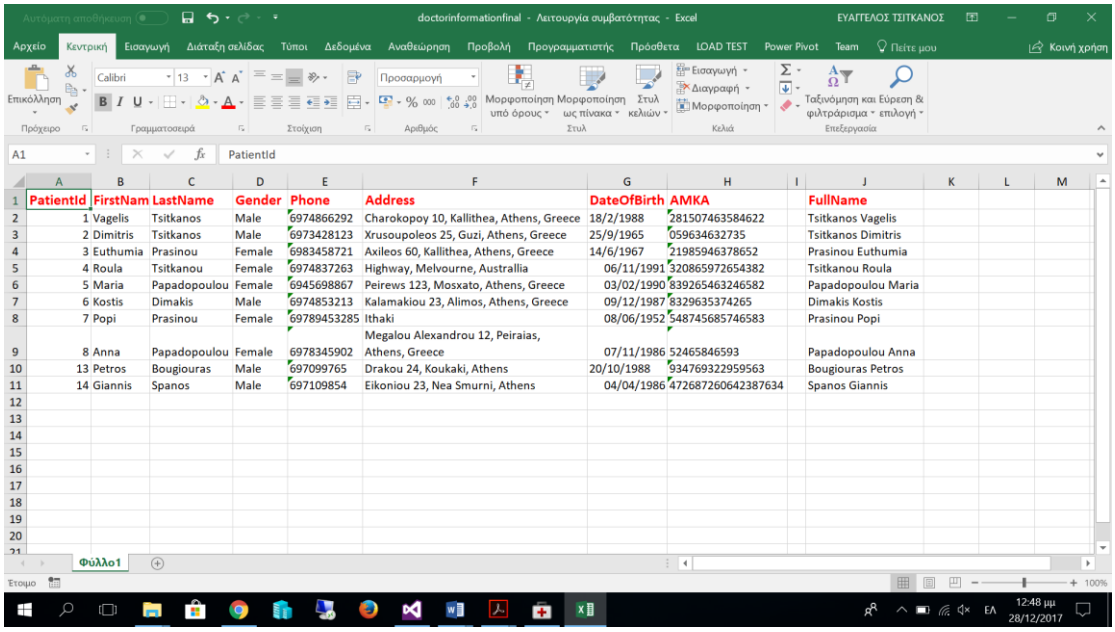
Στην φόρμα αυτή περνάμε τα στοιχεία του πελάτη καθώς έχουμε την δυνατότητα να περάσουμε και φωτογραφία του. Η φωτογραφία εισάγεται από το κουμπί ‘Browse Photo’ και η αποθήκευση στην βάση μας γίνεται όταν πατάμε το κουμπί ‘Save’. Στην περίπτωση που η εγγραφή έγινε ομαλά μας βγαίνει μήνυμα ‘Insert Successful’ αλλιώς αν δεν συμπληρώσουμε τα πεδία του ονόματος και του επιθέτου μας βγάζει μήνυμα ‘Please Insert FirstName and LastName’ και δεν γίνεται καμία εγγραφή.

Η δεύτερη επιλογή είναι η επεξεργασία των δεδομένων του ασθενούς.



Εικόνα 4. Επεξεργασία ασθενούς

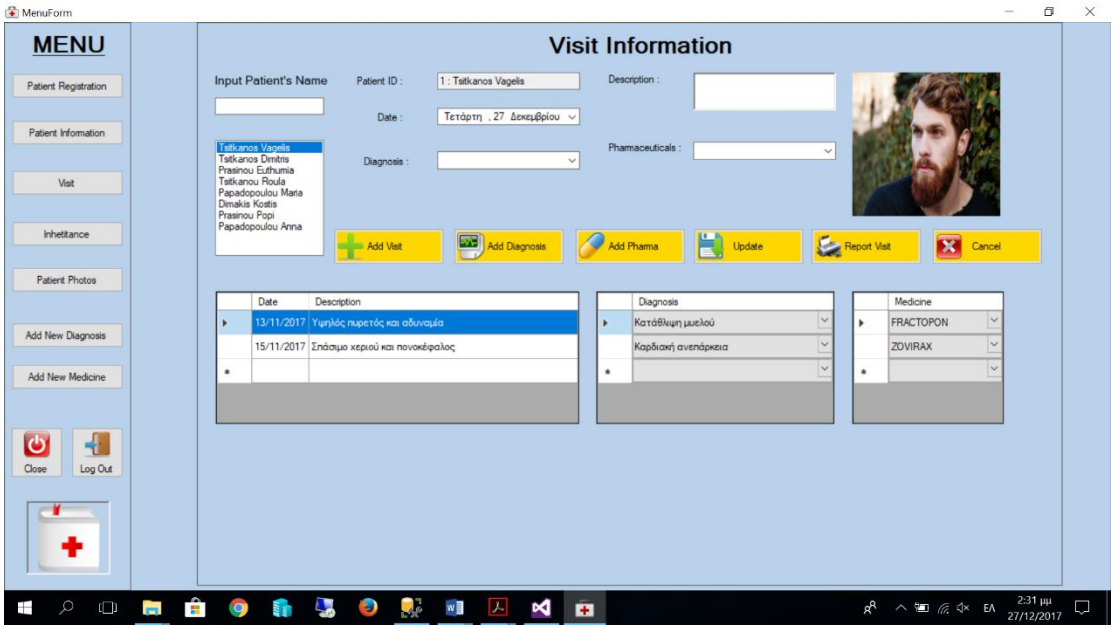
Σε αυτή την καρτέλα ο χρήστης μπορεί να κάνει αναζήτηση κάποιου ασθενούς με βάση το όνομα ή το επίθετο του. Στην συνέχεια μπορεί να αλλάξει κάτι από τα στοιχεία του και πατώντας το κουμπί Update να ενημερώσει τη βάση δεδομένων. Ακόμη του δίνεται η δυνατότητα να κάνει διαγραφή ενός ασθενούς από την βάση με το κουμπί Delete. Για λόγους ασφάλειας όταν πατηθεί το κουμπί Delete βγαίνει μήνυμα στον χρήστη ‘Are you sure to remove the record’ και του δίνει τις επιλογές ‘Yes’ και ‘No’. Στην πρώτη περίπτωση αν η διαγραφή ολοκληρωθεί επιτυχώς εμφανίζεται μήνυμα ‘Patient Record Removed…’ ενώ στην δεύτερη περίπτωση δεν γίνεται καμία ενέργεια. Όταν πατηθεί το κουμπί Diseases εμφανίζεται διάγραμμα που μας δείχνει τις αρρώστιες και το πόσες φορές έχουν παρουσιαστεί στον συγκεκριμένο ασθενή. Ακριβώς δίπλα είναι το κουμπί InherDis που όταν το πατήσει ο χρήστης εμφανίζονται στο διάγραμμα οι αρρώστιες και ο αριθμός τους που έχουν καταγραφεί σε συγγενείς του επιλεγμένου ασθενή. Τα κουμπιά Update Photo και Save Photo είναι μόνο για την αλλαγή και την αποθήκευση της φωτογραφίας του ασθενούς. Τέλος το κουμπί Data To Excel δημιουργεί ένα αρχείο excel που περιέχει όλους τους ασθενείς και τα προσωπικά τους στοιχεία που έχουν καταχωρηθεί στην πρώτη φόρμα εκτός από τις φωτογραφίες τους. Στο πάτημα του κουμπιού εμφανίζεται μήνυμα ότι το αρχείο δημιουργήθηκε και σε ποιο φάκελο βρίσκεται.



PatientId	FirstNam	LastName	Gender	Phone	Address	DateOfBirth	AMKA	FullName
1	Vagelis	Tsitkanos	Male	6974866292	Charokopoy 10, Kallithea, Athens, Greece	18/2/1988	281507463584622	Tsitkanos Vagelis
2	Dimitris	Tsitkanos	Male	6973428123	Xrusoupoleos 25, Guzi, Athens, Greece	25/9/1965	059634632735	Tsitkanos Dimitris
3	Euthumia	Prasinou	Female	6983458721	Axileos 60, Kallithea, Athens, Greece	14/6/1967	21985946378652	Prasinou Euthumia
4	Roula	Tsitkanou	Female	6974837263	Highway, Melvourne, Australlia	06/11/1991	320865972654382	Tsitkanou Roula
5	Maria	Papadopoulou	Female	6945698867	Peirews 123, Mosxato, Athens, Greece	03/02/1990	839265463246582	Papadopoulou Maria
6	Kostis	Dimakis	Male	6974853213	Kalamakiou 23, Alimos, Athens, Greece	09/12/1987	8329635374265	Dimakis Kostis
7	Popi	Prasinou	Female	69789453285	Ithaki	08/06/1952	548745685746583	Prasinou Popi
8	Anna	Papadopoulou	Female	6978345902	Megalou Alexandrou 12, Peiraias, Athens, Greece	07/11/1986	52465846593	Papadopoulou Anna
13	Petros	Bougiouras	Male	697099765	Drakou 24, Koukaki, Athens	20/10/1988	934769322959563	Bougiouras Petros
14	Giannis	Spanos	Male	697109854	Eikoniou 23, Nea Smurni, Athens	04/04/1986	472687260642387634	Spanos Giannis

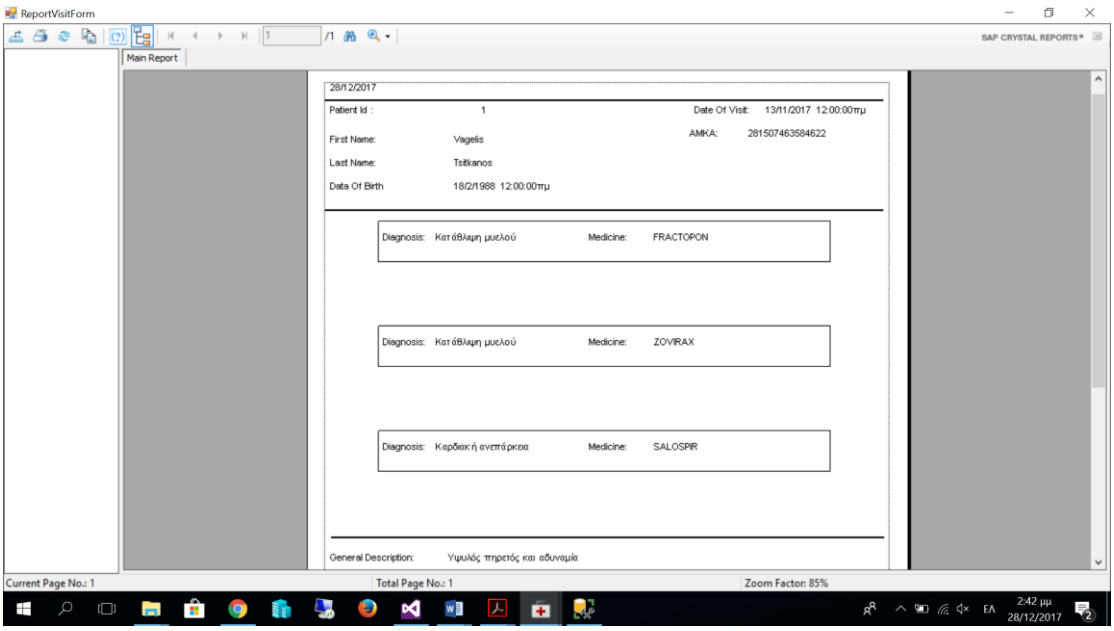
Εικόνα 5. Στοιχεία ασθενών σε αρχείο excel

Στην τρίτη επιλογή γίνεται η καταχώρηση και η επεξεργασία των επισκέψεων.



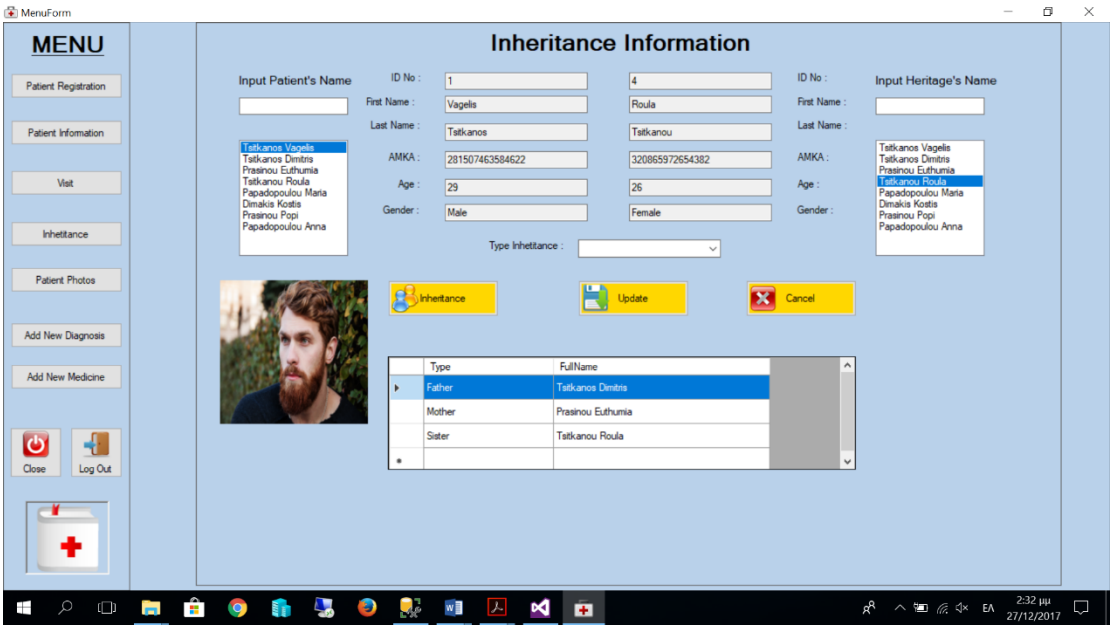
Εικόνα 6. Εισαγωγή και επεξεργασία επισκέψεων ασθενών

Σε αυτήν την καρτέλα γίνεται η εισαγωγή και η επεξεργασία των επισκέψεων των ασθενών. Υπάρχει μια ιεραρχία όσο αναφορά την καταχώρηση των επισκέψεων. Αρχικά ο χρήστης πρέπει να καταχωρήσει ημερομηνία επίσκεψης και μια μικρή περιγραφή. Πατώντας το κουμπί Add Visit καταχωρείτε στην βάση η επίσκεψη. Στην συνέχεια για την συγκεκριμένη επίσκεψη μπορεί να προσθέσει διαγνώσεις που τις έχει καταχωρήσει από πριν στην βάση. Τις διαγνώσεις μπορεί να τις επιλέξει με δύο τρόπους. Πρώτος τρόπος από το κουτί που βρίσκεται κάτω από την ημερομηνία και δεύτερος τρόπος από το κουτί που βρίσκεται μέσα στον πίνακα των διαγνώσεων. Πατώντας το κουμπί Add Diagnosis καταχωρούνται στην βάση οι διαγνώσεις της συγκεκριμένης επίσκεψης. Με τον ίδιο τρόπο γίνεται και η καταχώρηση των φαρμάκων για κάθε διάγνωση του ασθενούς ξεχωριστά. Δεν μπορεί να καταχωρηθεί φάρμακο αν δεν έχει επιλεχτεί κάποια διάγνωση. Αντίστοιχα πατώντας το κουμπί Add Pharma αποθηκεύουμε στην βάση μας τα φάρμακα που περιέχει η διάγνωση συγκεκριμένης επίσκεψης του ασθενούς. Για την διαγραφή οποιασδήποτε γραμμής πίνακα απλά την επιλέγουμε και πατάμε το κουμπί Delete από το πληκτρολόγιο μας. Για οποιαδήποτε αλλαγή και αν κάνει ο χρήστης μετά πρέπει να πατήσει το κουμπί Update ώστε να καταχωρηθούν οι αλλαγές επιτυχώς στην βάση. Το κουμπί Report Visit δημιουργεί ένα report της επίσκεψης του ασθενούς.



Εικόνα 7. Αναφορά επίσκεψης ασθενούς

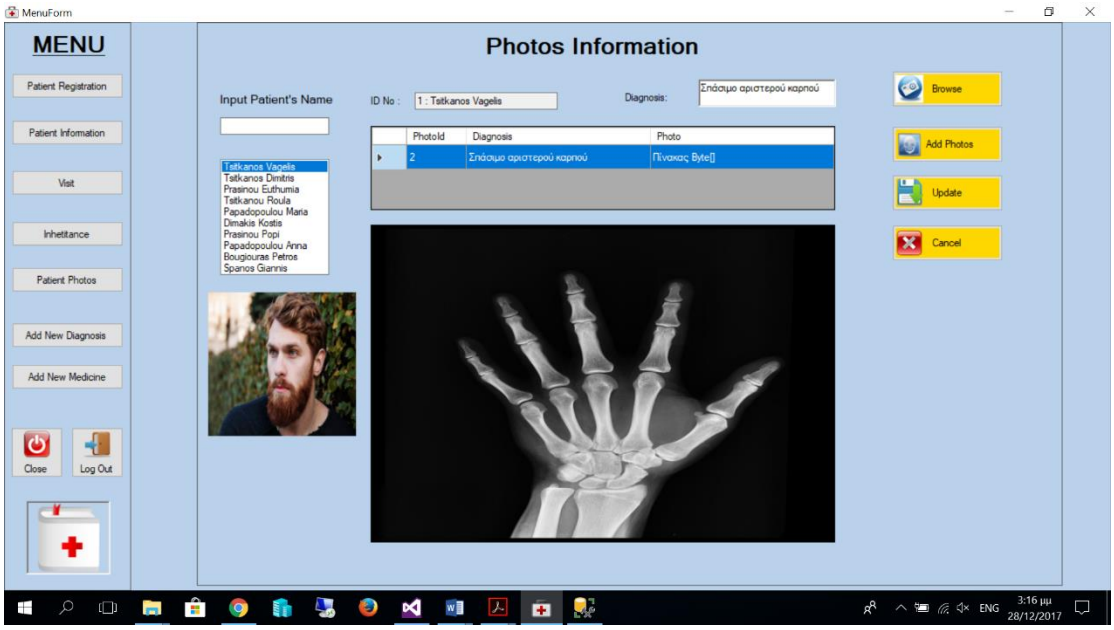
Στην τέταρτη επιλογή γίνεται η επεξεργασία και η καταχώρηση της σχέσης που έχουν μεταξύ τους οι ασθενείς.



Εικόνα 8. Εισαγωγή και επεξεργασία κληρονομικότητας ασθενών

Σε αυτή την καρτέλα γίνεται αναζήτηση ασθενών που έχουν συγγένεια μεταξύ τους. Η επιλογή τύπου συγγένειας γίνεται από το κουτί και η καταχώρηση στην βάση όταν πατηθεί το κουμπί Inheritance. Υπάρχει η δυνατότητα να αλλαχτούν τα δεδομένα στον πίνακα κάνοντας κλικ μέσα στο κελί που θέλει ο χρήστης καθώς και η διαγραφή κάποιας σχέσης πατώντας Delete από το πληκτρολόγιο έχοντας επιλέξει την γραμμή που είναι προς διαγραφή από τον πίνακα. Μετά από κάθε επεξεργασία των δεδομένων πατώντας το κουμπί Update ενημερώνεται η βάση.

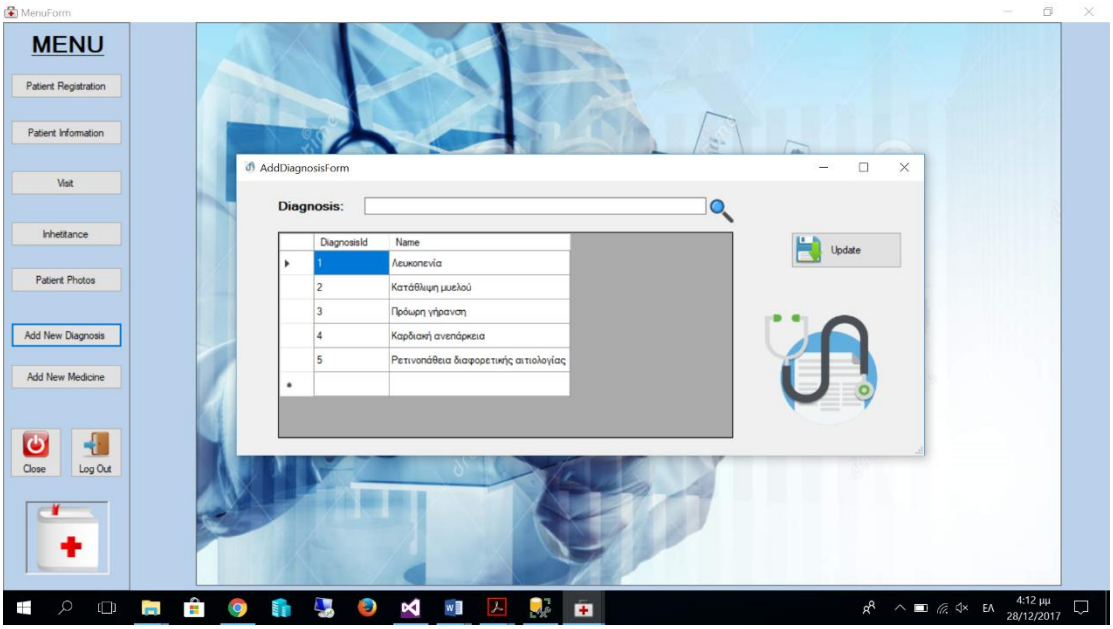
Στην πέμπτη επιλογή γίνεται η επεξεργασία ιατρικών φωτογραφιών του κάθε ασθενούς.



Εικόνα 9. Εισαγωγή και επεξεργασία φωτογραφιών των ασθενών

Σε αυτή την καρτέλα γίνεται αναζήτηση του ασθενούς για να δούμε το ιστορικό των φωτογραφιών του. Υπάρχει η δυνατότητα πρόσθεσης φωτογραφίας πατώντας το κουμπί Browse. Η αποθήκευση γίνεται πατώντας το κουμπί Add Photos ενώ αν χρειαστεί να γίνει αλλαγή στην διάγνωση πρέπει να πατηθεί το κουμπί Update ώστε να αποθηκευτούν οι αλλαγές στην βάση. Επίσης η διαγραφή φωτογραφίας γίνεται επιλέγοντας την γραμμή από τον πίνακα και πατώντας Delete από το πληκτρολόγιο. Μετά από κάθε διαγραφή πρέπει να γίνεται η χρήση του κουμπιού Update ώστε να καταχωρηθούν οι αλλαγές στην βάση.

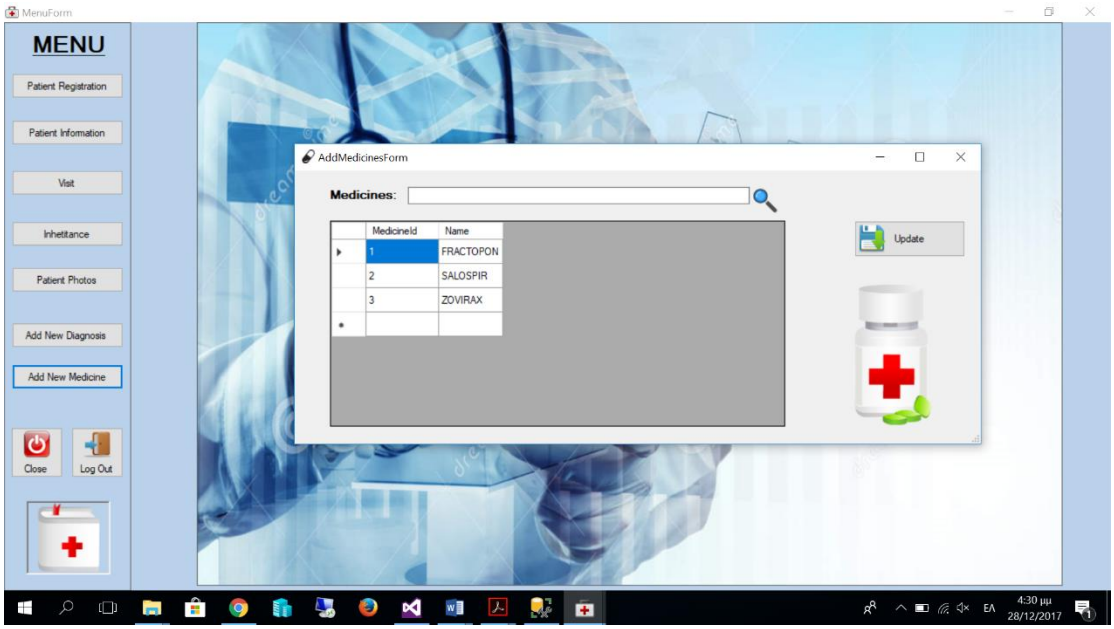
Η επόμενη επιλογή δίνει την δυνατότητα στον χρήστη να καταχωρήσει διαγνώσεις στην βάση ώστε να μπορεί να τις επιλέγει όταν κάνει καταχώρηση επίσκεψης ασθενούς.



