Πανεπιστήμιο Πειραιώς – Τμήμα Πληροφορικής
Πρόγραμμα Μεταπτυχιακών Σπουδών
«Πληροφορική»

Μεταπτυχιακή Διατριβή

Τίτλος Διατριβής	**Δημιουργία συστήματος ηλεκτρονικού ιατρικού φακέλου με εμφάνιση εκτιμήσεων για κληρονομικές ασθένειες** **Design and development of electronic medical record of a patient with presentation of assessments for hereditary diseases, providing statistics with charts**
Ονοματεπώνυμο Φοιτητή	**Ιωάννης Μπεημανάβης**
Πατρώνυμο	**Κωνσταντίνος**
Αριθμός Μητρώου	**ΜΠΠΛ-14055**
Επιβλέπων	**Ευθύμιος Αλέπης, Επίκουρος Καθηγητής**

Ημερομηνία Παράδοσης **Φεβρουάριος 2017**

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ «ΠΛΗΡΟΦΟΡΙΚΗ»

ΔΗΜΙΟΥΡΓΙΑ ΣΥΣΤΗΜΑΤΟΣ ΗΛΕΚΤΡΟΝΙΚΟΥ ΙΑΤΡΙΚΟΥ ΦΑΚΕΛΟΥ ΜΕ ΕΜΦΑΝΙΣΗ ΕΚΤΙΜΗΣΕΩΝ ΓΙΑ ΚΛΗΡΟΝΟΜΙΚΕΣ ΑΣΘΕΝΕΙΕΣ

ΜΕΤΑΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ
του
Ιωάννη Μπεημανάβη

ΑΜ: ΜΠΠΛ-14055
Email: beimanavis@gmail.com
Τηλέφωνο Επικοινωνίας: 6947600699

Αλέπης Ευθύμιος, Επιβλέπων Καθηγητής

Εγκρίθηκε από την Τριμελή Εξεταστική Επιτροπή

(υπογραφή)	**(υπογραφή)**	**(υπογραφή)**
Όνομα Επώνυμο Βαθμίδα	**Όνομα Επώνυμο Βαθμίδα**	**Όνομα Επώνυμο Βαθμίδα**

Πειραιάς, Φεβρουάριος 2017

© 2017 – Με επιφύλαξη παντός δικαιώματος. All rights reserved.

Απαγορεύεται η αντιγραφή, αποθήκευση και διανομή της παρούσας εργασίας, εξ' ολοκλήρου ή τμήματος αυτής, για εμπορικό σκοπό. Επιτρέπεται η ανατύπωση, αποθήκευση και διανομή για σκοπό μη κερδοσκοπικό, εκπαιδευτικής ή ερευνητικής φύσης, υπό την προϋπόθεση να αναφέρεται η πηγή προέλευσης και να διατηρείται το παρόν μήνυμα. Ερωτήματα που αφορούν στη χρήση της εργασίας πρέπει να απευθύνονται προς τον επιβλέπων καθηγητή ή στον συγγραφέα.

Οι απόψεις και τα συμπεράσματα που περιέχονται σε αυτό το έγγραφο εκφράζουν τον συγγραφέα και δεν πρέπει να ερμηνευτεί ότι αντιπροσωπεύουν τις επίσημες θέσεις του Πανεπιστημίου Πειραιώς.

ΕΥΧΑΡΙΣΤΙΕΣ

Σε αυτό το σημείο, με τη λήξη του ακαδημαϊκού κύκλου, θα ήθελα να ευχαριστήσω θερμά όλους όσους στάθηκαν στο πλευρό μου και με στήριξαν καθ' όλη την διάρκεια της προσπάθειάς μου να ανταπεξέλθω στις απαιτήσεις του Προγράμματος Μεταπτυχιακών Σπουδών.

Πρώτα από όλους οφείλω ένα μεγάλο ευχαριστώ στην γυναίκα μου, στη μητέρα και την αδερφή μου, για την αμέριστη αγάπη τους, τη συνεχή κατανόηση, την ψυχολογική και ηθική συμπαράσταση που μου παρείχαν όλα αυτά τα χρόνια.

Τέλος, θα ήθελα επίσης να ευχαριστήσω θερμά τους καθηγητές μου, για την εμπιστοσύνη και την υποστήριξή τους, την άψογη συνεργασία που είχαμε και για την συμβολή τους στη διεύρυνση των γνώσεων που μου προσφέρανε.

ΠΕΡΙΛΗΨΗ

Είναι γεγονός ότι ο Ηλεκτρονικός Ιατρικός Φάκελος Ασθενούς αποτελεί μία σύγχρονη εφαρμογή πληροφοριακού συστήματος στο τομέα της υγείας, καθώς στην πράξη είναι επιτακτική ανάγκη η συγκέντρωση όσο το δυνατόν μεγαλύτερου όγκου πληροφοριών για την πληρέστερη εικόνα της υγείας του ασθενούς.

Η παρούσα διπλωματική εργασία ασχολείται με την επέκταση των λειτουργιών και των δυνατοτήτων του παραδοσιακού ιατρικού φακέλου.

Με βάση τα παραπάνω, γίνεται σχεδιασμός και ανάπτυξη ενός τέτοιου ΗΙΦΑ με αξιοποίηση σύγχρονων τεχνολογιών και εργαλείων βάσεων δεδομένων και διαδικτύου. Η σημαντικότητα της εφαρμογής έγκειται στο γεγονός ότι τα στοιχεία αποθηκεύονται σε μία βάση δεδομένων με ευκολία πρόσβασης ανά πάσα στιγμή από τον Γιατρό, την επεξεργασία τους για την εξαγωγή σωστών συμπερασμάτων σε ότι αφορά την εικόνα των παρελθόντων ιατρικών εξετάσεων και των μελλοντικών ενεργειών που αφορούν τον ασθενή. Επίσης δόθηκε η δυνατότητα άντλησης πληροφοριών στατιστικής φύσεως στο σύνολο των καταχωρημένων ασθενών, αλλά και για την κληρονομικότητα κάποιων νοσημάτων.

Με την οργάνωση των κεφαλαίων της εργασίας, παρουσιάζεται αρχικά το θεωρητικό υπόβαθρο με ένα τρόπο συνοπτικό της προσέγγισης του θέματος από ιατρική αλλά και τεχνολογική πλευρά. Έτσι, η εργασία αυτή αρχικά περιγράφει τον ρόλο του Ιατρικού Φακέλου Ασθενούς, γίνεται μία σύντομη ιστορική αναδρομή και τονίζεται η σημασία εφαρμογής του σε ηλεκτρονική μορφή. Στην συνέχεια, γίνεται αναφορά στα πρότυπα που υπάρχουν για τον ηλεκτρονικό ιατρικό φάκελο και για τα Πληροφοριακά Συστήματα παροχής Υπηρεσιών Υγειάς που υπάρχουν στην αγορά