Εικόνα 10 Φόρμα εισαγωγής και επεξεργασίας διαγνώσεων

Στο πάνω μέρος της φόρμας γίνεται αναζήτηση των διαγνώσεων και κάνοντας κλικ σε κενό κελί της στήλης με τα ονόματα γίνεται εισαγωγή μιας νέας διάγνωσης. Με τον ίδιο τρόπο αν το κλικ γίνει σε κελί που έχει ήδη διάγνωση υπάρχει η δυνατότητα επεξεργασίας. Η διαγραφή διάγνωσης γίνεται όταν επιλεγεί η σειρά που θέλουμε από τον πίνακα και πατήσουμε Delete από το πληκτρολόγιο μας. Ότι ενέργεια και να γίνει στο τέλος πρέπει να γίνει κλικ στο κουμπί Update για να αποθηκευτούν οι αλλαγές στην βάση δεδομένων.

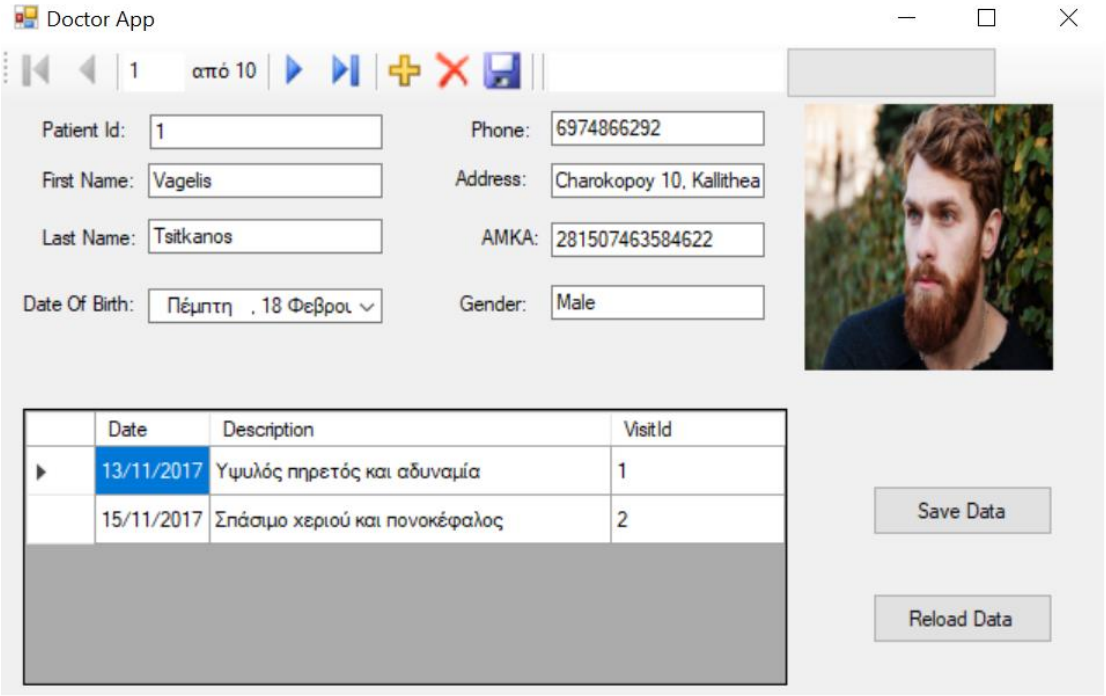
Ακριβώς με τον ίδιο τρόπο γίνεται και η επεξεργασία των φαρμάκων κάνοντας κλικ στην επόμενη επιλογή και μας εμφανίζεται η παρακάτω φόρμα.



Εικόνα 11. Φόρμα εισαγωγής και επεξεργασίας φαρμάκων

Τέλος υπάρχουν δύο επιλογές. Το κλικ πάνω στο κουμπί Close που κλείνει την εφαρμογή και το κουμπί Log Out που όταν γίνει κλικ βγαίνει από το κύριο μενού της εφαρμογής και εμφανίζεται η αρχική Login Form.

Η δεύτερη εφαρμογή μπορεί να εγκατασταθεί σε λειτουργικό windows έχει την παρακάτω μορφή



Εικόνα 12. Φόρμα εισαγωγής και επεξεργασίας δεδομένων

Σε αυτή την φόρμα πατώντας το κουμπί Reload Data φορτώνουν τα δεδομένα της βάσης που βρίσκονται στον απομακρυσμένο server. Μπορεί να γίνει προσθήκη ασθενούς, επεξεργασία και διαγραφή κάποιου υπάρχοντος. Η ενημέρωση των αλλαγών στη βάση γίνεται με το πάτημα του κουμπιού Save Data.

4 Κύκλος Ζωής Λογισμικού

Η διαδικασία ανάπτυξης λογισμικού αποκαλείται και κύκλος ζωής του λογισμικού επειδή περιγράφει τη ζωή ενός λογισμικού προϊόντος από την σύλληψη της ιδέας μέχρι την υλοποίηση, την παράδοση, τη χρήση και την συντήρηση του. Οι παρακάτω ενέργειες είναι σημαντικές επειδή επιβάλλουν συνέπεια και δομή σε ένα σύνολο δραστηριοτήτων. Η ανάπτυξη λογισμικού περιλαμβάνει τα εξής στάδια:

- Διερευνητική Μελέτη
- Μελέτη Σκοπιμότητάς
- Ανάλυση Απαιτήσεων
- Σχεδιασμός Συστήματος
- Υλοποίηση
- Εγκατάσταση
- Λειτουργία και Συντήρηση

Στην πρώτη διαδικασία γίνεται ο ορισμός του συστήματος και του προβλήματος, και η παρουσίαση ενναλακτικών λύσεων. Εδώ πραγματοποιείται επιλογή της βέλτιστης λύσης για μελέτη.

Στην δεύτερη διαδικασία γίνεται ο έλεγχος κατά πόσο εφικτή είναι η λύση που επιλέχτηκε. Παρουσιάζονται εναλλακτικοί τρόποι υλοποίησης και γίνεται η ανάλυση του κόστους/οφέλους της λύσης που προτάθηκε. Τέλος δίνεται μια ολοκληρωμένη περιγραφή της λύσης που θα πραγματοποιηθεί.

Το στάδιο της ανάλυσης απαιτήσεων, επικεντρώνεται στο τι πρέπει να κάνει ένα σύστημα λογισμικού και παρέχεται μια μηχανική περιγραφή των αντικειμένων, των λειτουργιών, και των καταστάσεων ενός τέτοιου συστήματος. Παρουσιάζονται οι λειτουργίες του συστήματος, οι ειδικές απαιτήσεις και τα κριτήρια επικύρωσης/αποδοχής του προγράμματος. Εδώ δίνεται μια περίληψη στο τι θα κάνει το πληροφοριακό σύστημα.

Στην επόμενη διαδικασία που είναι ο σχεδιασμός συστήματος γίνεται αναφορά στην δομή του συστήματος. Επιλέγεται ο εξοπλισμός (υλικό / λογισμικό), εκτελούνται οι απαιτούμενες διαδικασίες και συλλέγονται οι προδιαγραφές δοκιμών ελέγχου. Εδώ γίνεται μια εκτενής ανάλυση της λειτουργίας της εφαρμογής και αναφορά στις τεχνικές προδιαγραφές του υλικού / λογισμικού.

Στο στάδιο της υλοποίησης γίνεται έλεγχος στην εφικτότατα της λύσης που έχει επιλεγεί. Παρουσιάζονται εναλλακτικοί τρόποι υλοποίησης του συστήματος καθώς γίνεται και η ανάλυση κόστους και οφέλους. Εδώ γίνεται η τεκμηρίωση του υλικού / λογισμικού και των διαδικασιών που έχουν επιλεχθεί.

Στο επόμενο στάδιο της εγκατάστασης γίνεται ο έλεγχος της λειτουργίας του συστήματος. Δημιουργία εγχειρίδιων χρήσης του συστήματος και οδηγίες παραλαβής του.

Τελευταίο στάδιο είναι η λειτουργία και η συντήρηση όπου εκεί γίνονται ενέργειες για την ομαλή λειτουργία του συστήματος καθώς και συνεχές προσθήκες, αλλαγές και βελτιώσεις του προγράμματος.

5 Ανάπτυξη Εφαρμογής

Οι πραγματικές διαδικασίες λογισμικού είναι αλληλένδετες αλληλουχίες τεχνικών συνεργατικών και διαχειριστικών δραστηριοτήτων με γενικό στόχο τον προσδιορισμό, το σχεδιασμό, την υλοποίηση και τη δοκιμή ενός συστήματος λογισμικού. Οι προγραμματιστές λογισμικού χρησιμοποιούν μια ποικιλία διαφορετικών λογισμικών εργαλείων στη δουλεία τους. Τα εργαλεία είναι ιδιαίτερα χρήσιμα για την υποστήριξη της ε[εξεργασίας διαφορετικών τύπων εγγράφων και για τη διαχείριση του τεράστιου όγκου λεπτομερών πληροφοριών και παράγονται σε ένα μεγάλο έργο λογισμικού.

Οι τέσσερις βασικές διεργασίες των προδιαγραφών, της ανάπτυξης, της επικύρωσης και της εξέλιξης οργανώνονται διαφορετικά με διαφορετικές διαδικασίες ανάπτυξης. Στο μοντέλο του καταρράκτη, οργανώνονται με σειρά, ενώ στην σταδιακή ανάπτυξη είναι παρεμβαλλόμενα. Ο τρόπος διεξαγωγής αυτών των δραστηριοτήτων εξαρτάται από τον τύπο του λογισμικού, τους ανθρώπους και τις σχετικές οργανωτικές δομές. Στον ακραίο προγραμματισμό για παράδειγμα, οι προδιαγραφές είναι γραμμένες σε κάρτες. Οι δοκιμές είναι εκτελέσιμες και αναπτύχθηκαν πριν από το πρόγραμμα. Η εξέλιξη μπορεί να περιλαμβάνει ουσιαστική αναδιάρθρωση του συστήματος.

Σύμφωνα με τα πρότυπα IEEE 1074-1995 καθορίζονται κάποιες βασικές διαδικασίες κατά την διάρκεια ανάπτυξης ενός λογισμικού. Αυτές είναι οι εξής:

- Ανάλυση Απαιτήσεων
- Σχεδιασμός
- Υλοποίηση
- Επικύρωση
- Συντήρηση

Η ανάλυση απαιτήσεων, η προδιαγραφή του λογισμικού ή η μηχανική των απαιτήσεων είναι η διαδικασία κατανόησης και καθορισμού για το ποιες υπηρεσίες απαιτεί το σύστημα και προσδιορίζει τους περιορισμούς σχετικά με την λειτουργία και την ανάπτυξη του συστήματος. Ο σχεδιασμός απαιτήσεων είναι ένα ιδιαίτερα κρίσιμο στάδιο της πορείας ανάπτυξης του λογισμικού καθώς τα λάθη σε αυτό το στάδιο αναπόφευκτα οδηγούν σε μεταγενέστερα προβλήματα πάνω στο σχεδιασμό και την υλοποίηση του συστήματος.

Η διαδικασία σχεδιασμού απαιτήσεων στοχεύει στην παραγωγή ενός συμφωνηθέντος που καθορίζει ένα σύστημα που ικανοποιεί τις απαιτήσεις των ενδιαφερόμενων. Οι απαιτήσεις παρουσιάζονται συνήθως σε δύο λεπτομερή επίπεδα. Τελικοί χρήστες και πελάτες χρειάζονται μια υψηλού επιπέδου δήλωση των απαιτήσεων. Οι προγραμματιστές του συστήματος χρειάζονται περισσότερο λεπτομερείς προδιαγραφές του συστήματος.

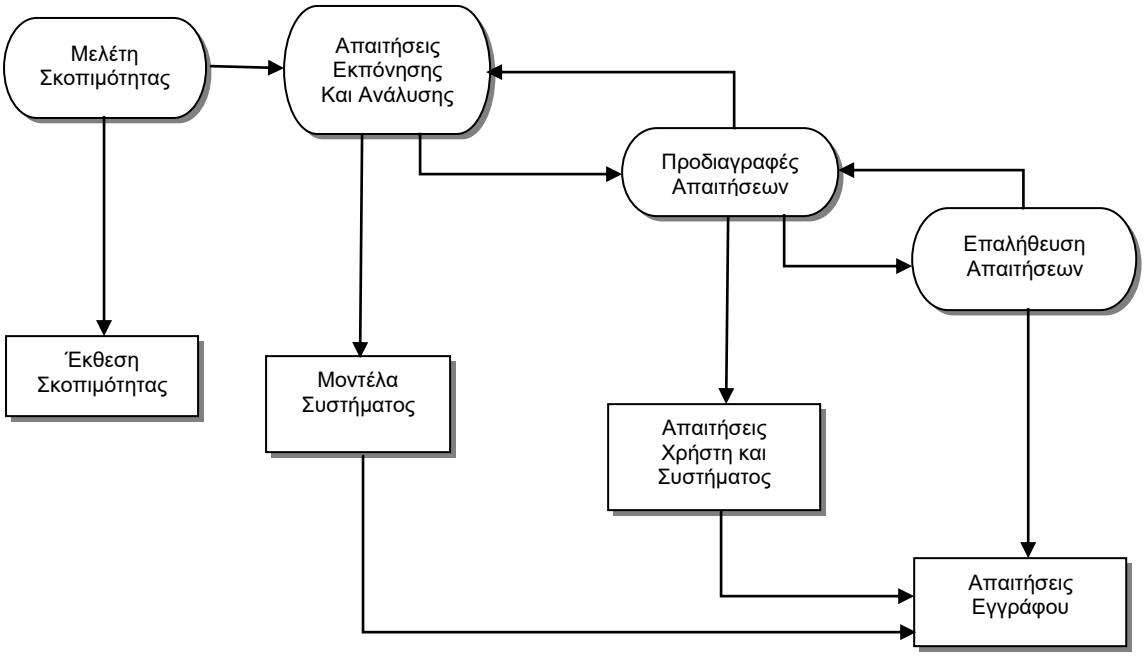
Υπάρχουν τέσσερις κύριες δραστηριότητες στη διαδικασία σχεδιασμού απαιτήσεων:

1. Μελέτη σκοπιμότητας: Εκτιμάται το κατά πόσον οι ανάγκες των αναγνωρισμένων χρηστών μπορεί να είναι ικανοποιημένοι χρησιμοποιώντας τις τρέχουσες τεχνολογίες λογισμικού και υλικού. Η μελέτη θεωρεί εάν το προτεινόμενο σύστημα θα είναι οικονομικά αποδοτικό από επιχειρηματική άποψη και εάν μπορεί να αναπτυχθεί εντός των υφιστάμενων δημοσιονομικών περιορισμών. Μια μελέτη σκοπιμότητας θα πρέπει να είναι σχετικά φτηνή και γρήγορη. Το αποτέλεσμα θα πρέπει να ενημερώνει την απόφαση για το αν πρέπει ή όχι να προχωρήσουμε σε μια πιο λεπτομερή ανάλυση.
2. Ανάκτηση και ανάλυση απαιτήσεων: Αυτή είναι η διαδικασία εξαγωγής απαιτήσεων του συστήματος μέσω της παρατήρησης των υφιστάμενων συστημάτων, συζητήσεις με τις δυνατότητες των χρηστών και των προμηθευτών, την ανάλυση εργασιών και ούτω καθεξής. Αυτό μπορεί να συνεπάγεται στην ανάπτυξη ενός ή περισσότερων μοντέλων

συστημάτων και προτύπων. Αυτές βοηθούν στην κατανόηση του συστήματος που θα καθοριστεί.

3. Προδιαγραφές απαιτήσεων: Η προδιαγραφή απαιτήσεων είναι η δραστηριότητα της μετάφρασης των πληροφοριών που συγκεντρώθηκαν κατά την διάρκεια της δραστηριότητας της ανάλυσης σε ένα έγγραφο που καθορίζει ένα σύνολο απαιτήσεων. Δύο τύποι απαιτήσεων μπορεί να περιλαμβάνονται σε αυτό το έγγραφο. Οι απαιτήσεις των χρηστών είναι αφηρημένες δηλώσεις των απαιτήσεων του συστήματος για τον πελάτη και τον τελικό χρήστη του συστήματος. Οι απαιτήσεις συστήματος είναι η λεπτομερέστερη περιγραφή της λειτουργικότητας που πρέπει να παρέχεται.
4. Έλεγχος απαιτήσεων: Αυτή η δραστηριότητα ελέγχει τις απαιτήσεις για ρεαλισμό, συνέπεια και πληρότητα. Κατά την διάρκεια αυτής της διαδικασίας, τα σφάλματα του εγγράφου απαιτήσεων ανακαλύπτονται αναπόφευκτα. Στη συνέχεια πρέπει να τροποποιηθεί για να διορθώσει αυτά τα προβλήματα.

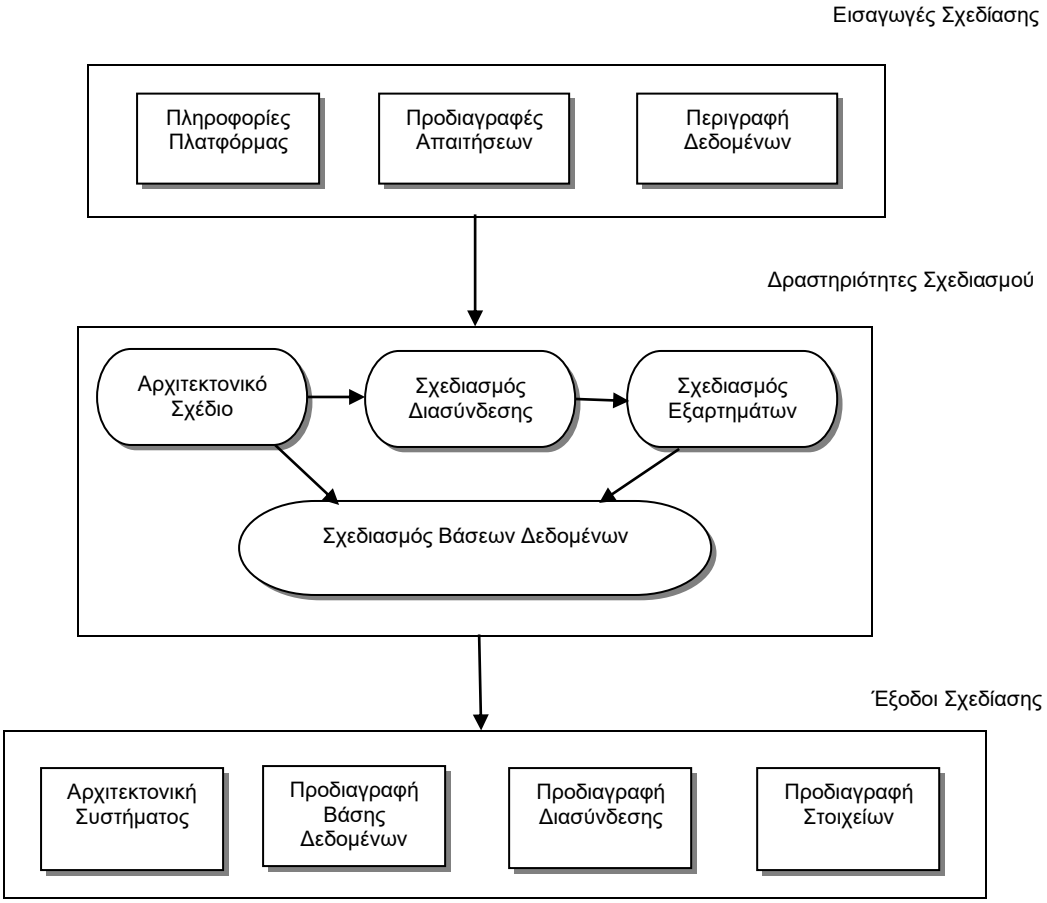
Φυσικά, οι δραστηριότητες στη διαδικασία των απαιτήσεων δεν εκτελούνται απλά σε μια αυστηρή σειρά. Η ανάλυση των αναγκών συνεχίζεται κατά την διάρκεια του ορισμού και των προδιαγραφών και νέες απαιτήσεις έρχονται στο φως καθ' όλη την διαδικασία. Ως εκ τούτου, οι δραστηριότητες της ανάλυσης, του ορισμού και των προδιαγραφών αλληλοσυνδέονται. Σε εύστροφες μεθόδους, όπως ακραίου προγραμματισμού, οι απαιτήσεις αναπτύσσονται σταδιακά σύμφωνα με τις προτεραιότητες των χρηστών και η εκπόνηση των απαιτήσεων προέρχεται από χρήστες που είναι μέλη της ομάδας ανάπτυξης.



Σχήμα 5.1 Η διαδικασία σχεδιασμού απαιτήσεων

Το στάδιο υλοποίησης της ανάπτυξης λογισμικού είναι η διαδικασία μετατροπής ενός συστήματος με προδιαγραφές σε ένα εκτελέσιμο σύστημα. Πάντα συμπεριλαμβάνει διαδικασίες σχεδιασμού και προγραμματισμού του λογισμικού, αλλά αν υπάρχει μια βαθμιαία προσέγγιση στην ανάπτυξη χρησιμοποιούνται, που μπορεί επίσης αυτό να συνεπάγεται στη βελτίωση των προδιαγραφών του λογισμικού.

Ο σχεδιασμός λογισμικού είναι μια περιγραφή της δομής του λογισμικού που πρόκειται να εφαρμοστεί, τα μοντέλα δεδομένων και τις δομές που χρησιμοποιούνται από το σύστημα, τις διεπαφές μεταξύ των στοιχείων του συστήματος και μερικές φορές τους αλγόριθμους που χρησιμοποιούνται. Οι σχεδιαστές δεν φθάνουν στο τέλος, σχεδιάζουν άμεσα άλλα αναπτύσσουν το σχέδιο με τρόπο επαναληπτικό. Προσθέτουν διατυπώσεις και λεπτομέρειες όπως αναπτύσσουν το σχεδιασμό τους με συνεχή αναστροφή για να διορθώσουν προηγούμενα σχέδια.



Σχήμα 5.2 Ένα γενικό μοντέλο της διαδικασίας σχεδιασμού

Στο παραπάνω σχήμα φαίνεται ένα αφηρημένο μοντέλο αυτής της διαδικασίας δείχνοντας τις εισροές της διαδικασίας σχεδιασμού, τις διαδικασίες επεξεργασίας και τα έγγραφα που παράγονται ως αποτέλεσμα αυτής της διαδικασίας.

Το διάγραμμα υποδεικνύει ότι τα στάδια της διαδικασίας σχεδιασμού είναι διαδοχικά. Στην πραγματικότητα, οι διεργασίες σχεδιασμού διεμπλέκονταί. Η ανατροφοδότηση από το ένα στάδιο στο άλλο και η επακόλουθη σχεδίαση είναι αναπόφευκτη σε όλες τις διαδικασίες σχεδιασμού.

Οι περισσότερες διασυνδέσεις λογισμικού με άλλα συστήματα λογισμικού. Αυτές περιλαμβάνουν το λειτουργικό σύστημα, τη βάση δεδομένων, το μεσαίο λογισμικό και άλλα συστήματα εφαρμογών. Αυτά αποτελούν τη “πλατφόρμα λογισμικού”, το περιβάλλον στο οποίο θα εκτελεστεί το λογισμικό. Η πληροφορία για αυτή την πλατφόρμα αποτελεί βασική συνεισφορά στη διαδικασία σχεδιασμού, καθώς οι σχεδιαστές πρέπει να αποφασίσουν ποιος είναι καλύτερος τρόπος να το ενσωματώσουν στο περιβάλλον του λογισμικού. Η προδιαγραφή των απαιτήσεων είναι η περιγραφή της λειτουργικότητας που πρέπει να παρέχει το λογισμικό, τις επιδώσεις του και τις απαιτήσεις αξιοπιστίας του. Εάν το σύστημα πρόκειται να επεξεργαστεί υπάρχοντα δεδομένα, τότε η περιγραφή των δεδομένων αυτών μπορεί να περιλαμβάνεται στην προδιαγραφή της πλατφόρμας. Διαφορετικά, η περιγραφή των δεδομένων πρέπει να είναι μια εισαγωγή στη διαδικασία σχεδιασμού έτσι να οριστεί η οργάνωση των δεδομένων του συστήματος.

Οι δραστηριότητες στη διαδικασία σχεδιασμού ποικίλλουν, ανάλογα με τον τύπο του συστήματος που αναπτύσσεται. Για παράδειγμα τα συστήματα πραγματικού χρόνου απαιτούν σχεδιασμό χρονισμού, αλλά δεν μπορούν να περιλαμβάνουν μια βάση δεδομένων, ώστε να μην υπάρχει σχεδίαση βάσης δεδομένων. Το παραπάνω σχήμα δείχνει τέσσερις δραστηριότητες που μπορεί να είναι μέρος της διαδικασίας σχεδιασμού για συστήματα πληροφοριών:

1. Αρχιτεκτονικός σχεδιασμός, όπου εκεί προσδιορίζετε η συνολική δομή του συστήματος, τα βασικά στοιχεία (μερικές φορές αποκαλούμενα υποσυστήματα ή ενότητες), τις σχέσεις τους και τον τρόπο κατανομή τους.
2. Σχεδιασμός διεπαφών, όπου εκεί ορίζονται οι διεπαφές μεταξύ των στοιχείων του συστήματος. Αυτή η προδιαγραφή διεπαφής πρέπει να είναι ξεκάθαρη. Με μια ακριβής διεπαφή, ένα στοιχείο μπορεί να χρησιμοποιηθεί χωρίς άλλα στοιχεία που πρέπει να γνωρίζουν πως αυτό εφαρμόζεται. Αφού συμφωνηθούν οι προδιαγραφές διεπαφών, τα στοιχεία μπορούν να σχεδιαστούν και να αναπτυχθούν ταυτόχρονα.
3. Σχεδιασμός εξαρτήματος, όπου λαμβάνετε κάθε στοιχείο του συστήματος και σχεδιάζετε ο τρόπος λειτουργίας του. Αυτό μπορεί να είναι μια απλή κατάσταση της αναμενόμενης λειτουργικότητας που θα εφαρμοστεί, με το συγκεκριμένο σχέδιο να αφεθεί στον προγραμματιστή. Εναλλακτικά, μπορεί να είναι μια λίστα από αλλαγές που πρέπει να γίνουν σε ένα επαναχρησιμοποιήσιμο στοιχείο ή σε ένα λεπτομερές μοντέλο σχεδιασμού. Το μοντέλο σχεδιασμού μπορεί να χρησιμοποιηθεί για την αυτόματη δημιουργία μιας υλοποίησης.
4. Σχεδιασμός βάσης δεδομένων, όπου σχεδιάζονται οι δομές δεδομένων του συστήματος και πως αυτές πρέπει να παριστάνονται σε μια βάση δεδομένων. Και πάλι, η εργασία εξαρτάται από το εάν πρόκειται να επαναχρησιμοποιηθεί μια βάση δεδομένων ή να δημιουργηθεί μια νέα βάση δεδομένων.

Αυτές οι δραστηριότητες οδηγούν σε ένα σύνολο εξόδων σχεδιασμού, τα οποία φαίνονται στο παραπάνω σχήμα. Η λεπτομέρεια και η αναπαράσταση αυτών διαφέρουν σημαντικά. Για κρίσιμα συστήματα, πρέπει να παραχθούν λεπτομερή έγγραφα σχεδιασμού που να καθορίζουν ακριβώς και ακριβείς περιγραφές του συστήματος. Εάν χρησιμοποιείται μια προσέγγιση με γνώμονα το μοντέλο, αυτές οι έξοδοι μπορεί να είναι κατά βάση διαγράμματα. Όταν χρησιμοποιούνται ευέλικτες μέθοδοι ανάπτυξης, τα αποτελέσματα της διαδικασίας σχεδιασμού μπορεί να μην είναι χωριστά έγγραφα προδιαγραφών, αλλά να μπορούν να αναπαρασταθούν στον κώδικα του προγράμματος.

Οι δομημένες μέθοδοι σχεδιασμού αναπτύχθηκαν τη δεκαετία του 1970 και του 1980 και αποτέλεσαν πρόδρομο του UML και του αντικειμενοστραφούς σχεδίου. Βασίζονται στην παραγωγή γραφικών μοντέλων του συστήματος και σε πολλές περιπτώσεις, στην αυτόματη παραγωγή κώδικα από αυτά τα μοντέλα. Η μοντελοποιημένη ανάπτυξη ή η μοντελοποιημένη μηχανική όπου μοντέλα του λογισμικού που δημιουργούνται σε διαφορετικά επίπεδα αφαίρεσης, είναι μία εξέλιξη δομημένων μεθόδων. Στην μοντελοποιημένη ανάπτυξη, δίνεται μεγαλύτερη έμφαση στα αρχιτεκτονικά μοντέλα με διαχωρισμό μεταξύ αφηρημένων μοντέλων ανεξάρτητα από την εφαρμογή και μοντέλα συγκεκριμένης υλοποίησης. Τα μοντέλα αναπτύσσονται με αρκετή λεπτομέρεια ώστε το εκτελέσιμο σύστημα να μπορεί να δημιουργηθεί από αυτά.

Η ανάπτυξη ενός προγράμματος για την υλοποίηση του συστήματος ακολουθεί φυσικά από τις διαδικασίες σχεδιασμού του συστήματος. Αν και μερικές τάξεις του προγράμματος, όπως η κρίσιμη για την ασφάλεια του συστήματος, συνήθως σχεδιάζονται λεπτομερώς πριν ξεκινήσει οποιαδήποτε εφαρμογή, είναι συνηθέστερο για τα μεταγενέστερα στάδια του σχεδιασμού και της ανάπτυξης του προγράμματος να παρεμβάλλονται. Τα εργαλεία ανάπτυξης λογισμικού μπορούν να χρησιμοποιηθούν για τη δημιουργία του σκελετού του προγράμματος από ένα σχέδιο. Αυτό περιλαμβάνει τον κώδικα για τον ορισμό και την υλοποίηση διεπαφών και σε πολλές περιπτώσεις ο προγραμματιστής χρειάζεται μόνο να προσθέσει λεπτομέρειες σχετικά με τη λειτουργία κάθε συστατικού του προγράμματος.

Ο προγραμματισμός είναι μια προσωπική δραστηριότητα και δεν υπάρχει γενική διαδικασία που συνήθως ακολουθείται. Μερικοί προγραμματιστές αρχίζουν με συστατικά που κατανοούν, αναπτύσσουν αυτά, και στη συνέχεια προχωρούν σε λιγότερο κατανοητά συστατικά. Άλλοι υιοθετούν την αντίθετη προσέγγιση, αφήνοντας τα γνωστά συστατικά μέχρι το τέλος επειδή γνωρίζουν πως να τα αναπτύξουν. Μερικοί προγραμματιστές επιθυμούν να ορίσουν τα δεδομένα νωρίς στη διαδικασία, στη συνέχεια, να τα χρησιμοποιήσουν για να οδηγήσουν την ανάπτυξη του προγράμματος, άλλοι αφήνουν τα δεδομένα απροσδιόριστα όσο το δυνατόν περισσότερο.

Κανονικά οι προγραμματιστές πραγματοποιούν ορισμένες δοκιμές του κώδικα που έχουν αναπτύξει. Αυτό συχνά αποκαλύπτει ελαττώματα του προγράμματος που πρέπει να αφαιρεθούν από το πρόγραμμα. Αυτό ονομάζεται εντοπισμός σφαλμάτων (debugging). Οι δοκιμές βλαβών και η αποσφαλμάτωση είναι διαφορετικές διαδικασίες. Η δοκιμή διαπιστώνει την ύπαρξη ελαττωμάτων. Η αποσφαλμάτωση αφορά τον εντοπισμό και τη διόρθωση αυτών των ελλατωμάτων.

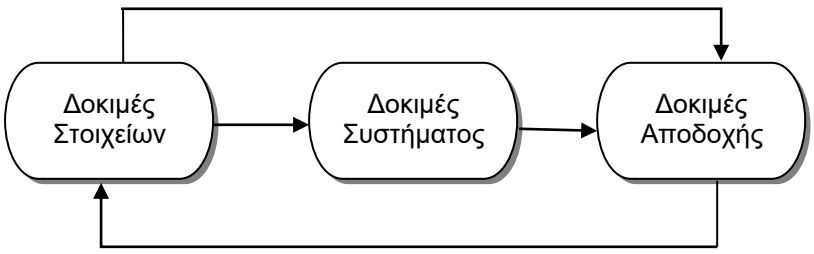
Όταν γίνεται αποσφαλμάτωση, πρέπει να δημιουργούνται υποθέσεις σχετικά με την παρατηρούμενη συμπεριφορά του προγράμματος, στη συνέχεια να δοκιμάζονται αυτές οι υποθέσεις με την ελπίδα να βρεθεί το σφάλμα που προκάλεσε την ανωμαλία εξόδου. Η εξέταση των υποθέσεων μπορεί να περιλαμβάνει τον εντοπισμό του κώδικα του προγράμματος με μη αυτόματο τρόπο. Μπορεί να απαιτούνται νέες περιπτώσεις δοκιμής για να εντοπιστεί το πρόβλημα. Διαδραστικά εργαλεία εντοπισμού σφαλμάτων, τα οποία δείχνουν τις ενδιάμεσες τιμές των μεταβλητών του προγράμματος και μια σειρά καταστάσεων που έχουν εκτελεσθεί, μπορούν να χρησιμοποιηθούν για να υποστηρίξουν τη διαδικασία εντοπισμού σφαλμάτων.

Η επικύρωση του λογισμικού ή γενικότερα, η επαλήθευση και η επικύρωση (V & V) αποσκοπεί στο να αποδείξει ότι ένα σύστημα ανταποκρίνεται στις προδιαγραφές του και ότι ανταποκρίνεται στις προσδοκίες του πελάτη του συστήματος. Δοκιμή προγράμματος, όπου εκτελείται το σύστημα χρησιμοποιώντας προσομοιωμένα δεδομένα δοκιμών, είναι η κύρια τεχνική επικύρωσης. Η επικύρωση μπορεί επίσης να περιλαμβάνει διαδικασίες ελέγχου, όπως επιθεωρήσεις και ανασκοπήσεις, σε κάθε στάδιο της διαδικασίας λογισμικού, από τον ορισμό των απαιτήσεων των χρηστών μέχρι την ανάπτυξη του προγράμματος. Λόγω της υπεροχής των δοκιμών, το μεγαλύτερο μέρος του κόστους επικύρωσης πραγματοποιείται κατά τη διάρκεια και μετά την εκτέλεση.

Εκτός από τα μικρά προγράμματα, τα συστήματα δεν θα πρέπει να ελέγχονται ως ενιαία, μονολιθική μονάδα. Το σχήμα 5.3 παρουσιάζει μια διαδικασία δοκιμής τριών σταδίων, στην οποία δοκιμάζονται τα συστατικά του συστήματος, στη συνέχεια δοκιμάζεται το ολοκληρωμένο σύστημα και τελικά το σύστημα δοκιμάζεται με τα δεδομένα του πελάτη. Στην ιδανική περίπτωση, τα ελαττώματα των συστατικών ανακαλύπτονται νωρίς στη διαδικασία και τα προβλήματα διεπαφής εντοπίζονται όταν ολοκληρωθεί το σύστημα. Ωστόσο, καθώς εντοπίζονται ελαττώματα, το πρόγραμμα πρέπει να διορθωθεί και αυτό μπορεί να απαιτήσει την επανάληψη άλλων σταδίων της διαδικασίας δοκιμής. Σφάλματα στα στοιχεία του προγράμματος, για παράδειγμα, μπορεί να έρθουν στο φως κατά τη διάρκεια της δοκιμής του συστήματος. Η διαδικασία είναι επομένως επαναληπτική με πληροφορίες που τροφοδοτούνται από τα μεταγενέστερα στάδια σε προηγούμενα μέρη της διαδικασίας.

Τα στάδια της διαδικασίας δοκιμής είναι τα εξής:

1. Δοκιμές Ανάπτυξης. Τα συστατικά που αποτελούν το σύστημα ελέγχονται από τα άτομα που αναπτύσσουν το σύστημα. Κάθε στοιχείο δοκιμάζεται ανεξάρτητα, χωρίς άλλα στοιχεία του συστήματος. Τα στοιχεία μπορεί να είναι απλές οντότητες ‘όπως λειτουργίες ή κλάσεις αντικειμένων ή μπορεί να είναι συνεκτικές ομαδοποιήσεις αυτών των οντοτήτων. Χρησιμοποιούνται συνήθως εργαλεία αυτοματοποίησης δοκιμών, που μπορούν να ξαναχρησιμοποιούν δοκιμές συστατικών στοιχείων και όταν δημιουργούνται νέες εκδόσεις του.
2. Δοκιμές Συστήματος. Τα στοιχεία του συστήματος είναι ενσωματωμένα για τη δημιουργία ενός πλήρους συστήματος. Αυτή η διαδικασία αφορά την εύρεση σφαλμάτων που προκύπτουν από απρόβλεπτες αλληλεπιδράσεις μεταξύ των συστατικών στοιχείων και των στοιχείων με προβλήματα διεπαφής. Αφορά επίσης την απόδειξη ότι το σύστημα πληροί τις λειτουργικές και μη λειτουργικές του απαιτήσεις και ελέγχει τις ιδιότητες του αναδυόμενου συστήματος. Για τα μεγάλα συστήματα, αυτή μπορεί να είναι μια διαδικασία πολλαπλών σταδίων όπου τα στοιχεία είναι ενσωματωμένα για να σχηματίζουν υποσυστήματα που εξετάζονται μεμονωμένα πριν τα εν λόγω υποσυστήματα ενσωματωθούν για να διαμορφώσουν το τελικό σύστημα.
3. Δοκιμές Αποδοχής. Αυτό είναι το τελικό στάδιο της διαδικασίας ελέγχου πριν γίνει αποδεκτό το σύστημα για επιχειρησιακή χρήση. Το σύστημα δοκιμάζεται με τα δεδομένα που παρέχει το σύστημα του πελάτη και όχι με προσομοιωμένα δεδομένα δοκιμών. Οι δοκιμές αποδοχής ενδέχεται να αποκαλύψουν σφάλματα και παραλείψεις στον ορισμό των απαιτήσεων συστήματος, επειδή τα πραγματικά δεδομένα ασκούν το σύστημα με διαφορετικούς τρόπους από τα δεδομένα δοκιμών. Οι δοκιμές αποδοχής ενδέχεται επίσης να αποκαλύψουν προβλήματα στα οποία οι ευκολίες του συστήματος δεν ανταποκρίνονται στις ανάγκες του χρήστη ή η απόδοση του συστήματος είναι απαράδεκτη.



Σχήμα 5.3 Στάδια Δοκιμών

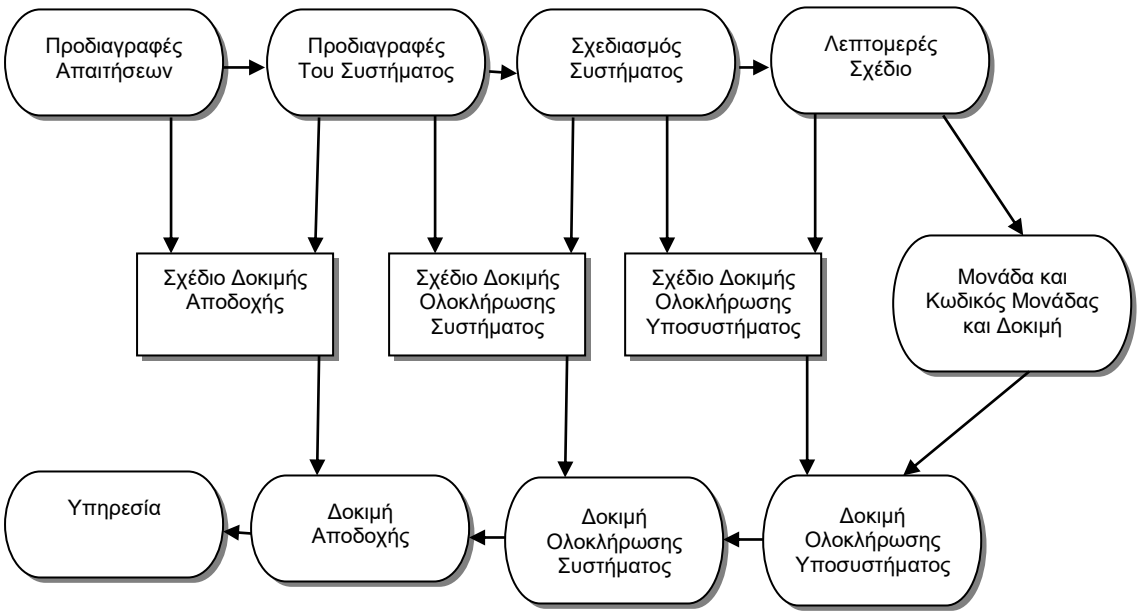
Κανονικά, οι διεργασίες ανάπτυξης και δοκιμής εξαρτημάτων διεμπλέκονται. Οι προγραμματιστές συνθέτουν τα δικά τους δεδομένα δοκιμών και δοκιμάζουν διαδοχικά τον κώδικα όπως έχει αναπτυχθεί. Πρόκειται για μια οικονομικά λογική προσέγγιση, καθώς ο προγραμματιστής γνωρίζει το στοιχείο και ως εκ τούτου είναι το καλύτερο άτομο για τη δημιουργία δοκιμαστικών περιπτώσεων.

Εάν χρησιμοποιείται μια βαθμιαία προσέγγιση ανάπτυξης, κάθε αύξηση θα πρέπει να δοκιμάζεται όπως έχει αναπτυχθεί, με αυτές τις δοκιμές να βασίζονται στις απαιτήσεις για αυτή την αύξηση. Στον ακραίο προγραμματισμό, αναπτύσσονται δοκιμές μαζί με τις απαιτήσεις πριν ξεκινήσει η ανάπτυξη. Αυτό βοηθά τους δοκιμαστές και τους προγραμματιστές να κατανοήσουν τις απαιτήσεις και διασφαλίζει ότι δεν υπάρχουν καθυστερήσεις στην δημιουργία δοκιμαστικών περιπτώσεων.

Όταν χρησιμοποιείται μια προγραμματισμένη διαδικασία λογισμικού (π.χ. για την ανάπτυξη κρίσιμων συστημάτων), η δοκιμή καθοδηγείται από ένα σύνολο σχεδίων δοκιμών. Μια ανεξάρτητη ομάδα δοκιμαστών λειτουργεί από αυτά τα προκατασκευασμένα σχέδια δοκιμών, τα οποία έχουν αναπτυχθεί από την προδιαγραφή και το σχεδιασμό του συστήματος. Το σχήμα 5.4 δείχνει πως τα σχέδια δοκιμών είναι η σχέση μεταξύ των δοκιμών και των δραστηριοτήτων ανάπτυξης. Αυτό καλείται μερικές φορές το V-μοντέλο ανάπτυξης.

Ο έλεγχος αποδοχής ονομάζεται μερικές φορές ως «δοκιμή άλφα». Τα προσαρμοσμένα συστήματα αναπτύσσονται για έναν μόνο πελάτη. Η διαδικασία της δοκιμής άλφα συνεχίζεται μέχρι ο προγραμματιστής του συστήματος και ο πελάτης συμφωνούν ότι το παραδοθέν σύστημα είναι μια αποδεκτή εφαρμογή των απαιτήσεων.

Όταν ένα σύστημα πρόκειται να διατεθεί στο εμπόριο ως προϊόν λογισμικού, χρησιμοποιείται συχνά μια διαδικασία δοκιμής που ονομάζεται «δοκιμή beta». Η δοκιμή beta περιλαμβάνει την παροχή ενός συστήματος σε έναν αριθμό πιθανών πελατών που συμφωνούν να χρησιμοποιήσουν αυτό το σύστημα. Αναφέρουν προβλήματα στους προγραμματιστές του συστήματος. Αυτό εκθέτει το προϊόν σε πραγματική χρήση και ανιχνεύει σφάλματα που ενδέχεται να μην είχαν προβλεφθεί από τους κατασκευαστές συστημάτων. Μετά από αυτήν την ανατροφοδότηση, το σύστημα τροποποιείται και κυκλοφορεί είτε για περαιτέρω δοκιμές beta είτε για γενική πώληση.



Σχήμα 5.4 Δοκιμαστικές φάσεις σε μια διαδικασία λογισμικού με γνώμονα το σχέδιο

Η ευελιξία των συστημάτων λογισμικού είναι ένας από τους κύριους λόγους για τους οποίους όλο και περισσότερα λογισμικά ενσωματώνονται σε μεγάλα, σύνθετα συστήματα. Μόλις ληφθεί απόφαση για την κατασκευή υλικού, είναι πολύ ακριβό να γίνουν αλλαγές στον σχεδιασμό υλικού. Ωστόσο, μπορεί να γίνουν αλλαγές στο λογισμικό οποιαδήποτε στιγμή κατά τη διάρκεια ή μετά την ανάπτυξη του συστήματος. Ακόμη και εκτεταμένες αλλαγές εξακολουθούν να είναι πολύ φθηνότερες από τις αντίστοιχες αλλαγές στο υλικό του συστήματος.

Ιστορικά, υπήρχε πάντα ένας διαχωρισμός μεταξύ της διαδικασίας ανάπτυξης λογισμικού και της διαδικασίας εξέλιξης λογισμικού.(συντήρηση λογισμικού). Οι άνθρωποι σκέφτονται την ανάπτυξη λογισμικού ως μια δημιουργική δραστηριότητα στην οποία αναπτύσσεται ένα σύστημα λογισμικού από μια αρχική ιδέα έως ένα λειτουργικό σύστημα. Ωστόσο, μερικές φορές σκέφτονται τη συντήρηση λογισμικού ως ανιαρή και αδιάφορη. Αν και το κόστος συντήρησης είναι συχνά αρκετές φορές το αρχικό κόστος ανάπτυξης, οι διαδικασίες συντήρησης μερικές φορές θεωρούνται λιγότερο προκλητικές από την αρχική ανάπτυξη λογισμικού.

Αυτή η διάκριση μεταξύ ανάπτυξης και συντήρησης είναι όλο και πιο άσχετη. Σχεδόν όλα τα συστήματα λογισμικού δεν είναι εντελώς νέα συστήματα και καθίσταται πολύ πιο λογικό να δούμε την ανάπτυξη και τη συντήρηση σαν συνέχεια. Αντί για δύο διαφορετικές διαδικασίες, είναι πιο ρεαλιστικό να σκεφτόμαστε την τεχνολογία λογισμικού ως εξελικτική διαδικασία κατά την οποία το λογισμικό μεταβάλλεται συνεχώς καθ' όλη τη διάρκεια ζωής του, ανταποκρινόμενη στις μεταβαλλόμενες απαιτήσεις και τις ανάγκες των πελατών.

Οι τρεις βασικές διεργασίες που με μελετήθηκαν εκτενώς και πάνω σε αυτές στηρίχθηκε η δημιουργία του πληροφοριακού μας συστήματος είναι οι παρακάτω.

Στο στάδιο της ανάλυσης απαιτήσεων, ορίζεται το τι πρέπει να κάνει ένα σύστημα λογισμικού, και παρέχεται μια μηχανική περιγραφή των αντικειμένων, των λειτουργιών, και των καταστάσεων ενός συστήματος λογισμικού. Κατά τη διάρκεια της διαδικασίας των απαιτήσεων αναπτύσσονται οι προτεραιότητες, η ολοκλήρωση του λογισμικού και οι ανάγκες της διεπαφής, τα μοντέλα ροής δεδομένων πληροφοριών, η λεπτομερής ανάλυση των κινδύνων, και οι δοκιμές και σχέδια εγκατάστασης. Τα επίσημα μοντέλα που εκθέτουν τη δομή των συστημάτων λογισμικού (εισόδους, εξόδους, και συνδέσεις μεταξύ των δραστηριοτήτων) είναι συνήθως ένα αποτέλεσμα της διαδικασίας απαιτήσεων.

Η φάση σχεδιασμού εστιάζει στον τρόπο με τον οποίο οι απαιτήσεις λογισμικού μπορούν να πραγματοποιηθούν. Σε αυτό το στάδιο, οι κύριες ανησυχίες είναι οι λειτουργίες και η δομή του συστήματος που αναπτύσσεται. Αυτό σημαίνει ότι επιλέγονται οι αρχιτεκτονικές λογισμικού, οι συγκεκριμένες διεπαφές, και οι αλγόριθμοι. Οι επιλογές επικυρώνονται ότι ικανοποιούν τους προσδιορισμούς απαιτήσεων. Τα προϊόντα του σχεδιασμού των δραστηριοτήτων πρέπει επίσης να είναι ελεγμένα σχετικά με συγκεκριμένους περιορισμούς που προσδιορίζονται στην αρχή της φάσης σχεδιασμού.

Τέλος, κατά τη διάρκεια της διαδικασίας υλοποίησης, παράγεται ο πηγαίος κώδικας. Ο πηγαίος κώδικας πρέπει να επιβεβαιώνεται ότι ικανοποιεί τους προσδιορισμούς απαιτήσεων, και πρέπει να ελεγχθεί σχετικά με τους περιορισμούς του σχεδιασμού. Τα δεδομένα δοκιμών παράγονται κατά τη διάρκεια της εκτέλεσης του προκύπτοντος πηγαίου κώδικα. Το σύστημα λειτουργίας είναι τεκμηριωμένο (π.χ., αποτελέσματα των δοκιμών, κόστος λογισμικού, μετρήσεις της ποιότητας λογισμικού, εγχειρίδια χρηστών). Στον έλεγχο του κώδικα γίνεται επαλήθευση ότι αναπτύχθηκε σωστά το λογισμικό με βάση αυτά που έχουν καθοριστεί στην ανάλυση και το σχεδιασμό. Επιπλέον γίνεται επικύρωση ότι όντως αναπτύχθηκε το προϊόν σύμφωνα με τις απαιτήσεις του χρήστη.

Στα επόμενα κεφάλαια, θα γίνει εκτενής αναφορά στις παραπάνω φάσεις ανάπτυξης του Πληροφοριακού Συστήματος Ιατρικών Φακέλων Ασθενών

5.1 Ανάλυση Απαιτήσεων

Ως πρώτο στάδιο ανάπτυξης ενός λογισμικού, ορίζεται ο καθορισμός απαιτήσεων του, όπως αναφέρθηκε παραπάνω. Συνεπώς, για το Πληροφοριακό Σύστημα των Ιατρικών Φακέλων Ασθενών, ορίζονται οι παρακάτω απαιτήσεις – παραδοχές:

1) Πιστοποίηση Χρήστη: Ο χρήστης ανοίγοντας την εφαρμογή καταχωρεί όνομα (username) και κωδικό (password) προκειμένου να πιστοποιηθεί η πρόσβαση του στην εφαρμογή. Σε περίπτωση λάθους λαμβάνει μήνυμα σφάλματος και παροτρύνεται να προσπαθήσει ξανά. Σε άλλη περίπτωση καλείται να επικοινωνήσει με τον διαχειριστή της εφαρμογής. Σε περίπτωση επιτυχούς εισόδου εμφανίζεται μήνυμα επιτυχούς εισόδου και μεταφερόμαστε στο κύρια φόρμα της εφαρμογής.
2) Εγγραφή Ασθενούς: Πραγματοποιείται όταν ο ασθενής εισαχθεί στο ιατρείο. Καταχωρούνται τα δημογραφικά στοιχεία του ασθενούς, στοιχεία επικοινωνίας του καθώς και η φωτογραφία του. Στην περίπτωση που δεν συμπληρωθούν όνομα και επίθετο δεν θα γίνεται καταχώρηση στην βάση και θα εμφανίζεται το κατάλληλο μήνυμα στον χρήστη.
3) Πληροφορίες Ασθενούς: Οι χρήστες του συστήματος θα μπορούν να διαχειρίζονται τις εγγραφές των ασθενών. Θα μπορούν να τους αναζητήσουν με βάση το ονοματεπώνυμο τους και να επεξεργαστούν τα δεδομένα τους. Ακόμη θα έχουν την επιλογή της διαγραφής κάποιου ασθενούς. Στην περίπτωση της ανανέωσης των στοιχείων εμφανίζεται μήνυμα ενημέρωσης ενώ στην περίπτωση διαγραφής βγαίνει μήνυμα επιβεβαίωσης αυτής της ενέργειας και δίνει τη δυνατότητα στον χρήστη να την ακυρώσει ή να την συνεχίσει.
4) Καταχώρηση Επισκέψεων: Ο χρήστης θα μπορεί να καταχωρεί τις επισκέψεις των ασθενών του οι οποίες θα περιέχουν ημερομηνία και μια μικρή περιγραφή. Για κάθε επίσκεψη θα έχει την δυνατότητα καταχώρησης διαγνώσεων και για κάθε διάγνωση καταχώρηση φαρμάκων. Επίσης θα μπορεί να επεξεργάζεται όλα αυτά τα στοιχεία και αν θέλει να τα διαγράφει. Σε περίπτωση που πάει να γίνει εισαγωγή διάγνωσης χωρίς να έχει επιλεχθεί κάποιά συγκεκριμένη επίσκεψη το σύστημα δεν τον αφήνει τον χρήστη και του βγάζει το κατάλληλο μήνυμα. Αυτό ισχύει και στην περίπτωση που ο χρήστης προσπαθήσει να κάνει εισαγωγή φαρμάκου χωρίς πριν να έχει επιλέξει διάγνωση.
5) Αναφορά Επίσκεψης: Ο χρήστης θα έχει τη δυνατότητα να βγάζει αναφορά επίσκεψης που θα περιέχει τα στοιχεία του ασθενούς τις αρρώστιες που διαγνώστηκαν καθώς και τα φάρμακα που του δόθηκαν για κάθε διάγνωση.
6) Κληρονομικότητα Ασθενών: Ο χρήστης θα μπορεί να επιλέγει την σχέση κληρονομικότητας μεταξύ των ασθενών του αν αυτή υπάρχει. Θα μπορεί να γίνει επεξεργασία και διαγραφή της σχέσης αυτής.
7) Φωτογραφίες Ασθενών: Το πρόγραμμα θα μπορεί να αποθηκεύει φωτογραφίες για κάθε ασθενή. Κάθε φωτογραφία θα μπορεί να περιέχει και μια σύντομη περιγραφή. Θα υπάρχει η δυνατότητα επεξεργασίας της περιγραφής καθώς επίσης και τη διαγραφή των φωτογραφιών. Κατά την διαγραφή φωτογραφιών ή την επεξεργασία τους εμφανίζονται αντίστοιχα μηνύματα.
8) Προσθήκη Διάγνωσης: Ο χρήστης / γιατρός θα έχει τη δυνατότητα να προσθέσει διαγνώσεις, να τις επεξεργαστεί ή να διαγράψει από τις υπάρχουσες. Σε οποιαδήποτε αλλαγή από της προηγούμενες θα εμφανίζεται μήνυμα επιβεβαίωσης. Επίσης θα υπάρχει και η δυνατότητα γρήγορης αναζήτησης των διαγνώσεων με βάση το όνομα τους.
9) Προσθήκη Φαρμάκων: Ο χρήστης / γιατρός θα έχει τη δυνατότητα να προσθέσει φάρμακα, να τα επεξεργαστεί ή να διαγράψει από τα υπάρχοντα. Σε οποιαδήποτε αλλαγή από τις προηγούμενες θα εμφανίζεται μήνυμα επιβεβαίωσης. Επίσης θα

υπάρχει και η δυνατότητα γρήγορης αναζήτησης των φαρμάκων με βάση το όνομα τους.

10) Στατιστικά Δεδομένα: Το σύστημα θα παρέχει στατιστικά δεδομένα για τους ασθενείς. Στην πρώτη περίπτωση θα μπορεί ο χρήστης να δει σε διάγραμμα το ιστορικό από τίς αρρώστιες που έχει περάσει κάποιος ασθενής καθώς και το πλήθος κάθε μια από αυτές. Στην δεύτερη περίπτωση ο το πρόγραμμα θα εμφανίζει διάγραμμα με τις αρρώστιες και το πλήθος από αυτές που έχουν περάσει οι συγγενείς κάποιου ασθενούς.
11) Δεδομένα Σε Excel: Ο χρήστης θα μπορεί να αποθηκεύσει σε αρχείο excel σε λίστα όλους τους ασθενείς μαζί με τα δημογραφικά τους στοιχεία.

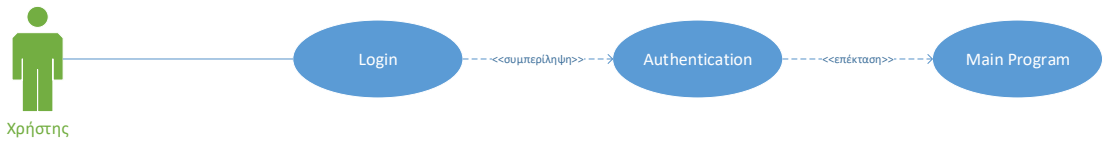
5.2 Σχεδιασμός

5.2.1 Σχεδιασμός με χρήση UML

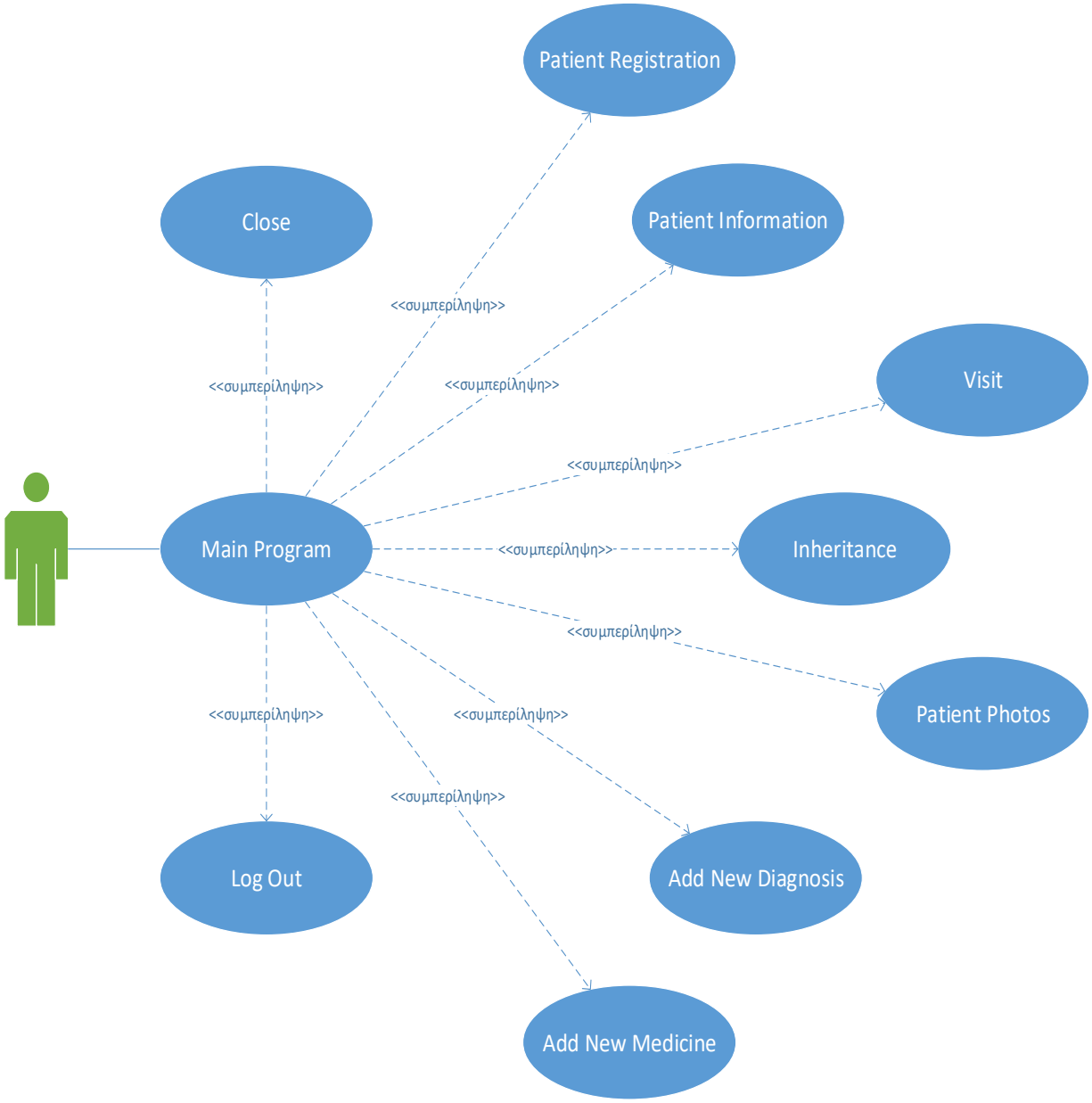
Η Unified Modeling Language (UML) είναι μια γραφική γλώσσα αναπαράστασης αντικειμενοστραφών μοντέλων συστημάτων. Η UML αναπτύχθηκε το 1997 από τους Booch, Rambaugh και Jacobson με σκοπό να λύσει το πρόβλημα των πολλών διαφορετικών αναπαραστάσεων που υπήρχαν μέχρι τότε για τα μοντέλα αντικειμένων. Έκτοτε η UML έχει πράγματι αποτελέσει ένα διεθνές πρότυπο αντικειμενοστραφούς γλώσσας μοντελοποίησης. Η UML είναι γλώσσα μοντελοποίησης και όχι μεθοδολογία. Η μεθοδολογία αποτελείται από μια γλώσσα μοντελοποίησης και μια διαδικασία. Η UML ορίζει μόνο τη γλώσσα μοντελοποίησης και όχι τη διαδικασία. Η γλώσσα μοντελοποίησης βοηθάει στην περιγραφή του σχεδιασμού. Η διαδικασία ορίζει τον τρόπο δημιουργίας του σχεδιασμού.

Διαγράμματα Περιπτώσεων Χρήσης

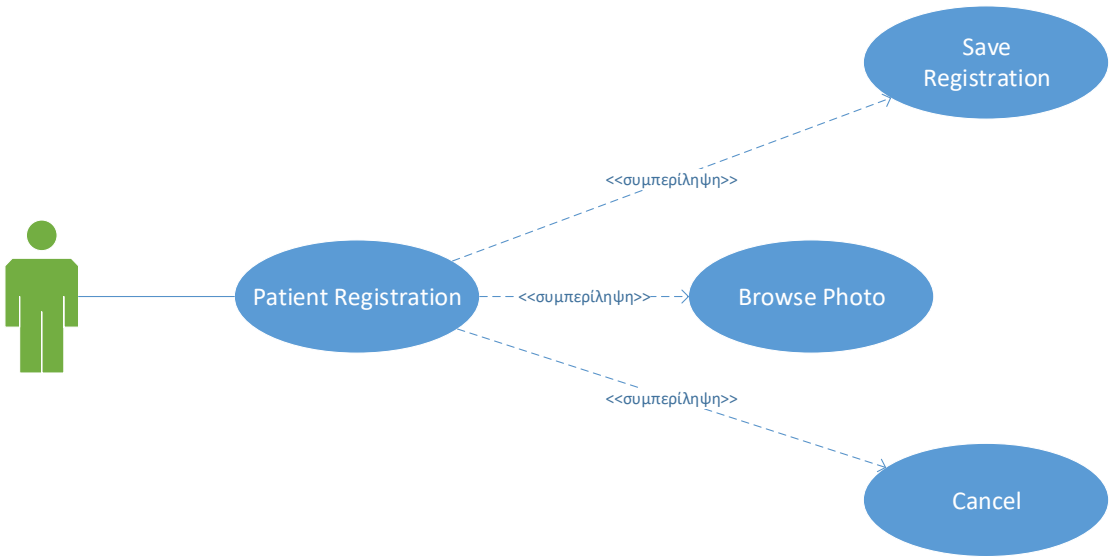
Τα διαγράμματα περιπτώσεων χρήσης περιγράφουν τη συμπεριφορά ενός συστήματος από την απτική γωνία ενός χρήστη. Επιτρέπουν τον ορισμό των ορίων του συστήματος και του περιβάλλοντος.



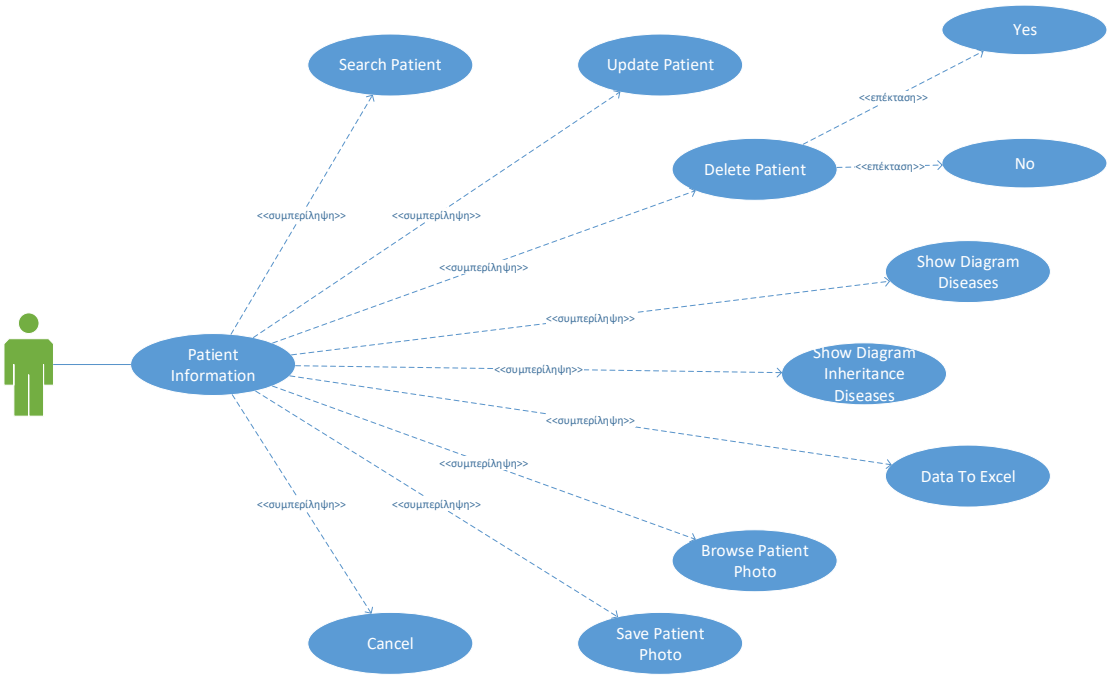
Εικόνα 1. Use Case: Login



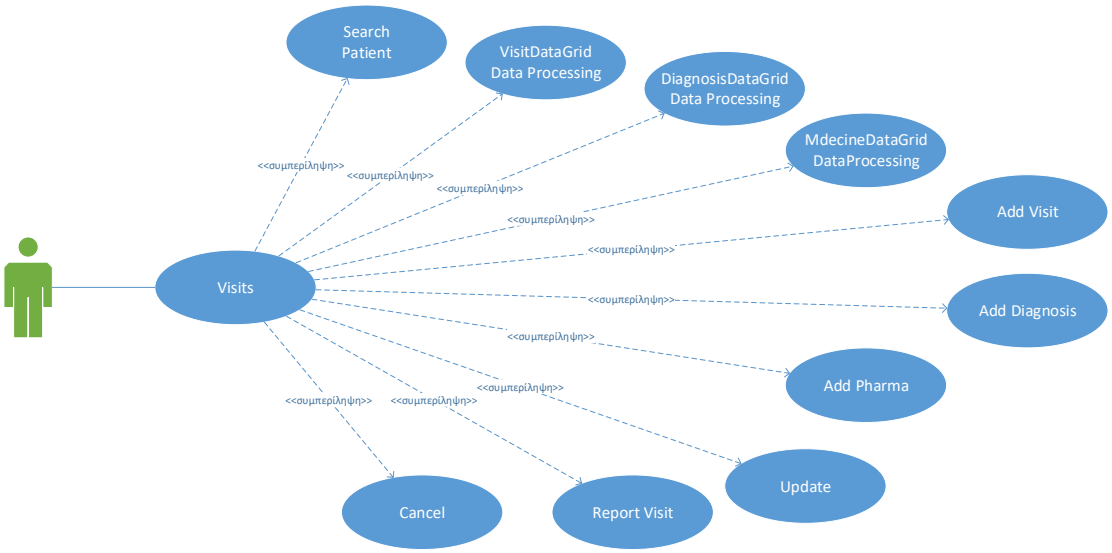
Εικόνα 2. Use Case: Main Program



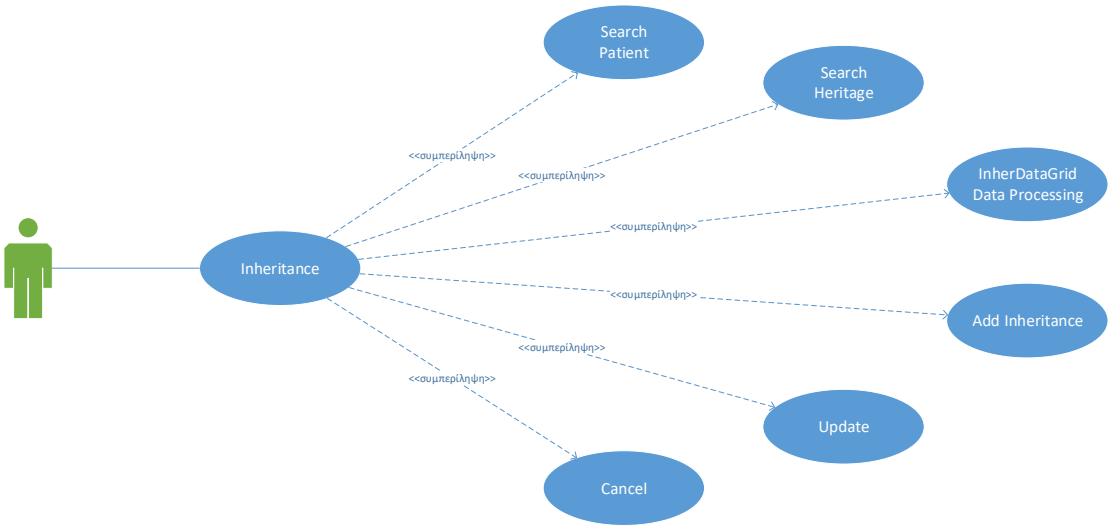
Εικόνα 3. Use Case: Patient Registration



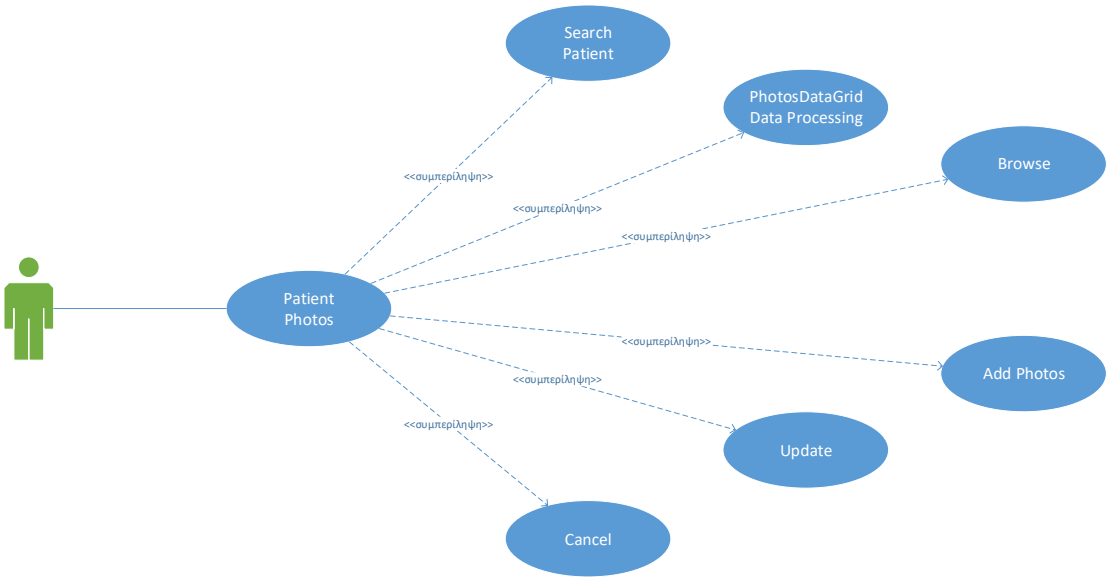
Εικόνα 4. Use Case: Patient Information



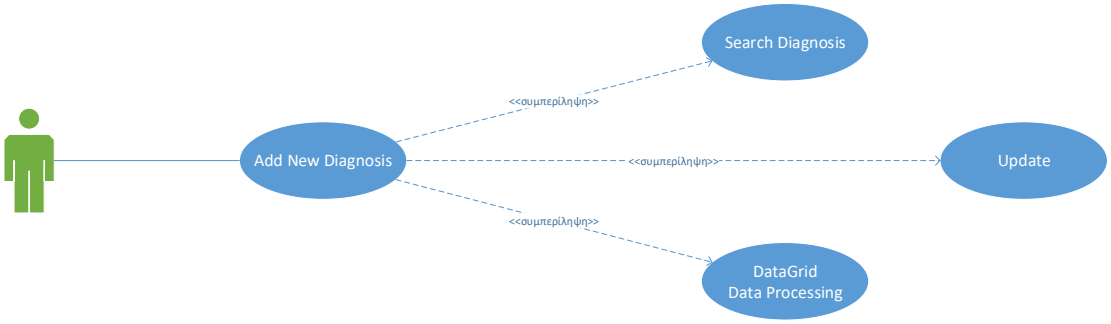
Εικόνα 5. Use Case: Visits



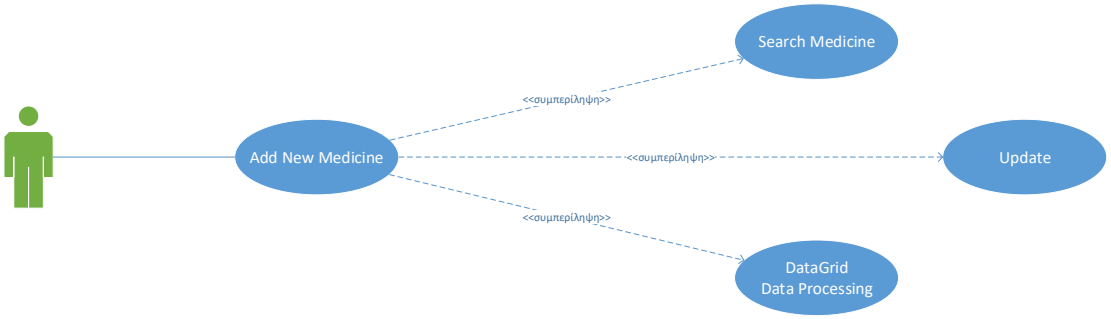
Εικόνα 6. Use Case: Inheritance



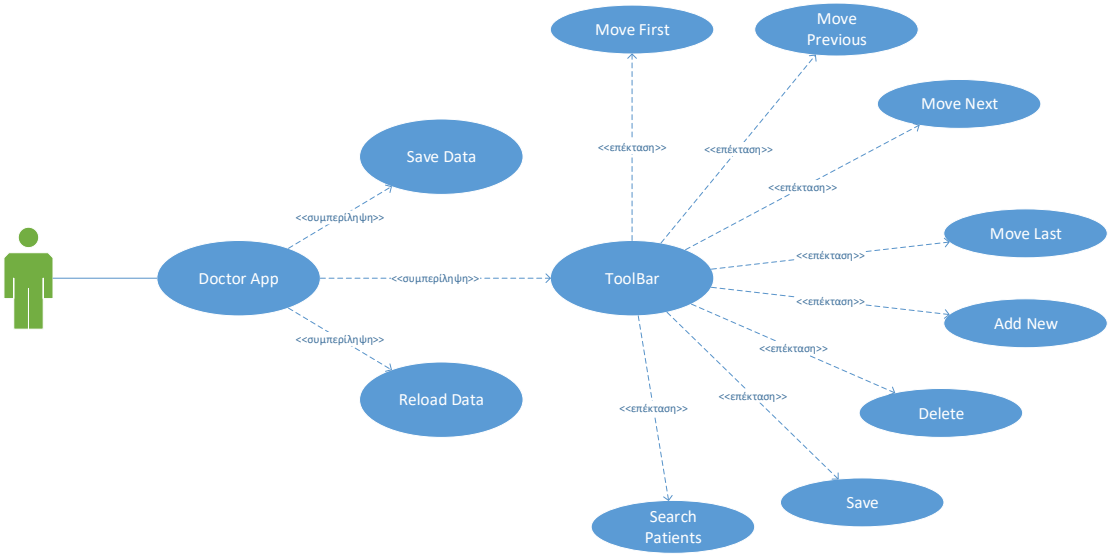
Εικόνα 7. Use Case: Patient Photos



Εικόνα 8. Use Case: Add New Diagnosis



Εικόνα 9. Use Case: Add New Medicine



Εικόνα 10. Use Case: Doctor App

Με τα παραπάνω διαγράμματα απεικονίζεται ένα δυναμικό στιγμιότυπο του συστήματος για κάθε περίπτωση ξεχωριστά. Συνεπώς, έχουμε μία εποπτική εικόνα για τις λειτουργίες του ΠΣ. Στις επόμενες ενότητες θα επιχειρηθεί μία λεπτομερής ανάλυση στην υλοποίηση του υπό εξέταση ΠΣ.

5.3 Υλοποίηση

5.3.1 Βάση Δεδομένων

Το σημείο εκκίνησης για την λεπτομερή ανάλυση δημιουργίας του πληροφοριακού συστήματος είναι η Βάση Δεδομένων. Αυτό γίνεται καθώς η σωστή κατασκευή του σχήματος της ΒΔ ευθύνεται για την αποτελεσματικότητα και λειτουργικότητα της εφαρμογής σε πολύ μεγάλο βαθμό, για όλη την διάρκεια ζωής της.

Η βάση δεδομένων είναι μια μεγάλη ποσότητα ευρετηριασμένων ψηφιακών πληροφοριών που μπορούν να αναζητηθούν, να αναφερθούν, να συγκριθούν, να αλλάξουν ή να χειριστούν με οποιονδήποτε άλλο τρόπο με βέλτιστη ταχύτητα και ελάχιστη δαπάνη επεξεργασίας.

Με άλλα λόγια η Βάση Δεδομένων (Database) είναι ένας οργανωμένος τρόπος αποθήκευσης πληροφοριών και πρόσβασης σε αυτές. Μια βάση δεδομένων είναι κάτι παραπάνω από μια απλή συλλογή αποθηκευμένων στοιχείων. Ένας άλλος ορισμός είναι ότι μια βάση δεδομένων είναι ένα ολοκληρωμένο σύστημα που αποτελείται από δεδομένα και από το κατάλληλο λογισμικό, τα οποία χρησιμοποιώντας το υλικό (hardware) βοηθούν στην ενημέρωση και πληροφόρηση των χρηστών.

Στοιχεία Βάσης Δεδομένων

Μια βάση δεδομένων αποτελείται από πολλά κύρια στοιχεία:

- Σχήμα - Μια βάση δεδομένων περιέχει ένα ή περισσότερα σχήματα, τα οποία είναι βασικά μια συλλογή από έναν ή περισσότερους πίνακες δεδομένων.
- Πίνακας - Κάθε πίνακας περιέχει πολλές στήλες, οι οποίες είναι παρόμοιες με τις στήλες σε ένα υπολογιστικό φύλλο. Ένας πίνακας μπορεί να έχει μόνο δύο στήλες και περισσότερες από εκατό ή περισσότερες στήλες, ανάλογα με τον τύπο των δεδομένων που αποθηκεύονται στον πίνακα.
- Στήλη - Κάθε στήλη περιέχει έναν από διάφορους τύπους δεδομένων ή τιμών, όπως ημερομηνίες, αριθμητικές ή ακέραιες τιμές και αλφαριθμητικές τιμές (επίσης γνωστές ως varchar).
- Γραμμή - Τα δεδομένα σε έναν πίνακα παρατίθενται σε σειρές, οι οποίες είναι σαν σειρές δεδομένων σε ένα υπολογιστικό φύλλο. Συχνά υπάρχουν εκατοντάδες ή χιλιάδες σειρές δεδομένων σε έναν πίνακα.

Ένα πρόγραμμα που διαχειρίζεται βάσεις δεδομένων αποκαλείται Σύστημα Διαχείρισης Βάσεων Δεδομένων (DBMS, Database Management System) και με την βοήθειά του μπορούμε να αποθηκεύσουμε, προσθέσουμε, τροποποιήσουμε, εμφανίσουμε ή και διαγράψουμε τα αποθηκευμένα δεδομένα. Τα δεδομένα που υπάρχουν στις βάσεις δεδομένων πρέπει να είναι:

• Ολοκληρωμένα (Integrated), δηλαδή τα δεδομένα πρέπει να είναι αποθηκευμένα σε ομοιόμορφα οργανωμένα σύνολα αρχείων όπου δεν πρέπει να υπάρχει επανάληψη ή πλεονασμός (redundancy) των ίδιων στοιχείων.

• Καταμεριζόμενα (Shared), δηλαδή να μπορούν περισσότεροι του ενός χρήστες να βλέπουν και να μοιράζονται τα ίδια δεδομένα την ίδια χρονική στιγμή.

Το DBMS περιέχει κάποια εργαλεία γενικής χρήσης για να μπορούμε να δημιουργούμε και να χειριζόμαστε τα δεδομένα. Στα νεώτερα DBMS μπορούμε να έχουμε άμεση πληροφόρηση χωρίς να απαιτείται η παρουσία ενός προγραμματιστή. Τα δεδομένα ενός DBMS μπορούν να χρησιμοποιηθούν σε κάθε μορφής ερώτημα (query) ώστε να αντλήσουμε τις πληροφορίες θέλουμε.

Μια βάση δεδομένων δημιουργείται και συντηρείται χρησιμοποιώντας μια γλώσσα προγραμματισμού βάσης δεδομένων. Η πιο κοινή γλώσσα της βάσης δεδομένων είναι η SQL, αλλά υπάρχουν πολλές "εκδοχές" της SQL, ανάλογα με τον τύπο της βάσης δεδομένων που χρησιμοποιείται. Κάθε εκδοχή της SQL έχει διαφορές στη σύνταξη SQL και έχει σχεδιαστεί για να χρησιμοποιείται με συγκεκριμένο τύπο βάσης δεδομένων. Για παράδειγμα, μια βάση δεδομένων Oracle χρησιμοποιεί PL / SQL και Oracle SQL (έκδοση Oracle της SQL). Μια βάση δεδομένων της Microsoft χρησιμοποιεί Transact-SQL (T-SQL).

Αυτή την στιγμή κυκλοφορούν αρκετά RDBMS στην αγορά. Μερικά από αυτά είναι:

α) MS SQL Server

β) MySQL

γ) Oracle

δ) SAP SQL Anywhere

ε) MariaDB

στ) Postgres SQL

Όπως ήδη αναφέρθηκε, το RDBMS που χρησιμοποιήθηκε για την κατασκευή της βάσης δεδομλενων είναι ο MSSQL Server 2016. Η επιλογή του συγκεκριμένου RDBMS έγινε διότι συνεργάζεται άψογα με το Visual Studio και είναι δωρεάν από το MSDNAA. Επίσης, αξιοποιήθηκαν οι γνώσεις που αποκτήθηκαν στο αντίστοιχο μάθημα Β Εξαμήνου «Βάσεις Δεδομένων», στο οποίο είχε γίνει αναλυτική παρουσίαση του τρόπου λειτουργίας της Postgres SQL.

Στη συνέχεια θα μελετήσουμε το σχεσιακό μοντέλο που περιγράφει τη Βάση Δεδομένων και οργανώνει τις εγγραφές με βάση τις σχέσεις. Γι' αυτό το λόγο μια Βάση δεδομένων σχεδιασμένη με βάση το σχεσιακό μοντέλο, μπορεί εύκολα να υλοποιηθεί με ένα μοντέλο Οντοτήτων - Συσχετίσεων. Στις σχεσιακές Βάσεις Δεδομένων, οι εγγραφές οργανώνονται σε πίνακες. Οι πίνακες σε μια σχεσιακή Βάση Δεδομένων, αποτελούνται από μια ή περισσότερες στήλες που αντιστοιχούν σε τιμές πεδίων (ή στα χαρακτηριστικά για τα μοντέλα Οντοτήτων - Συσχετίσεων) και από γραμμές που αντιστοιχούν σε εγγραφές για αυτά τα πεδία.

5.3.2 Μοντέλο Οντοτήτων-Σχέσεων (E-R)

Ένα μοντέλο Οντοτήτων-Σχέσεων (Ο-Σ) περιγράφει την σχέση μεταξύ των αντικειμένων ενδιαφέροντος σε ένα συγκεκριμένο γνωστικό τομέα και αποτελεί συνήθως το αποτέλεσμα μίας συστηματικής ανάλυσης, με σκοπό να ορίσει και να περιγράψει τι είναι σημαντικό και σε ποιο βαθμό. Συγκεκριμένα στον τομέα του software engineer, χρησιμοποιείται για να παρέχει ένα εννοιολογικό σχήμα κατά την σχεδίαση Βάσεων Δεδομένων, ως μοντέλο δεδομένων ενός συστήματος και των απαιτήσεων του με προσέγγιση top-down. Ένα διάγραμμα που δημιουργείται με αυτή την διαδικασία σχεδίασης καλείται Διάγραμμα Οντοτήτων – Σχέσεων (E-R Diagram).

Το μοντέλο Οντοτήτων-Σχέσεων χρησιμοποιείται στο πρώτο στάδιο σχεδίασης ενός συστήματος πληροφοριών, κατά την ανάλυση των απαιτήσεών του. Σκοπός του είναι να περιγράφει τις αναγκαίες πληροφορίες οι οποίες πρόκειται να αποθηκευτούν στη βάση δεδομένων ή τον τύπο τους. Η μοντελοποίηση δεδομένων γίνεται για την περιγραφή των χρησιμοποιούμενων όρων και των σχέσεων τους σε έναν ορισμένο τομέα ενδιαφέροντος. Στην περίπτωση σχεδιασμού ενός συστήματος πληροφοριών, που στηρίζεται σε μια βάση δεδομένων, το εννοιολογικό μοντέλο δεδομένων χαρτογραφείται σε προχωρημένο στάδιο σε ένα λογικό μοντέλο δεδομένων, όπως το σχεσιακό μοντέλο δεδομένων. Το στάδιο αυτό

ονομάζεται συνήθως στάδιο λογικού σχεδιασμού. Ύστερα, κατά τη διάρκεια του φυσικού σχεδιασμού το λογικό μοντέλο χαρτογραφείται σε κάποιο φυσικό μοντέλο. Ορισμένες φορές και οι δύο φάσεις αναφέρονται ως "φυσικός σχεδιασμός".

Τα βασικά στοιχεία ενός μοντέλου Ο-Σ είναι τα εξής:

α) **Οντότητα**. Αποτελεί ένα αντικείμενο ενδιαφέροντος στον πραγματικό κόσμο, το οποίο ξεχωρίζει από τα υπόλοιπα. Μια οντότητα λειτουργεί αφαιρετικά σε έναν πολύπλοκο τομέα. Στιγμιότυπο μιας οντότητας είναι μία συγκεκριμένη περίπτωση ενός τύπου οντότητας.

β) **Τύπος Οντότητας**. Αποτελεί μία συλλογή χαρακτηριστικών, που περιγράφουν την οντότητα.

γ) **Χαρακτηριστικό**. Κάθε οντότητα έχει διάφορα στοιχεία που την προσδιορίζουν. Ένα τέτοιο στοιχείο ονομάζεται ιδιότητα, χαρακτηριστικό ή πεδίο της οντότητας, τα οποία μπορεί να είναι μονότιμα ή πλειότιμα.

δ) **Συσχέτιση**. Είναι η σύνδεση δύο ή περισσοτέρων τύπων οντοτήτων, η οποία παρουσιάζει ενδιαφέρον για σχεδιασμό. Ένας τύπος συσχέτισης παρουσιάζεται με ρόμβο.

ε) **Πληθικότητα**. Περιγράφει τον αριθμό των στιγμιότυπων ενός τύπου οντοτήτων, που μπορούν αντιστοιχίζονται με μία οντότητα ενός άλλου τύπου σε μία συσχέτιση. Μπορούμε να έχουμε συσχετίσεις με λόγο πληθικότητας 1-1 (ένα προς ένα), 1-N (ένα προς πολλά), N-N (πολλά προς πολλά).

στ) **Ασθενής Τύπος Οντότητας**. Είναι μία οντότητα, η οποία εξαρτάται από την ύπαρξη κάποιας άλλης.

ζ) Πλειότιμα Χαρακτηριστικά.

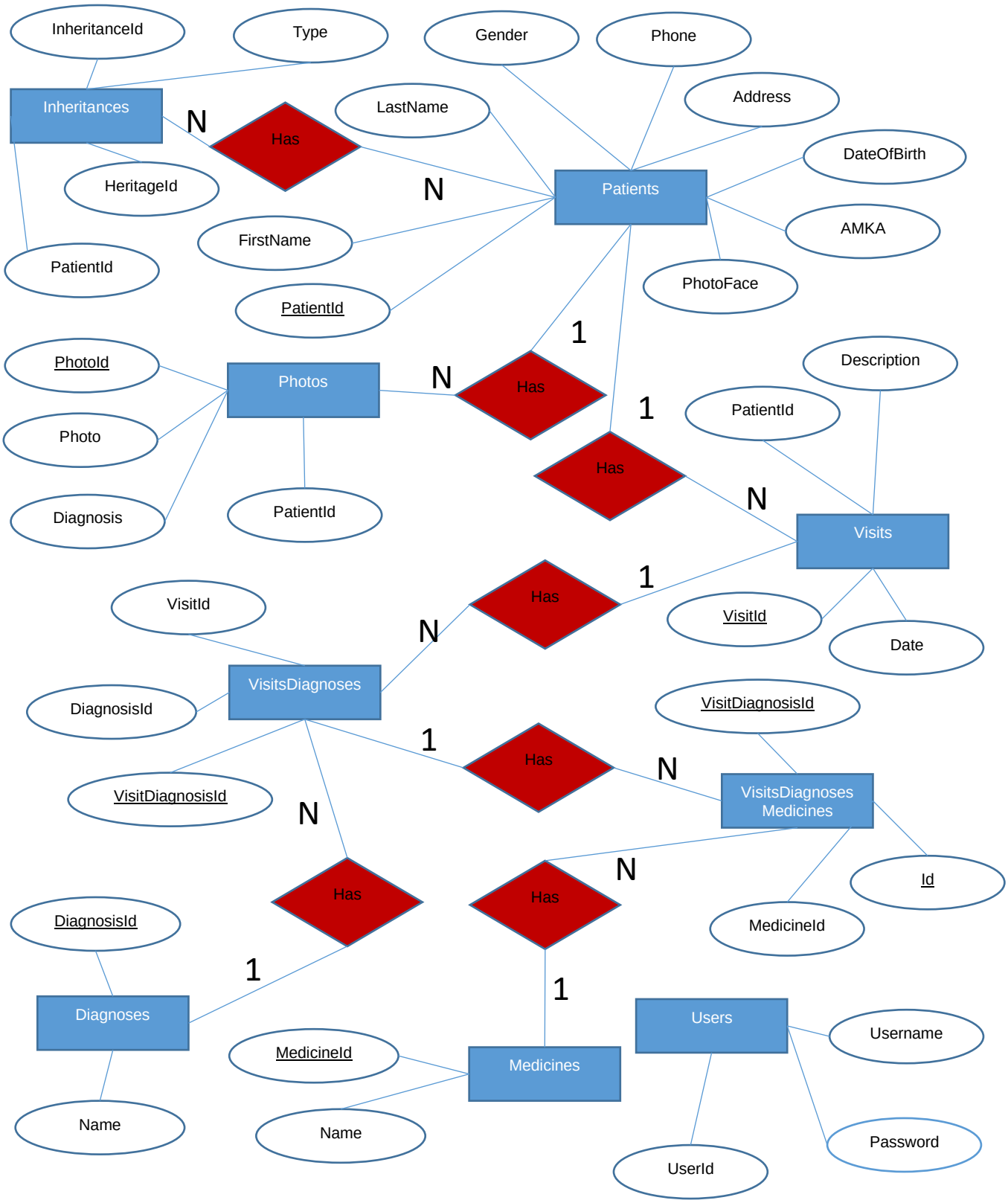
η) **Γενίκευση / Εξειδίκευση**. Με την έννοια γενίκευση (generalization) εννοούμε τον εντοπισμό ενός συνόλου οντοτήτων (κλάση) που έχουν κοινά χαρακτηριστικά με πιο γενικευμένα αντικείμενα (υπέρκλαση). Η εξειδίκευση (specialization) είναι το ακριβώς αντίθετο της γενίκευσης, δηλαδή ο εντοπισμός υποσυνόλων ενός τύπου οντοτήτων με κοινά χαρακτηριστικά, τα οποία τα διαφοροποιούν από τα υπόλοιπα μέλη του.

θ) **Κληρονομικότητα**. Σε κάθε επίπεδο της ιεραρχίας οι τύποι οντοτήτων κληρονομούν τα χαρακτηριστικά των τύπων του αμέσως ανώτερου επιπέδου.

Συνδυάζοντας όλα τα παραπάνω στοιχεία, συνθέτουμε το διάγραμμα Ο-Σ, όπως θα δούμε στην επόμενη ενότητα.

Στο παρακάτω διάγραμμα, με μπλε χρώμα φαίνονται 9 οντότητες, ενώ με ερυθρό χρώμα φαίνονται οι συσχετίσεις μεταξύ τους. Η ανάλυση του διαγράμματος οδηγεί στην δημιουργία ενός σωστού και αξιόπιστου σχήματος ΒΔ.

Με την βοήθεια του παρακάτω διαγράμματος και σε συνδυασμό με τις παραδοχές που έχουν οριστεί, διευκολύνεται κατά πολύ η δημιουργία των πινάκων και των σχέσεων της ΒΔ, όπως θα δούμε στην συνέχεια.


InheritanceId
Type
Gender
Phone
Inheritances
N
Has
LastName
Address
HeritageId
N
Patients
DateOfBirth
AMKA
FirstName
PatientId
PhotoFace
PatientId
1
PhotoId
Photos
N
Has
Description
Photo
PatientId
1
Has
Diagnosis
PatientId
N
Visits
1
Has
VisitId
VisitId
N
Date
VisitsDiagnoses
DiagnosisId
VisitDiagnosisId
1
Has
N
VisitsDiagnoses Medicines
VisitDiagnosisId
N
Has
Has
N
Id
DiagnosisId
MedicineId
1
1
Diagnoses
Users
Username
MedicineId
Medicines
Name
Name
Password
UserId

5.3.3 Σχεσιακό Μοντέλο Δεδομένων

Το Σχεσιακό Μοντέλο Δεδομένων αποτελεί το κύριο μοντέλο σχεδίασης Βάσης Δεδομένων παγκοσμίως. Το συγκεκριμένο μοντέλο χαρακτηρίζεται από την απλότητα του και έχει όλες εκείνες τις ιδιότητες και δυνατότητες, που απαιτούνται για την γρήγορη και αποτελεσματική επεξεργασία δεδομένων.

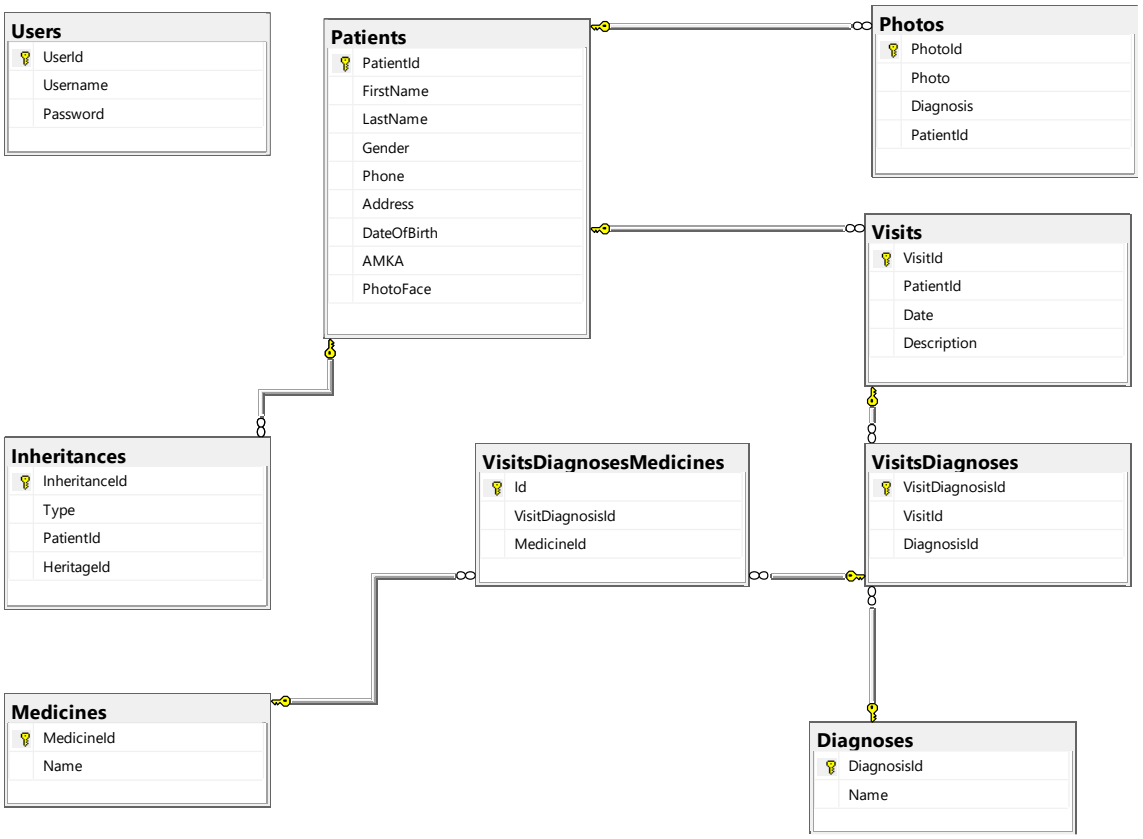
Τα δεδομένα είναι δυνατόν να αναδιοργανώνονται με πολλούς διαφορετικούς τρόπους, σε νοητούς πίνακες, χωρίς να είναι απαραίτητη η αναδιοργάνωση των φυσικών πινάκων που τα αποθηκεύουν. Οι ερωτήσεις, είτε από το χρήστη είτε από λογισμικό, προς τη βάση δεδομένων, γίνονται συνήθως μέσω της διαδεδομένης διαλογικής γλώσσας SQL (Structured Query Language).

Εκτελώντας ερωτήματα ο χρήστης (ή το λογισμικό που εκπροσωπεί το χρήστη) είναι δυνατόν, ανάλογα με τα δικαιώματά του, να δημιουργήσει, να μεταβάλλει και να διαγράψει δεδομένα στη βάση, ή να ανασύρει πληροφορίες με σύνθετα κριτήρια αναζήτησης.

Σε αυτό το μοντέλο, τα δεδομένα μιας εφαρμογής αναπαρίστανται ως ένα σύνολο από σχέσεις (relations) οι οποίες μπορεί να είναι πίνακες-αρχεία. Στις πιο πολλές περιπτώσεις υιοθετείται η χρήση πινάκων (tables) που περιέχουν ένα πλήθος γραμμών (rows) και στηλών (columns). Η κάθε μια από αυτές τις γραμμές – οι οποίες στην ορολογία του μοντέλου ονομάζονται και πλειάδες (tuples) – περιέχει ένα σύνολο απλών πεδίων (attributes), τα οποία συσχετίζονται μεταξύ τους. Επειδή οι πίνακες χρησιμοποιούνται για την αναπαράσταση των τύπων οντοτήτων καθώς και των τύπων συσχετίσεων που υφίστανται ανάμεσά τους, μπορούμε να θεωρήσουμε κάθε μια από τις γραμμές ενός πίνακα σαν ένα στιγμιότυπο οντότητας ή συσχέτισης ανάλογα με το αντικείμενο στο οποίο αναφέρεται.

Τέλος κάθε πεδίο ή στήλη ενός πίνακα, δέχεται τιμές οι οποίες ανήκουν σε ένα συγκεκριμένο και καθορισμένο, εκ των προτέρων, σύνολο τιμών (domain). Το είδος των τιμών αυτού του συνόλου καθορίζεται από τον τύπο δεδομένων του πεδίου του πίνακα, ο οποίος με τη σειρά του ορίζεται κατά το στάδιο της λογικής σχεδίασης της εφαρμογής.

Εφαρμόζοντας τα παραπάνω και με την βοήθεια του διαγράμματος Οντοτήτων – Σχέσεων, εξάγονται οι πίνακες και οι σχέσεις μεταξύ τους, καταλήγοντας στο σχήμα της Βάσης Δεδομένων της εφαρμογής, όπως φαίνεται στο ακόλουθο διάγραμμα:



Στο παραπάνω διάγραμμα παρουσιάζονται οι 9 πίνακες της ΒΔ καθώς και οι σχέσεις που υφίστανται μεταξύ τους. Εκ πρώτης όψεως φαίνεται ότι οι Οντότητες που περιλάμβανε το Μοντέλο Οντοτήτων-Σχέσεων, έχουν μετατραπεί σε πίνακες, καθώς αποτελεί βασικό κανόνα για την δημιουργία του Σχεσιακού Μοντέλου ΒΔ προερχόμενου από το Μοντέλο Οντοτήτων-Σχέσεων. Στην συνέχεια, το είδος των σχέσεων των Οντοτήτων καθώς και η πληθικότητα τους, καθόρισαν την δημιουργία επιπλέον πινάκων ή την προσθήκη επιπλέον πεδίων στους ήδη ορισμένους πίνακες

Σε ότι αφορά τις σχέσεις των πινάκων μεταξύ τους, αυτές είναι κυρίως σχέσεις τύπου ένα-προς-πολλά (1-N).

Παρακάτω παρατίθεται μία σύντομη παρουσίαση του κάθε πίνακα για το Σχεσιακό Μοντέλο της ΒΔ:

α) Πίνακας “**Users**”: Εδώ αποθηκεύονται τα usernames και τα passwords των χρηστών που θα έχουν πρόσβαση στην εφαρμογή.

β) Πίνακας “**Patients**”: Περιλαμβάνει ονοματεπώνυμο, δημογραφικά στοιχεία, στοιχεία επικοινωνίας και φωτογραφία των ασθενών.

γ) Πίνακας “**Diagnosis**”: Περιλαμβάνει τις ονομασίες των ασθενειών που προσθέτει ο χρήστης στην εφαρμογή.

δ) Πίνακας “**Medicines**”: Περιλαμβάνει τις ονομασίες των φαρμάκων που χορηγεί ο γιατρός στους ασθενείς.

ε) Πίνακας “**Photos**”: Περιλαμβάνει φωτογραφίες ασθενών μαζί με τις διαγνώσεις τους.

στ) Πίνακας “**Inheritances**”: Περιλαμβάνει τον τύπο κληρονομικότητας μεταξύ δύο ασθενών.

ζ) Πίνακας "**Visits**": Περιλαμβάνει τις επισκέψεις των ασθενών, την ημερομηνία που έγιναν και μια σύντομη περιγραφή της διάγνωσης του γιατρού.

η) Πίνακας "**VisitsDiagnoses**": Είναι ο πίνακας που ενώνει τις επισκέψεις με τις διαγνώσεις.

θ) Πίνακας "**VisitsDiagnosesMedicines**": Είναι ο πίνακας που ενώνει τον πίνακα επισκέψεων-διαγνώσεων με των πίνακα των φαρμάκων.

Στο Παράρτημα της παρούσας ΔΕ παρατίθενται οι εντολές SQL, οι οποίες απαιτούνται για την δημιουργία των παραπάνω πινάκων. Επίσης, φαίνονται τα primary keys, τα foreign keys και επιπλέον λεπτομέρειες όπως constraints για τον κάθε πίνακα.

5.3.4 Microsoft SQL Server

Αποτελεί το σύστημα διαχείρισης της σχεσιακής βάσης δεδομένων που έχει αναπτυχθεί από τη Microsoft. Σκοπός του συστήματος είναι η αποθήκευση δεδομένων και η ανάκτησή τους εφόσον απαιτηθεί από άλλες εφαρμογές. Μέσω των εργαλείων που παρέχονται από το Visual Studio μπορεί να εκτελεστεί οποιαδήποτε εργασία σχεδίασης βάσης δεδομένων στον server.

5.3.5 C#

Η C# είναι μια αντικειμενοστραφής γλώσσα προγραμματισμού γενικής χρήσης και ασφαλής τύπου. Ο στόχος της γλώσσας είναι η παραγωγικότητα του προγραμματιστή. Για το σκοπό αυτό. Η γλώσσα ισορροπεί την απλότητα, την εκφραστικότητα και την απόδοση. Ο κύριος αρχιτέκτονας της γλώσσας από την πρώτη της έκδοση είναι ο Andres Hejlsberg (δημιουργός της Turbo Pascal και αρχιτέκτονας της Delphi). Η γλώσσα C# είναι ουδέτερης πλατφόρμας αλλά γράφτηκε για να λειτουργεί καλά με το Microsoft .NET Framework.

Προσανατολισμός Αντικειμένων
Η C# είναι μια πλούσια εκτέλεση του αντικειμενοστραφούς προτύπου, το οποίο περιλαμβάνει ενθυλάκωση, κληρονομικότητα και πολυμορφισμό. Η ενθυλάκωση σημαίνει να δημιουργηθεί ένα όριο γύρο από το αντικείμενο, να χωριστεί η εξωτερική (δημόσια) συμπεριφορά του από τα εσωτερικά (ιδιωτικά) στοιχεία εφαρμογής του. Τα διακριτικά χαρακτηριστικά της C# από αντικειμενοστραφή προοπτική είναι:

Σύστημα ενοποιημένου τύπου
Το θεμελιώδες δομικό στοιχείο της C# είναι μία ενθυλακωμένη μονάδα δεδομένων και λειτουργιών που ονομάζεται τύπος. Η C# έχει ένα σύστημα ενοποιημένου τύπου, όπου όλοι οι τύποι τελικά μοιράζονται έναν κοινό τύπο βάσης. Αυτό σημαίνει ότι όλοι οι τύποι, είτε αντιπροσωπεύουν επιχειρησιακά αντικείμενα είτε είναι αρχικοί τύποι όπως αριθμοί, μοιράζονται το ίδιο βασικό σύνολο λειτουργικότητας. Για παράδειγμα, μια εμφάνιση οποιουδήποτε τύπου μπορεί να μετατραπεί σε μια συμβολοσειρά καλώντας τη μέθοδο ToString.

Τάξεις και διεπαφές
Σε ένα παραδοσιακό αντικειμενοστραφές πρότυπο, το μόνο είδος τύπου είναι μια τάξη. Στην C#, υπάρχουν διάφορα άλλα είδη τύπων, ένα από τα οποία είναι μια διεπαφή. Μια διεπαφή είναι σαν μια κλάση, εκτός από το ότι περιγράφει μόνο τα μέλη. Η υλοποίηση για αυτά τα μέλη προέρχεται από τύπους που υλοποιούν τη διεπαφή. Οι διεπαφές είναι ιδιαίτερα χρήσιμες σε σενάρια όπου απαιτείται πολλαπλή κληρονομικότητα (σε αντίθεση με γλώσσες όπως η C++ και η Eiffel, η C# δεν υποστιρίζει πολλαπλή κληρονομικότητα των τάξεων).

Ιδιότητες, μέθοδοι και συμβάντα
Στον καθαρό αντικειμενοστραφές πρότυπο, όλες οι λειτουργίες είναι μέθοδοι. Στην C#, οι μέθοδοι είναι μόνο ένα είδος μέλους λειτουργίας, η οποία περιλαμβάνει επίσης ιδιότητες και

γεγονότα. Οι ιδιότητες είναι μέλη λειτουργίας που ενσωματώνουν ένα κομμάτι της κατάστασης ενός αντικειμένου, όπως το χρώμα ενός κουμπιού ή το κείμενο μιας ετικέτας. Τα συμβάντα είναι μέλη λειτουργιών που απλοποιούν τη λειτουργία της αλλαγής κατάστασης των αντικειμένων.

Ενώ η C# είναι κυρίως αντικειμενοστραφής γλώσσα, δανείζεται επίσης από το λειτουργικό πρότυπο προγραμματισμού. Ειδικά:

Οι λειτουργίες μπορούν να θεωρηθούν ως τιμές

Μέσω της χρήσης των αντιπροσώπων (delegates), η C# επιτρέπει τις λειτουργίες να μεταβιβάζονται ως τιμές προς και από άλλες λειτουργίες.

Η C# υποστηρίζει μοτίβα για καθαρότητα

Ο πυρήνας του λειτουργικού προγραμματισμού αποφεύγει τη χρήση μεταβλητών των οποίων οι αξίες αλλάζουν, προς όφελος των δηλωτικών προτύπων. Η C# έχει βασικές λειτουργίες για να βοηθήσει με αυτά τα μοτίβα, συμπεριλαμβανομένης της δυνατότητας να γράφουν ανώνυμες λειτουργίες στον αέρα που τις μεταβιβάζουν στις μεταβλητές (lambda εκφράσεις) και τη δυνατότητα εκτέλεσης λίστας ή αντιδραστικού προγραμματισμού μέσω εκφράσεων ερωτημάτων. Η C# 6.0 περιλαμβάνει επίσης αυτόματες ιδιότητες ανάγνωσης για να σας βοηθήσει να γράφεται αμετάβλητους τύπους(μόνο για ανάγνωση).

Τύπου Ασφάλειας

Η C# είναι κατά κύριο λόγο μια γλώσσα ασφαλούς τύπου, που σημαίνει ότι υποδείξεις τύπων μπορούν να αλληλοεπιδρούν μόνο μέσω πρωτοκόλλων που ορίζουν, εξασφαλίζοντας έτσι την εσωτερική συνοχή του κάθε τύπου. Για παράδειγμα, η C# σας εμποδίζει να αλληλοεπιδράσετε με έναν τύπο συμβολοσειράς σαν να ήταν ένας ακέραιος τύπος.

Ειδικότερα, η C# υποστηρίζει τη στατική πληκτρολόγηση, πράγμα που σημαίνει ότι η γλώσσα επιβάλλει τον ασφαλή τύπο κατά τον χρόνο σύνταξης. Αυτό είναι επιπλέον προς τον ασφαλή τύπο που επιβάλλεται κατά το χρόνο εκτέλεσης.

Η στατική πληκτρολόγηση εξαλείφει μια μεγάλη κατηγορία σφαλμάτων πριν από την εκτέλεση ενός προγράμματος. Μετατοπίζει το βάρος από τις δοκιμές μονάδας χρόνου εκτέλεσης στον μεταγλωττιστή για να βεβαιωθεί ότι όλοι οι τύποι σε ένα πρόγραμμα ταιριάζουν σωστά μαζί. Αυτό κάνει τα μεγάλα προγράμματα πολύ ευκολότερα στη διαχείριση, πιο προβλέψιμα και πιο ισχυρά. Επιπλέον, η στατική πληκτρολόγηση επιτρέπει εργαλεία όπως το IntelliSense στο Visual Studio για να βοηθήσει να γραφτεί ένα πρόγραμμα, δεδομένου ότι γνωρίζει για μια δεδομένη μεταβλητή τι είδους είναι και, συνεπώς, ποιες μέθοδοι μπορούν να καλεστούν σε αυτή τη μεταβλητή.

Η C# ονομάζεται επίσης μια έντονη γλώσσα πληκτρολόγησης επειδή οι κανόνες τύπου (είτε επιβάλλονται στατικά ή κατά το χρόνο εκτέλεσης) είναι πολύ αυστηροί. Για παράδειγμα δεν μπορείτε να καλέσετε μια λειτουργία η οποία έχει σχεδιαστεί για να δέχεται έναν ακέραιο αριθμό με έναν κινητής υποδιαστολής, εκτός αν πρώτα μετατρέψετε ρητά τον αριθμό κινητής υποδιαστολής σε ακέραιο αριθμό. Αυτό βοηθά στην πρόληψη λαθών.

Η ισχυρή πληκτρολόγηση παίζει ρόλο στην ενεργοποίηση του κώδικα της C# για να τρέχει σε ένα sandbox – ένα περιβάλλον όπου κάθε πτυχή της ασφάλειας ελέγχεται από τον κεντρικό υπολογιστή. Σε ένα sandbox είναι σημαντικό να μην μπορεί να καταστραφεί αυθαίρετα η κατάσταση ενός αντικειμένου παρακάμπτοντας τους κανόνες του τύπου.

Διαχείριση Μνήμης

Η C# βασίζεται στο χρόνο εκτέλεσης για να εκτελέσει αυτόματα τη διαχείριση μνήμης. Το Common Language Runtime ή αλλιώς CLR διαθέτει συλλέκτη απορριμμάτων που εκτελείται ως μέρος του προγράμματος με σκοπό την ανάκτηση μνήμης για αντικείμενα που δεν

χρησιμοποιούνται πλέον. Αυτό απαλλάσσει τους προγραμματιστές από την ρητή ανακατανομή της μνήμης για ένα αντικείμενο εξαλείφοντας το πρόβλημα των εσφαλμένων δεικτών που συναντάμε σε γλώσσες όπως η C++.

Η C# δεν εξαλείφει τους δείκτες: απλώς τους καθιστά περιττούς για τις περισσότερες εργασίες προγραμματισμού. Για hotspots κρίσιμης απόδοσης και διαλειτουργικότητας, μπορούν να χρησιμοποιηθούν δείκτες, αλλά επιτρέπονται μόνο σε μπλοκ που χαρακτιρίζονται ρητά ως μη ασφαλείς.

Υποστήριξη Πλατφόρμας
Ιστορικά, η C# χρησιμοποιήθηκε σχεδόν εξ ολοκλήρου για την εγγραφή κώδικα για εκτέλεση σε πλατφόρμες Windows. Πρόσφατα, ωστόσο, η Microsoft και άλλες εταιρίες έχουν επενδύσει σε άλλες πλατφόρμες, συμπεριλαμβανομένων των Mac OS X και iOS, και του Android. Το Xamarin επιτρέπει την ανάπτυξη πλατφόρμας C# για εφαρμογές κινητών τηλεφώνων και οι βιβλιοθήκες φορητών κλάσεων είναι όλο και περισσότερο διαδεδομένες. Το ASP.NET 5 της Microsoft είναι ένα νέο framework φιλοξενίας ιστοσελίδων που μπορεί να εκτελεστεί είτε στο .NET Framework είτε σε .NET Core, ένα νέο μικρό, γρήγορο, ανοιχτού κώδικα, cross-platform runtime.

Σχέση C# με το CLR
Η C# εξαρτάται από ένα χρόνο εκτέλεσης που είναι εξοπλισμένος με πλήθος λειτουργιών, όπως η αυτόματη διαχείριση μνήμης και ο χειρισμός εξαιρέσεων. Ο σχεδιασμός της C# χαρτογραφεί λεπτομερώς το σχεδιασμό της Common Language Runtime της Microsoft (CLR), η οποία παρέχει αυτές τις λειτουργιές του χρόνου εκτέλεσης (αν και η C# είναι τεχνικά ανεξάρτητη από την CLR). Επιπλέον, το σύστημα τύπου της C# χαρτογραφεί προσεκτικά το σύστημα τύπου της CLR (π.χ., και οι δύο μοιράζονται τους ίδιους ορισμούς για προκαθορισμένους τύπους).

Το CLR και το .NET Framework
Το.NET Framework αποτελείται από το CLR συν ένα τεράστιο σύνολο βιβλιοθηκών. Οι βιβλιοθήκες αποτελούνται από βασικές βιβλιοθήκες και εφαρμοσμένες βιβλιοθήκες, οι οποίες εξαρτώνται από τις κεντρικές βιβλιοθήκες. Το CLR είναι ο χρόνος εκτέλεσης για την εκτέλεση του διαχειριζόμενου κώδικα. Η C# είναι μία από τις πολλές διαχειριζόμενες γλώσσες που καταρτίζονται σε διαχειριζόμενο κώδικα. Ο διαχειριζόμενος κώδικας είναι συσκευασμένος σε μία συναρμολόγηση, είτε με εκτελέσιμο αρχείο (.exe) είτε με βιβλιοθήκη (a.dll), μαζί με πληροφορίες τύπου ή μεταδεδομένα.

Ο διαχειριζόμενος κώδικας εκπροσωπείται στην ενδιάμεση γλώσσα. Όταν το CLR φορτώνει ένα αρχείο συναρμολόγησης (assembly), μετατρέπει την ενδιάμεση γλώσσα στον εγγενή κώδικα του μηχανήματος, όπως το x86. Αυτή η μετατροπή γίνεται από τον μεταγλωττιστή JIT (just-in-time) της CLR. Μία συναρμολόγηση διατηρεί σχεδόν όλες τις αρχικές δομές της γλώσσας προέλευσης, γεγονός που καθιστά εύκολο τον έλεγχο και ακόμη και τη δυναμική δημιουργία κώδικα.

Όταν γράφετε εφαρμογή των Windows Store, υπάρχει απευθείας η δυνατότητα δημιουργίας εγγενή κώδικα (“.NET Native”). Αυτό βελτιώνει την απόδοση εκκίνησης και τη χρήση της μνήμης (η οποία είναι ιδιαίτερα επωφελής για τις κινητές συσκευές) καθώς και την απόδοση εκτέλεσης μέσω στατικής σύνδεσης και άλλων βελτιστοποιήσεων.

Το CLR λειτουργεί ως κεντρικός υπολογιστής για πολλές υπηρεσίες χρόνου εκτέλεσης. Παραδείγματα αυτών των υπηρεσιών είναι η διαχείριση μνήμης, η φόρτωση βιβλιοθηκών και οι υπηρεσίες ασφαλείας. Το CLR είναι ουδέτερη γλώσσα, επιτρέποντας στους προγραμματιστές να δημιουργούν εφαρμογές σε πολλές γλώσσες (π.χ. C#, F#, Visual Basic .NET και Managed C++).

Το .NET Framework περιέχει βιβλιοθήκες για τη σύνταξη σχεδόν οποιασδήποτε εφαρμογής με Windows ή Web εφαρμογών.

C# and Windows Runtime
Η C# συνεργάζεται επίσης με τις βιβλιοθήκες του Windows Runtime (WinRT). Το WinRT είναι μια διεπαφή εκτέλεσης και ένα περιβάλλον εκτέλεσης για την πρόσβαση στις βιβλιοθήκες με ουδέτερη γλώσσα και αντικειμενοστραφή τρόπο διαμόρφωσης. Εφοδιάζεται με Windows 8 και νεότερα και είναι (εν μέρει) μια βελτιωμένη έκδοση του Μοντέλου Αντικειμένου Συστήματος (Component Object Model ή COM) της Microsoft.

Τα Windows 8 και οι νεότερες εκδόσεις περιέχουν ένα σύνολο μη διαχειριζόμενων βιβλιοθηκών WinRT που χρησιμεύουν ως πλαισίου για εφαρμογές με δυνατότητα άμεσης σύνδεσης που παρέχονται μέσω της εφαρμογής Microsoft Store. (Ο όρος WinRT αναφέρεται επίσης σε αυτές τις βιβλιοθήκες). Όντας WinRT, οι βιβλιοθήκες μπορούν εύκολα να χρησιμοποιηθούν όχι μόνο από την C# και την VB, αλλά και το την C++ και την JavaScript.

Οι βιβλιοθήκες WinRT υποστηρίζουν τη νέα ‘’σύγχρονη’’ διεπαφή χρήστη (για την εγγραφή συναρπαστικών εφαρμογών αφής), τις λειτουργίες που σχετίζονται με συγκεκριμένες συσκευές κινητής τηλεφωνίας (αισθητήρες, μηνύματα κειμένου κ.λπ.) και μια σειρά βασικών λειτουργιών που επικαλύπτονται με τμήματα του .NET Framework. Εξαιτίας αυτής της επικάλυψης, το Visual Studio περιλαμβάνει ένα προφίλ αναφοράς (ένα σύνολο συναρμολογούμενων αναφορών .NET) για εφαρμογές των Windows Store που αποκρύπτει τα τμήματα του .NET Framework που επικαλύπτονται με το WinRT. Αυτό το προφίλ επίσης κρύβει μεγάλα τμήματα του .NET Framework που θεωρούνται περιττά για εφαρμογές tablet (όπως πρόσβαση σε βάση δεδομένων). Το κατάστημα εφαρμογών της Microsoft, το οποίο ελέγχει τη διανομή του λογισμικού σε καταναλωτικές συσκευές, απορρίπτει οποιοδήποτε πρόγραμμα που προσπαθεί να αποκτήσει πρόσβαση σε κρυφό τύπο.

Η απόκρυψη του μεγαλύτερου μέρους του .NET Framework διευκολύνει την καμπύλη μάθησης για προγραμματιστές νέους στην πλατφόρμα της Microsoft, αν και υπάρχουν δύο πιο σημαντικοί στόχοι:

- Στις sandboxes εφαρμογές (περιορίζεται η λειτουργικότητα για να μειώσει την επίδραση του κακόβουλου λογισμικού). Για παράδειγμα, η αυθαίρετη πρόσβαση στο αρχείο είναι απαγορευμένη και υπάρχει η δυνατότητα να ξεκινήσετε ή να επικοινωνήσετε με άλλα προγράμματα στον υπολογιστή η οποία είναι εξαιρετικά περιορισμένη.

- Επιτρέπει τη χαμηλή ισχύ των WinRT – Μόνο tablets που φορτώνουν ένα μειωμένο .NET Framework, μειώνοντας το αποτύπωμα του λειτουργικού συστήματος (OS).

Αυτό που διακρίνει το WinRT από το συνηθισμένο COM είναι ότι το WinRT προβάλλει τις βιβλιοθήκες του σε πολλές γλώσσες, όπως C#, VB, C++ και JavaScript, έτσι ώστε κάθε γλώσσα να βλέπει τους τύπους WinRT σχεδόν σαν να γράφτηκαν ειδικά για αυτό. Για παράδειγμα, το WinRT θα προσαρμόσει τους κανόνες κεφαλαιοποίησης ώστε να ταιριάζει με τα πρότυπα της γλώσσας στόχου και θα αποδώσει ακόμη και μερικές λειτουργίες και διεπαφές. Οι συναρμολογήσεις WinRT στέλνονται επίσης με πλούσια μεταδεδομένα σε αρχεία .winmd, τα οποία έχουν την ίδια μορφή με τα .NET αρχεία συναρμολόγησης, επιτρέποντας τη διαφανή κατανάλωση χωρίς ειδική τελετουργία. Στην πραγματικότητα, ίσως να μην είναι γνωστό στους προγραμματιστές ότι χρησιμοποιούν τύπους WinRT αντί για τύπους .NET εκτός από διαφορές ονομάτων. Μια άλλη ιδέα είναι ότι οι τύποι WinRT υπόκεινται σε περιορισμούς τύπου COM. Για παράδειγμα, προσφέρουν περιορισμένη υποστήριξη για την κληρονομικότητα και τα γενόσημα.

5.3.6 .NET Framework

Το .NET Framework (προφέρεται "dot net") είναι ένα πλαίσιο λογισμικού που βασίζεται κυρίως στο Microsoft Windows. Περιλαμβάνει μια μεγάλη βιβλιοθήκη και υποστηρίζει πολλές γλώσσες προγραμματισμού τα οποία επιτρέπουν τη δια λειτουργικότητα των γλωσσών (κάθε γλώσσα μπορεί να χρησιμοποιήσει κώδικα γραμμένο σε άλλο Γλώσσες). Η βιβλιοθήκη .NET είναι διαθέσιμη σε όλες τις γλώσσες προγραμματισμού που χρησιμοποιούν το .NET υποστηρίζει. Τα προγράμματα που έχουν γραφτεί για το .NET Framework εκτελούνται σε περιβάλλον λογισμικού, γνωστό ως Κοινό Γλώσσα χρόνου εκτέλεσης (CLR), μια εικονική μηχανή εφαρμογής που παρέχει σημαντικές υπηρεσίες όπως ασφάλεια, διαχείριση μνήμης και χειρισμός εξαιρέσεων. Η βιβλιοθήκη τάξεων και το CLR μαζί αποτελούν το .NET Framework.

Το .NET Framework είναι ένα πλαίσιο εργασίας το οποίο περιλαμβάνει ένα σύνολο τεχνολογιών που περιγράφονται παρακάτω:

Γλώσσες προγραμματισμού .NET: Οι γλώσσες που χρησιμοποιούνται στο πλαίσιο .NET είναι οι C#, Visual Basic .NET (VB .NET) οι οποίες περιλαμβάνουν τις Jscript .NET (έκδοση server-side της JavaScript), J# και C++.

Common Language Runtime (CLR): Η μηχανή που εκτελεί όλα τα προγράμματα που χρησιμοποιούν το .NET, το οποίο παρέχει αυτόματα υπηρεσίες στις εφαρμογές αυτές όπως έλεγχο ασφάλειας, διαχείριση μνήμης και βελτιστοποίηση.

Βιβλιοθήκη κλάσεων .NET: Στη βιβλιοθήκη αυτή συλλέγονται χιλιάδες έτοιμα κομμάτια κώδικα που παρέχουν λειτουργικότητα προς χρήση από τις εφαρμογές που αναπτύσσονται. Τα χαρακτηριστικά αυτά οργανώνονται σε ομάδες ανάλογα με την τεχνολογία τους (π.χ. ADO.NET, τεχνολογία για την δημιουργία εφαρμογών που χρησιμοποιούν βάσεις δεδομένων).

Οι γλώσσες προγραμματισμού που χρησιμοποιούνται στο .NET μεταγλωττίζονται σε μια άλλη γλώσσα χαμηλότερου επιπέδου και στη συνέχεια εκτελείται ο κώδικας από το CLR. Η γλώσσα αυτή ονομάζεται Ενδιάμεση Γλώσσα (Microsoft Intermediate Language – MSIL ή IL) και είναι η μόνη που μπορεί να χρησιμοποιήσει το CLR. Στο παρακάτω σχεδιάγραμμα φαίνεται ο τρόπος με τον οποίο οι γλώσσες .NET μεταγλωττίζονται σε IL. Οποιοδήποτε αρχείο *.exe ή *.dll που χτίζεται με γλώσσα .NET περιέχει κώδικα IL. Αυτά είναι και τα αρχεία που αναπτύσσονται σε άλλους υπολογιστές ώστε να μπορεί να εκτελείται η εφαρμογή από αυτούς, ανεξάρτητα από τον πηγαίο κώδικά της.

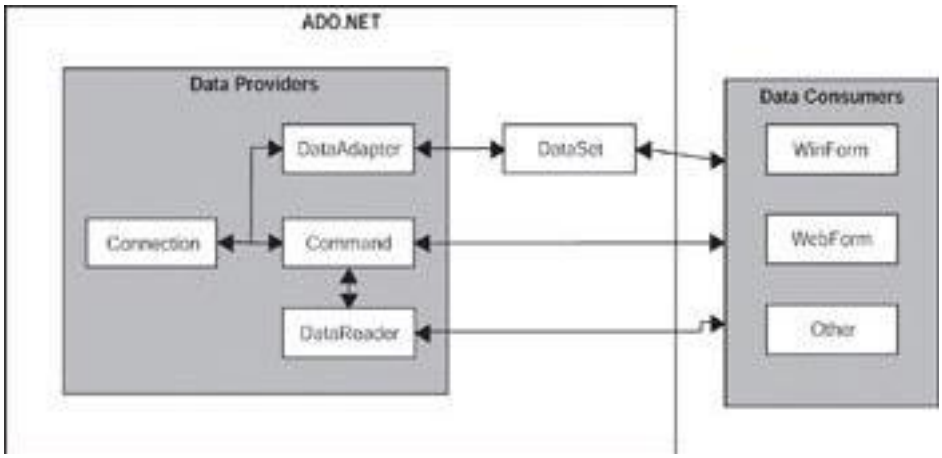
5.3.7 ADO.NET Framework

Το ADO.NET είναι ένα νέο μοντέλο προγραμματισμού που βασίζεται στο .NET Framework και μοιράζεται έναν κοινό τύπο σύστημα, τα πρότυπα σχεδίασης και τις συμβάσεις ονομασίας. Οι δηλωμένοι στόχοι του ADO.NET είναι να:

- Παρέχει μια αρχιτεκτονική δεδομένων αποσυνδεδεμένης (εκτός σύνδεσης) εκτός από την υποστήριξη της συνδεδεμένης λειτουργίας
- Ενσωματώνει στενά την XML
- Αλληλεπιδρά με μια ποικιλία βάσεων δεδομένων μέσω μιας κοινής αναπαράστασης δεδομένων
- Βελτιστοποιήστε την πρόσβαση στην προέλευση δεδομένων

Το ADO.NET έχει σχεδιαστεί για να παρέχει σταθερή πρόσβαση σε πηγές δεδομένων. Αυτό επιτυγχάνεται μέσω των ADO.NET Data Providers που παρέχουν μεθόδους για τη σύνδεση με βάσεις δεδομένων καθώς και για την ανάκτηση, τον χειρισμό και την ενημέρωση δεδομένων τόσο σε συνδεδεμένα όσο και σε αποσυνδεδεμένα περιβάλλοντα.

Οι κλάσεις ADO.NET χωρίζονται σε δύο συνιστώσες: Στους Data Providers(μερικές φορές που ονομάζονται Managed Providers), οι οποίοι χειρίζονται την επικοινωνία με ένα φυσικό κατάστημα δεδομένων και το DataSet, το οποίο αντιπροσωπεύει τα πραγματικά δεδομένα. Οποιοδήποτε στοιχείο μπορεί να επικοινωνήσει με τους Data Consumers όπως WebForms και WinForms.



Ένας ADO.NET Data Provider συνδέεται σε μια βάση δεδομένων όπως SQL Server, Oracle ή δεδομένα OLE DB και παρέχει έναν τρόπο εκτέλεσης εντολών κατά της βάσης δεδομένων με συνεπή τρόπο που είναι ανεξάρτητη από την βάση δεδομένων και τη λειτουργικότητά της. Ωστόσο, εκτός από ένα βασικό σύνολο παρόμοιων δυνατοτήτων, δεν υπάρχει εγγύηση ότι θα υπάρχει η ίδια λειτουργικότητα στον κάθε Data Provider. Αυτό οφείλεται σε διαφορές μεταξύ των βάσεων δεδομένων(για παράδειγμα, ο SQL Server παρέχει περισσότερες δυνατότητες από την Access) και τις υλοποιήσεις παρόχων (για παράδειγμα, και οι δύο Microsoft και Oracle προσφέρουν ADO.NET Data Providers για τον διακομιστή(server) δεδομένων της Oracle με μικρές διαφορές στην υλοποίηση).

Ένας πλήρης .NET Data Provider περιλαμβάνει τις ακόλουθες κλάσεις:

Connection: Το αντικείμενο της Connection κλάσης αντιπροσωπεύει τη φυσική σύνδεση με μια πηγή δεδομένων. Οι ιδιότητες του προσδιορίζουν τον data provider, την πηγή δεδομένων και τη βάση δεδομένων με την οποία θα συνδεθεί και τη συμβολοσειρά που θα χρησιμοποιηθεί κατά τη διάρκεια της σύνδεσης. Οι μέθοδοι του είναι αρκετά απλές: Μπορεί να ανοίξει και να κλείσει τη σύνδεση, να αλλάξει τη βάση δεδομένων και να διαχειριστεί τις συναλλαγές.

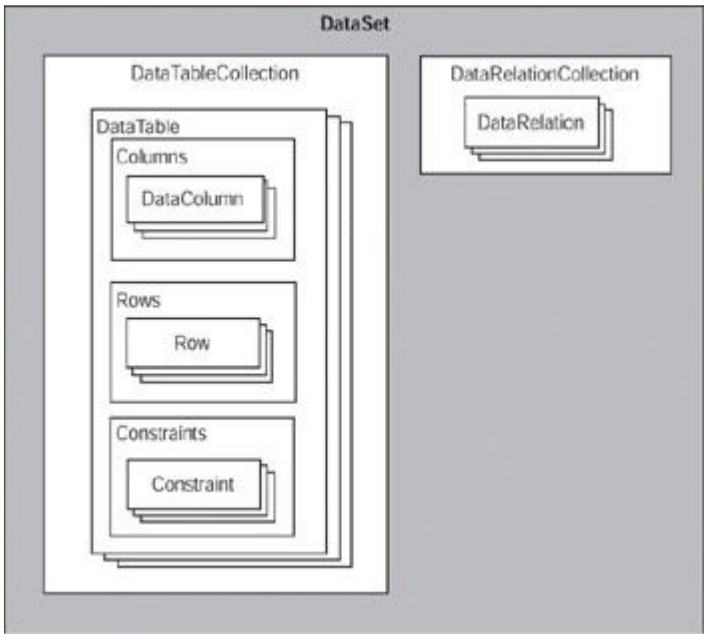
Command: Το αντικείμενο της κλάσης Command αντιπροσωπεύει μια εντολή SQL ή μια αποθηκευμένη διαδικασία που θα εκτελεστεί στη πηγή δεδομένων. Τα αντικείμενα της Command κλάσης μπορούν να δημιουργηθούν και να εκτελεστούν ανεξάρτητα έναντι σε αντικείμενο της Connection κλάσης, και χρησιμοποιούνται από αντικείμενα του DataAdapter για να χειριστούν επικοινωνίες από ένα DataSet πίσω σε μια πηγή δεδομένων. Τα αντικείμενα της Command κλάσης μπορούν να υποστηρίξουν SQL δηλώσεις και αποθηκευμένες διαδικασίες που επιστρέφουν μεμονωμένες τιμές, ένα ή περισσότερα σετ σειρές ή καθόλου τιμές.

DataReader: Ένας DataReader είναι ένα γρήγορο, χαμηλού κόστους αντικείμενο για την απόκτηση μόνο προς τα εμπρός, μόνο για ανάγνωση μιας ροής δεδομένων από μια πηγή δεδομένων. Δεν μπορούν να δημιουργηθούν απευθείας στον κώδικα. Δημιουργούνται μόνο με την κλήση της μεθόδου ExecuteReader της Command κλάσης.

DataAdapter: Ο DataAdapter είναι λειτουργικά το πιο πολύπλοκο αντικείμενο σε έναν Data Provider. Παρέχει τη γέφυρα ανάμεσα σε ένα Connection και ένα DataSet. Ο DataAdapter περιέχει τέσσερα Command αντικείμενα: SelectCommand, UpdateCommand, InsertCommand και DeleteCommand. Ο DataAdapter χρησιμοποιεί το SelectCommand για να γεμίσει ένα DataSet και χρησιμοποιεί τις τρεις υπόλοιπες εντολές για τη μετάδοση αλλαγών πίσω στην πηγή δεδομένων, όπως απαιτείται.

Ο DataAdapter και το DataSet δεν είναι ταυτόσημα στο ADO.NET

Το DataSet είναι μια αναπαράσταση δεδομένων μόνιμης μνήμης. Η δομή του εμφανίζεται στην παρακάτω εικόνα. Το DataSet μπορεί να θεωρηθεί μια κάπως απλουστευμένη σχεσιακή βάση δεδομένων, που αποτελείται από πίνακες και τις σχέσεις τους. Είναι σημαντικό να καταλάβουμε, ωστόσο, ότι το DataSet αποσυνδέεται πάντοτε από την πηγή δεδομένων - δεν γνωρίζει από πού προέρχονται τα δεδομένα και στην πραγματικότητα μπορεί να περιέχει δεδομένα από πολλαπλές πηγές.



Το DataSet αποτελείται από δύο κύρια αντικείμενα: το DataTableCollection και το DataRelationCollection. Το DataTableCollection περιέχει μηδέν ή περισσότερα DataTable αντικείμενα, τα οποία με τη σειρά τους αποτελούνται από τρεις συλλογές: στήλες, σειρές και περιορισμούς. Η DataRelationCollection περιέχει μηδέν ή περισσότερες DataRelations.

Η συλλογή Columns της DataTable ορίζει τις στήλες που συνθέτουν το DataTable. Εκτός από τις ιδιότητες ColumnName και DataType, οι ιδιότητες του DataColumn επιτρέπουν να ορίσουν τέτοια πράγματα, ανεξάρτητα από το αν επιτρέπει ή όχι null τιμές (AllowDBNull), το μέγιστο του

μήκος (MaxLength), ακόμα και μια έκφραση που χρησιμοποιείται για τον υπολογισμό της αξίας του (Expression).

Η συλλογή γραμμών του DataTable, η οποία μπορεί να είναι κενή, περιέχει τα πραγματικά δεδομένα που καθορίζεται από τη συλλογή Columns. Για κάθε σειρά, το DataTable διατηρεί το πρωτότυπο, τρέχουσες και προτεινόμενες τιμές. Όπως θα δούμε, αυτή η δυνατότητα απλοποιεί σε μεγάλο βαθμό ορισμένα είδη προγραμματισμού.

5.3.8 Visual Studio 2015

Όλες οι παραπάνω τεχνολογίες έχουν εφαρμοστεί με την βοήθεια του Visual Studio. Το Visual Studio είναι ένα ολοκληρωμένο περιβάλλον ανάπτυξης (Integrated Development Environment) όπου διατίθεται από την Microsoft και μπορεί κανείς να αναπτύξει εφαρμογές συμβατές με λειτουργικά συστήματα της Microsoft και με .NET πλατφόρμες. Με το πρόγραμμα αυτό, οι προγραμματιστές έχουν την δυνατότητα να δημιουργήσουν ιστοσελίδες (web sites), εφαρμογές διαδικτύου (web applications), διαδικτυακές υπηρεσίες (web services) καθώς και εφαρμογές κονσόλας (console applications). Επιπλέον υποστηρίζει πολλές γλώσσες προγραμματισμού με τις οποίες έχουμε την επιλογή να γράψουμε τον κώδικα μας. Αυτές είναι οι C, C++, C#, Visual Basic και J#.

Ίσως ένα από τα πιο σημαντικά χαρακτηριστικά του Visual Studio είναι η δυνατότητα που δίνεται στους προγραμματιστές για έλεγχο σφαλμάτων που πιθανόν να περιέχονται στον κώδικα (debugging). Ο μηχανισμός αυτός λειτουργεί και σε επίπεδο πηγαίου κώδικα και σε επίπεδο γλώσσας μηχανής. Επίσης μπορεί και χειρίζεται με την ίδια ευκολία κώδικες είτε είναι γραμμένοι με διαχειρίσιμο κώδικα (managed code) είτε είναι γραμμένοι σε φυσική γλώσσα (native code) καθώς και εφαρμογές που είναι γραμμένες σε γλώσσα προγραμματισμού που υποστηρίζεται από το Visual Studio. Επιπλέον μπορεί να επισυναφθεί σε μια διαδικασία οποία τρέχει και να μπορεί κάποιος να την παρακολουθήσει και να διορθώσει επιτόπου τυχόν σφάλματα. Σε περίπτωση που ο κώδικας είναι διαθέσιμος μπορούμε να τον βλέπουμε ταυτόχρονα με την διαδικασία που τρέχει, αλλιώς, σε αντίθετη περίπτωση, βλέπουμε την αντίστοιχη γλώσσα μηχανής. Επίσης ο debbuger μπορεί και δημιουργεί έναν μηχανισμό συλλογής απορριμμάτων (garbage collection), τον οποίο ενεργοποιεί όταν χρειαστεί να τρέξει η διαδικασία ελέγχου σφαλμάτων. Επιπρόσθετα, ο μηχανισμός εντοπισμού σφαλμάτων μας επιτρέπει να σταματάμε το τρέξιμο του κώδικα σε ένα ορισμένο από εμάς σημείο και να παρακολουθούμε τις τιμές των τρέχων μεταβλητών που χρησιμοποιούνται στον κώδικα. Τέλος είναι δυνατή η διόρθωση του κώδικά καθώς ελέγχεται για σφάλματα.

Ένα άλλο πολύ σημαντικό χαρακτηριστικό του Visual Studio είναι από μόνο του το περιβάλλον όπου μπορεί να γράψει κανείς κώδικα (code editor). Καθώς γράφουμε κώδικά, παράλληλα τρέχει και μια λειτουργεία, γνωστή ως IntelliSense, η οποία μπορεί να συμπληρώσει αυτόματα μεταβλητές, συναρτήσεις, μεθόδους, επαναληπτικούς βρόγχους και LINQ ερωτήματα. Η δυνατότητα αυτή υποστηρίζει τις γλώσσες προγραμματισμού που υποστηρίζει και το Visual Studio, XML, CSS και Javascript. Στον editor μπορούμε επίσης να προσθέσουμε κάποιον σελιδοδείκτη για να συνεχίσουμε από κει που μείναμε την τελευταία φορά αλλά και να συμπτύξουμε κάποιος μπλοκ κώδικα. Επίσης έχεις την δυνατότητα να δημιουργήσει κώδικα και να τον χρησιμοποιήσεις όπου και όποτε το επιθυμείς κάθε φορά που φτιάχνει κάποιος ένα καινούργιο έργο.

Τέλος, αξίζει να αναφέρουμε και μερικά ακόμα χαρακτηριστικά του Visual Studio τα οποία έχουν ενδιαφέρον. Η εφαρμογή περιλαμβάνει μια σειρά από οπτικούς σχεδιαστές που βοηθούν στην ανάπτυξη των εφαρμογών. Τα εργαλεία αυτά περιλαμβάνουν το Windows Forms Designer. Το Windows Forms Designer χρησιμοποιείται για τη δημιουργία GUI εφαρμογών που χρησιμοποιεί

Windows Forms. Έλεγχοι που εμφανίζουν δεδομένα (όπως κείμενο, πλαίσιο λίστας, προβολή πλέγματος, κλπ.) μπορεί να δεσμεύονται με πηγές δεδομένων, όπως βάσεις δεδομένων ή ερωτήματα. Ο σχεδιαστής δημιουργεί είτε C # ή VB.NET κώδικα για την εφαρμογή. Ο σχεδιαστής WPF, με την κωδική ονομασία Cider, εισήχθη με το Visual Studio 2008. Όπως και ο σχεδιαστής φορμών των Windows που υποστηρίζει το drag and drop. Χρησιμοποιείται για τη συγγραφή των user interfaces, με στόχο το Windows Presentation Foundation. Υποστηρίζει όλες τις λειτουργίες WPF, συμπεριλαμβανομένων των συνδέσεων δεδομένων και την αυτόματη διαχείριση της διάταξης. Παράγει XAML κώδικα για το UI. Το παραγόμενο XAML αρχείο είναι συμβατό με το Microsoft Expression Design. Ο κωδικός XAML συνδέεται με τον κωδικό χρησιμοποιώντας ένα κωδικό πίσω από το μοντέλο. Το Visual Studio περιλαμβάνει επίσης έναν επεξεργαστή website και σχεδιαστή που επιτρέπει την σχεδίαση ιστοσελίδων με μεταφορά και απόθεση widgets. Χρησιμοποιείται για την ανάπτυξη ASP.NET εφαρμογών και υποστηρίζει HTML, CSS και JavaScript. Χρησιμοποιεί έναν κώδικα για να συνδέεται με το ASP.NET. Από το Visual Studio 2008 και μετά, η διάταξη που χρησιμοποιείται από τον web designer, είναι από κοινού με το Microsoft Expression Web. Ένα άλλο σημαντικό εργαλείο που διαθέτει το πρόγραμμα είναι το designer Class, το οποίο χρησιμοποιείται από τον συγγραφέα ο οποίος μπορεί να επεξεργαστεί τις τάξεις (συμπεριλαμβανομένων των μελών της και την πρόσβασή τους) με τη χρήση του μοντέλου UML. Ο σχεδιαστής μπορεί να δημιουργήσει κώδικα κλάσης C # και VB.NET , και να ορίσει τις τάξεις και τις μεθόδους. Μπορεί επίσης να δημιουργήσει διαγράμματα κλάσεων από τις κατηγορίες χειρόγραφα. Τέλος διατίθεται ένας σχεδιαστής δεδομένων ο οποίος μπορεί να χρησιμοποιηθεί για να επεξεργαστείτε γραφικά σχήματα βάσεων δεδομένων, συμπεριλαμβανομένων των δακτυλογραφημένων πινάκων, των πρωτεύον και των ξένων κλειδιών και των περιορισμών. Μπορεί επίσης να χρησιμοποιηθεί για το σχεδιασμό ερωτημάτων από την προβολή γραφικών.

5.3.9 Windows Form Application (WinForms)

Οι WinForms εφαρμογές χρησιμοποιούνται για την παροχή επιχειρηματικών εφαρμογών και εργαλείων που είναι ενσωματωμένα στην πλατφόρμα των Windows. Συνήθως επιλέγουμε WinForms εφαρμογή όταν πρέπει να δημιουργηθεί μια λύση που να αξιοποιήσει τους πόρους της μηχανής του χρήστη. Αυτό σημαίνει ότι η εφαρμογή είναι εγκατεστημένη και οι χρήστες προσδοκούν να εκτελεστεί με μεγαλύτερη απόκριση από την τυπική Web εφαρμογή. Οι WinForms εφαρμογές μπορούν να είναι ανεξάρτητες ή με βάση δεδομένων (συχνά client-server). Οι WinForms εφαρμογές ενδέχεται να συνδεθούν με υπηρεσίες ιστού και να λειτουργήσουν και στις δύο συνδεδεμένα και μη συνδεδεμένα σενάρια.

Το Visual Studio παρέχει ένα ώριμο, πλούσιο σε χαρακτηριστικά σύνολο εργαλείων για την ταχεία ανάπτυξη των Windows εφαρμογών που περιλαμβάνει φόρμα σχεδίασης drag-and-drop και πολλά στοιχεία ελέγχου μέσα στην εργαλειοθήκη της φόρμας. Με αυτά τα εργαλεία, οι προγραμματιστές μπορούν να δημιουργήσουν γρήγορα μια Windows εφαρμογή που περιλαμβάνει μενού, γραμμές εργαλείων, πρόσβαση σε δεδομένα, καρτέλες, υποστήριξη αλλαγής μεγέθους, κοινά στοιχεία ελέγχου για την εργασία και εκτύπωση αρχείων και πολλά άλλα.

5.3.10 Crystal Reports

Το Crystal Reports είναι ένας δημοφιλής πρόγραμμα δημιουργίας αναφορών που βασίζεται στα και επιτρέπει σε έναν προγραμματιστή να δημιουργεί αναφορές από μια ποικιλία πηγών δεδομένων με ελάχιστο γραπτό κώδικα. Αναπτύχθηκε από το λογισμικό Seagate, το Crystal Reports μπορεί να έχει πρόσβαση σε δεδομένα από τις πιο ευρέως χρησιμοποιούμενες βάσεις δεδομένων και μπορεί να ενσωματώνει δεδομένα από πολλαπλές βάσεις δεδομένων μέσα σε μία αναφορά χρησιμοποιώντας Open Database Connectivity (ODBC).

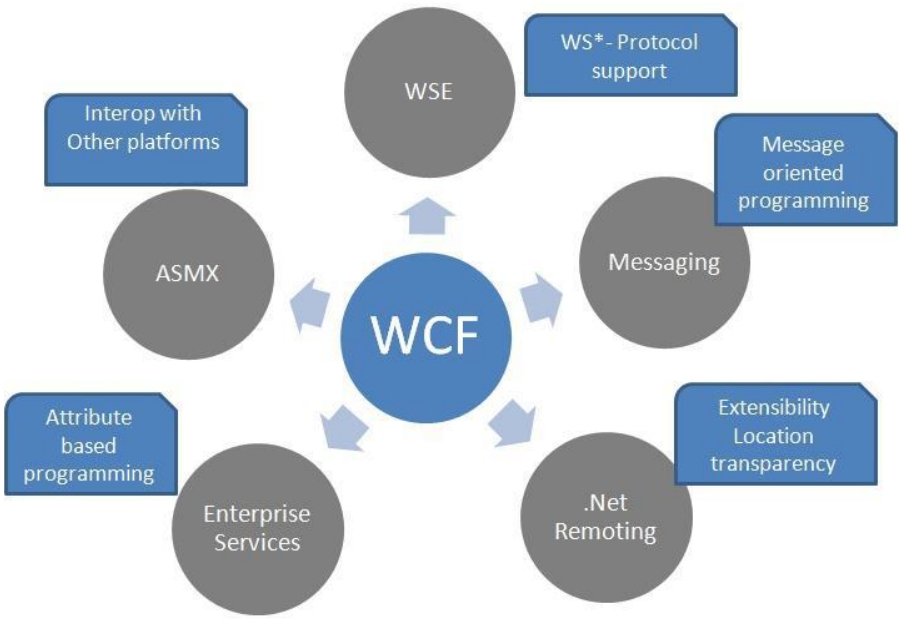
Το Crystal Reports χρησιμοποιεί ένα στοιχείο ελέγχου ActiveX που ονομάζεται CrystalReport για να δημιουργήσει μια σύνδεση με ένα άλλο πρόγραμμα. Ένας προγραμματιστής μπορεί να ορίσει ιδιότητες του ελέγχου CrystalReport κατά τη διάρκεια του σχεδιασμού ή του χρόνου εκτέλεσης.

Ο προγραμματιστής μπορεί να χρησιμοποιήσει εργαλεία αυτοματοποίησης που ονομάζονται Experts για να καθοδηγούνται μέσω κοινών εργασιών, όπως η σύνδεση και η ενσωμάτωση αναφορών. Το Crystal Reports αντιμετωπίζει όλα τα πεδία κειμένου, γραφικών και βάσεων δεδομένων ως αντικείμενα που ένας προγραμματιστής μπορεί να τοποθετήσει, να οργανώσει και να μορφοποιήσει σε φόρμες. Το πρόγραμμα δημιουργεί επίσης ένα αντικείμενο δέσμης εγγραφών και έναν κώδικα που απαιτείται για την εκτέλεση εργασιών προγραμματισμού, όπως βρόχοι ή μαθηματικοί υπολογισμοί.

Το Crystal Reports μπορεί να δημιουργήσει μια αναφορά σε κίνηση από μεταβλητές που έχουν οριστεί από τον χρήστη και να το μετατρέψει σε HTML και να το δημοσιεύσει αυτόματα στον Ιστό.

5.3.11 WCF Service

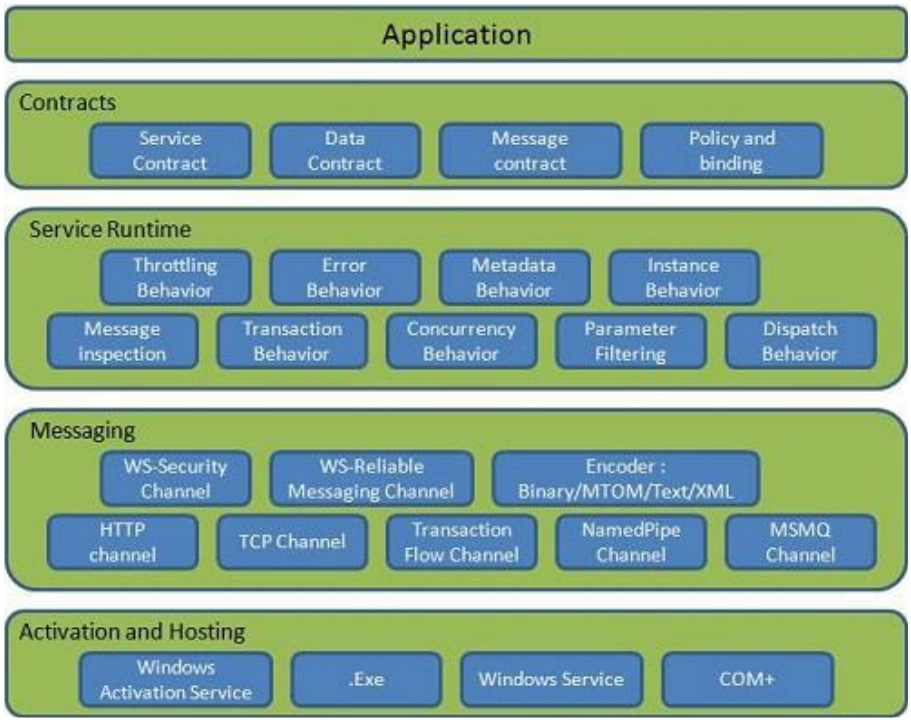
Το WCF είναι ένα εργαλείο ανάπτυξης λογισμικού για την ανάπτυξη και την βελτίωση υπηρεσιών σε Windows. Το WCF είναι ένα εξελιγμένο .NET framework. Παρέχει ένα περιβάλλον χρόνου εκτέλεσης για τις υπηρεσίες του χρήστη, επιτρέποντάς τον να εκθέσει Common Language Runtime (CLR) τύπους ως υπηρεσίες και να καταναλώσει άλλες υπηρεσίες ως CLR τύπους. Παρόλο που θεωρητικά θα μπορούσαν να χτιστούν υπηρεσίες χωρίς WCF, στην πράξη το χτίσιμο των υπηρεσιών είναι σημαντικά ευκολότερο με το WCF. Το WCF είναι εφαρμογή ενός συνόλου βιομηχανικών προτύπων της Microsoft που ορίζουν τις αλληλεπιδράσεις υπηρεσιών, τους τύπους μετατροπής, ταξινόμησης και διαχείρισης διάφορων πρωτοκόλλων. Συνεπώς, το WCF παρέχει δια λειτουργικότητα μεταξύ υπηρεσιών.



Το WCF παρέχει στους προγραμματιστές τα βασικά εργαλεία που απαιτούνται από όλες σχεδόν τις εφαρμογές, και ως εκ τούτου, αυτό αυξάνει σημαντικά την παραγωγικότητα. Η πρώτη έκδοση του WCF (ως μέρος του .NET 3.0) παρείχε πολλές χρήσιμες διευκολύνσεις για την ανάπτυξη υπηρεσιών, όπως φιλοξενία, διαχείριση υπηρεσιών παρουσίας, ασύγχρονες κλήσεις, αξιοπιστία, διαχείριση συναλλαγών, αποσυνδεδεμένες κλήσεις σε ουρές και ασφάλεια. Η δεύτερη έκδοση του WCF (ως μέρος του .NET 3.5) παρείχε πρόσθετα εργαλεία και επεκτείνει την αρχική έκδοση με πρόσθετες επιλογές επικοινωνίας. Στην τρίτη έκδοση (ως μέρος του .NET 4.0) περιλαμβάνονται αλλαγές διαμόρφωσης, μερικές επεκτάσεις και τα νέα χαρακτηριστικά ανακάλυψης και δρομολογητές. Με την τέταρτη έκδοση του WCF (ως μέρος του .NET 4.5), το WCF διαθέτει αρκετές νέες δυνατότητες απλοποίησης και πρόσθετες συνδέσεις, συμπεριλαμβανομένων των δεσμών UDP και WebSocket. Το WCF διαθέτει ένα κομψό μοντέλο επεκτασιμότητας που μπορείτε να χρησιμοποιηθεί για να εμπλουτίσει τη βασική έκδοση. Στην πραγματικότητα, το ίδιο το WCF γράφεται χρησιμοποιώντας αυτό το μοντέλο επεκτασιμότητας.

Το WCF είναι μέρος του .NET 4.5, επομένως μπορεί να εκτελείται μόνο σε λειτουργικά συστήματα που το υποστηρίζουν. Επί του παρόντος, αυτή η λίστα αποτελείται από τα Windows XP και μεταγενέστερα και τον Windows Server 2003 και νεότερα.

Οι περισσότερες λειτουργίες WCF περιλαμβάνονται σε μια συνδεσμολογία που ονομάζεται System.Service- Model.dll, που βρίσκεται στο χώρο ονομάτων System.ServiceModel.



Το WCF εφαρμόζει μία αρχιτεκτονική multi-layer, όπως φαίνεται στην παραπάνω εικόνα. Συγκεκριμένα, περιλαμβάνει τα εξής:

α) Contracts Layer: Αφορά την συμφωνία μεταξύ δύο μηχανών, η οποία ορίζει τους όρους και προϋποθέσεις για τα μηνύματα που θα ανταλλάσσονται. Περιλαμβάνει τα εξής είδη contract:

i) Data Contract: Ορίζει την δομή των δεδομένων και τις παραμέτρους που χρησιμοποιούνται από τα services, προκειμένου να πραγματοποιηθεί επιτυχώς αλληλεπίδραση του service με τον client.

ii) Message Contract: Παρέχει έλεγχο στα μηνύματα SOAP που παράγονται και καταναλώνονται από το WCF. Επίσης, παρέχουν απευθείας πρόσβαση στο σώμα και τις επικεφαλίδες του μηνύματος.

iii) Service Contract: Ορίζει το είδος της λειτουργίας που υποστηρίζει η υπηρεσία. Εκθέτει πληροφορίες στον client, όπως ο τύπος δεδομένων που έχει το μήνυμα, η τοποθεσία που βρίσκονται οι μέθοδοι, οι πληροφορίες του πρωτοκόλλου που εφαρμόζεται και η δομή serialization κ.α. Ουσιαστικά, ορίζονται οι μέθοδοι της υπηρεσίας.

iv) Policy και Binding: Παρέχει πληροφορίες σχετικές με την ασφάλεια και το πρωτόκολλο.

β) Service Runtime Layer: Ορίζει και επεξεργάζεται διάφορες συμπεριφορές της υπηρεσίας, οι οποίες συμβαίνουν κατά την λειτουργία της υπηρεσίας. Ονομάζονται και service behaviors και ουσιαστικά αποτελούν αντικείμενα που επηρεάζουν τα χαρακτηριστικά της WCF υπηρεσίας.

γ) Messaging Layer: Είναι υπεύθυνο για την δομή των μηνυμάτων που ανταλλάσσονται.

δ) Hosting και Application Layer: Παρέχει μεγάλο αριθμό επιλογών σχετικά με την ενεργοποίηση και φιλοξενία της υπηρεσίας.

Πλεονεκτήματα

Τα πλεονεκτήματα του WCF Framework είναι τα εξής:

α) Διαλειτουργικότητα με άλλες υπηρεσίες .

β) Καλύτερη αξιοπιστία και ασφάλεια σε σύγκριση με τις Web Services ASMX

γ) Δεν απαιτούνται μεγάλες αλλαγές στον κώδικα, προκειμένου να εφαρμοστεί ένα μοντέλο ασφάλειας και τροποποίηση των απαιτούμενων συνδέσεων (bindings).

δ) Φέρει ενσωματωμένο μηχανισμό logging.

Βασικές Έννοιες

Οι πιο βασικές έννοιες του WCF Framework είναι οι εξής:

α) Message: Αποτελεί μία μονάδα επικοινωνίας, η οποία περιλαμβάνει διάφορα τμήματα εκτός από το κυρίως σώμα. Τα μηνύματα στέλνονται και λαμβάνονται για όλους τους τύπους επικοινωνίας, ανάμεσα σε client και service.

β) Endpoint: Ορίζει την διεύθυνση, όπου το μήνυμα στέλνεται ή λαμβάνεται. Επίσης, καθορίζει τον μηχανισμό επικοινωνίας για να καθορίσει τα μηνύματα πώς θα αποστέλλονται. Μία δομή endpoint περιλαμβάνει τα ακόλουθα τμήματα:

i) Address: Αποτελεί την ακριβή τοποθεσία, που πρόκειται να παραλάβει τα μηνύματα και ορίζεται ως Unified Resource Identifier (URI). Για παράδειγμα η net.tcp://localhost:9000/ServiceA, αποτελεί μία διεύθυνση URI. Το κομμάτι «net.tcp», αφορά το σχήμα του πρωτοκόλλου TCP. Το domain είναι το «localhost», και το path είναι το «ServiceA».

ii) Binding: Ορίζει τον τρόπο επικοινωνίας ενός endpoint. Αποτελείται από στοιχεία σύνδεσης που δημιουργούν την υποδομή για την επικοινωνία και ορίζει το πρωτόκολλο επικοινωνίας που χρησιμοποιείται για την πρόσβαση στην υπηρεσία. Μερικά στοιχεία binding είναι τα basicHttpBinding, wsHttpBinding, wsDualHttpBinding κ.α.

iii) Contracts: Πρόκειται για μία συλλογή εργασιών, οι οποίες ορίζουν ποια λειτουργία θα εκθέτει το endpoint στον client.

γ) Hosting: Αναφέρεται στην φιλοξενία της WCF υπηρεσίας, η οποία μπορεί να πραγματοποιηθεί με ποικίλους τρόπους, όπως self-hosting, IIS hosting, WAS hosting κ.α.

δ) Metadata: Αυτή είναι μία σημαντική έννοια, δεδομένου ότι διευκολύνει την αλληλεπίδραση μεταξύ της εφαρμογής-client και της Web Service.

ε) WCF client: Αφορά μία εφαρμογή, η οποία δημιουργείται για να εκθέτει τις λειτουργίες WCF με την μορφή μεθόδων.

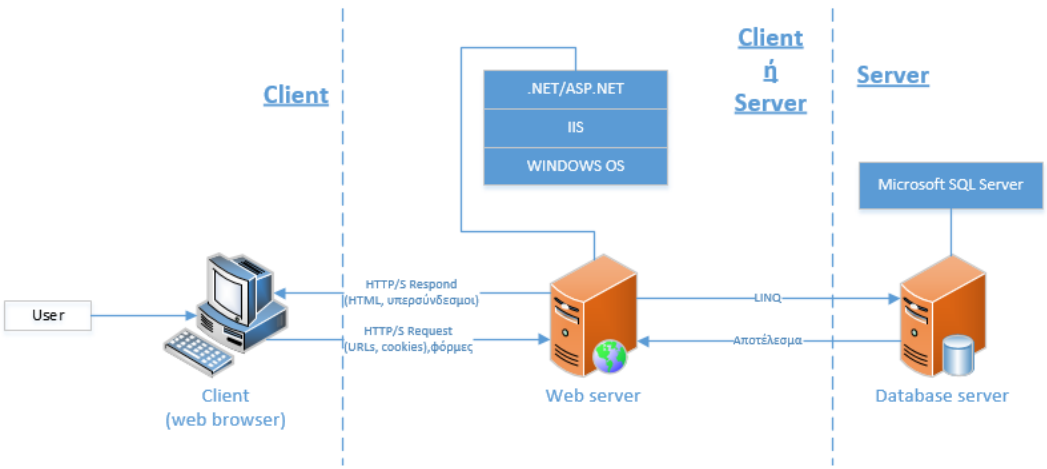
στ) Channel: Αφορά το μέσο στο οποίο ένας client επικοινωνεί με την υπηρεσία.

ζ) SOAP: Είναι ένα XML αρχείο, που αποτελείται από το σώμα και τις επικεφαλίδες.

Ένα WCF Service αποτελείται από ένα ή περισσότερα interfaces και τις κλάσεις που τα εφαρμόζουν. Τα interfaces περιλαμβάνουν τα contracts, όπως οριστήκαν παραπάνω. Στην συνέχεια, γίνεται η εφαρμογή των interfaces, όπου υλοποιείται με την χρήση μεθόδων, η λειτουργία που εκτελεί το Service. Στην συνέχεια, ορίζεται ο Host, που θα φιλοξενεί το Service και που θα το εκθέτει, προκειμένου να το εκμεταλλευτούν οι πιθανοί clients. Όλα τα απαιτούμενα βήματα για την επιτυχή δημιουργία ενός Web Service, θα εξηγηθούν με λεπτομέρεια κατά την ανάλυση του κώδικα των Projects.

5.3.12 Αρχιτεκτονική τριών επιπέδων

Η γενική αρχιτεκτονική που χρησιμοποιείται για τη σχεδίαση διαδικτυακών εφαρμογών είναι η client/server και πιο συγκεκριμένα αυτή των πολλαπλών βαθμίδων (multi-tier) Στην Εικόνα φαίνεται ένα παράδειγμα αρχιτεκτονικής σχεδίασης διαδικτυακής εφαρμογής τριών βαθμίδων Η συγκεκριμένη αρχιτεκτονική χρησιμοποιήθηκε για την υλοποίηση της εφαρμογής της παρούσας εργασίας και είναι προσανατολισμένη στο .NET Framework.



5.3.13 Internet Information Services (IIS) Server

Το Internet Information Services (IIS) είναι ένας ευέλικτος, γενικής χρήσης εξυπηρετητής ιστού από τη Microsoft που εκτελείται σε συστήματα Windows για την εξυπηρέτηση ζητούμενων σελίδων HTML ή αρχείων. Ένας εξυπηρετητής ιστού IIS δέχεται αιτήματα από υπολογιστές

απομακρυσμένων υπολογιστών-πελατών και επιστρέφει την κατάλληλη απάντηση. Αυτή η βασική λειτουργικότητα επιτρέπει στους διακομιστές ιστού να μοιράζονται και να παρέχουν πληροφορίες σε τοπικά δίκτυα, όπως εταιρικά intranets και δίκτυα ευρείας περιοχής, όπως το διαδίκτυο.

Ένας εξυπηρετητής ιστού μπορεί να παρέχει πληροφορίες σε χρήστες σε διάφορες μορφές, όπως στατικές ιστοσελίδες που κωδικοποιούνται σε HTML. μέσω ανταλλαγών αρχείων όπως λήψεις και μεταφορτώσεις. και έγγραφα κειμένου, αρχεία εικόνας και πολλά άλλα.

Το IIS λειτουργεί μέσω μιας ποικιλίας προτύπων γλωσσών και πρωτοκόλλων. Το HTML χρησιμοποιείται για τη δημιουργία στοιχείων όπως κείμενο, κουμπιά, τοποθετήσεις εικόνας, άμεσες αλληλεπιδράσεις / συμπεριφορές και υπερσυνδέσεις. Το πρωτόκολλο μεταφοράς υπερκειμένου (HTTP) είναι το βασικό πρωτόκολλο επικοινωνίας που χρησιμοποιείται για την ανταλλαγή πληροφοριών μεταξύ εξυπηρετητών ιστού και χρηστών. HTTPS - HTTP μέσω Secure Sockets Layer (SSL) - χρησιμοποιεί το Transport Layer Security ή SSL για να κρυπτογραφήσει την επικοινωνία για πρόσθετη ασφάλεια δεδομένων. Το Πρωτόκολλο μεταφοράς αρχείων ή η ασφαλή παραλλαγή του, FTPS, μπορεί να μεταφέρει αρχεία.

Τα πρόσθετα πρωτόκολλα που υποστηρίζονται περιλαμβάνουν το πρωτόκολλο μεταφοράς απλής αλληλογραφίας, την αποστολή και λήψη μηνυμάτων ηλεκτρονικού ταχυδρομείου και το πρωτόκολλο μεταφοράς ειδήσεων δικτύου, για την παράδοση άρθρων στο Usenet.

Η Microsoft παρέχει μια αυτόνομη έκδοση του IIS, που ονομάζεται IIS Express, για προγραμματιστές που δοκιμάζουν ιστοσελίδες. Η υπηρεσία IIS Express προσφέρει όλες τις μεγάλες δυνατότητες του πλήρους εξυπηρετητή ιστού IIS, αλλά επιτρέπει πολλές εργασίες να εκτελούνται χωρίς δικαιώματα διαχειριστή.

Οι πληροφορίες για την ρύθμιση του IIS server είναι αποθηκευμένες σε Extensible Markup Language (XML) αρχεία. Τα δύο αρχεία που ελέγχουν τις ρυθμίσεις του είναι τα applicationhost.config και web.config. Μέσω του web.config ελέγχεται η διαμόρφωσή του σε επίπεδο εφαρμογής ενώ μέσω του applicationhost.config ελέγχεται αποκλειστικά αυτός. Οι ρυθμίσεις διαμόρφωσης κληρονομούνται, άρα οι ρυθμίσεις που καθορίζονται στο web.config μπορούν να παρακάμψουν τις ρυθμίσεις του υψηλότερου επιπέδου, δηλαδή του αρχείου applicationhost.config. Το αρχείο applicationhost.config βρίσκεται στο κατάλογο %windir%\system32\inetsrv\config ενώ τα web.config στους υποκαταλόγους της εφαρμογής

5.3.14 Language Integrated Query (LINQ)

Τα ερωτήματα που γίνονται σε βάση δεδομένων μέσω του πλαισίου Entity Framework είναι γραμμένα σε γλώσσα LINQ. Η συγκεκριμένη γλώσσα είναι στενά συνδεδεμένη με το .NET Framework και αποτελεί μια ισχυρά τυποποιημένη γλώσσα ερωτημάτων βάσεων δεδομένων. Λέγοντας ότι είναι ισχυρά τυποποιημένη εννοείται ότι τα ερωτήματα καθορίζονται χρησιμοποιώντας τις κλάσεις που υπάρχουν στο μοντέλο δεδομένων της εφαρμογής.

6 Επίλογος

6.1 Σύνοψη και Συμπεράσματα

Ο Ηλεκτρονικός Ιατρικός Φάκελος αποτελεί σημαντικό εργαλείο για όλους τους φορείς Υγείας. Παρέχει άμεσα και γρήγορα, με ακρίβεια, πρακτικά και εύχρηστα σημαντικές πληροφορίες που πολλές φορές μπορεί να σώσουν και τη ζωή του ασθενούς.

Η όλη πορεία της υγείας του ασθενούς θα παρακολουθείται μέσω του ηλεκτρονικού συστήματος, που θα είναι συνδεδεμένο με όλη την επικράτεια. Ο Ηλεκτρονικός Ιατρικός Φάκελος Ασθενούς θα ακολουθεί τον ασθενή ανεξάρτητα από ποια μονάδα υγείας και σε ποια πόλη θα επισκεφτεί. Ο Φάκελος θα περιλαμβάνει όλες τις επισκέψεις, που έχει πραγματοποιήσει σε γιατρούς και μονάδες ανά την χώρα, τα φάρμακα που λαμβάνει και τις εξετάσεις που έχει κάνει. Θα περιγράφει αναλυτικά το ιστορικό του ασθενή με αποτέλεσμα να υπάρχει συνέχεια στην ιατρική παρακολούθηση.

6.2 Μελλοντικές Επεκτάσεις

Προφανώς και η υλοποίηση μιας τέτοιας εφαρμογής είναι αρκετά περίπλοκη όχι μόνο από τεχνικής σκοπιάς αλλά και από επιχειρησιακής σκοπιάς. Παρόλα αυτά, η προσέγγιση που παρουσιάζεται στην παρούσα εργασία δείχνει ότι τα τεχνολογικά ζητήματα είναι εφικτό να αντιμετωπιστούν αποτελεσματικά με την υιοθέτηση διεθνών προτύπων, βέλτιστων πρακτικών και αρχιτεκτονική πολλαπλών επιπέδων (multi-tier architecture).

Παρά το γεγονός ότι στο πλαίσιο της εργασίας αυτής, δεν περιορίστηκε στο σχεδιαστικό μέρος αλλά προχώρησε και στην υλοποίηση του έργου (software development), είναι σαφές ότι το ζήτημα δεν είναι δυνατόν να εξαντληθεί μέσω της εργασίας αυτής. Κατά συνέπεια, υπάρχουν αρκετά ζητήματα που θα μπορούσαν να μελετηθούν ως μελλοντικές επεκτάσεις της παρούσας εργασίας.

7 Βιβλιογραφία

[1] Software Engineering 9th Edition By Ian Sommerville
[2] ASP.NET and Web Programming (University College of Southeast Norway)
[3] ADO.NET in a Nutshell By Matthew MacDonald, Bill Hamilton
[4] Microsoft ADO.NET Step By Step By Rebecca M. Riordan
[5] Microsoft Visual Studio 2015 Unleashed 3rd Edition By Lars Powers, Mike Snell
[6] Crystal Report With Visual Studio .NET
[7] Programming WCF Services 4th Edition By Juval Lowy & Michael Montgomery
[8] Professionl Microsoft IIS 8 By Ken Schaefer, Jeff Cochram, Scott Forysth
[9] Learning SQL By Alan Beaulieu
[10] Database System Concepts By Abraham Silberschtz, Henry F. Korth, S. Sudarsham
[11] C# 6.0 And The .NET 4.6 Framework 7th Edition By Andrew Troelsen, Philip Japikse
[12] C# Programming Yellow Book By Rob Miles “Bananas” 7th Edition
[13] C# 6.0 in a Nutshell 6th Edition Joseph Albahari & Ben Albahari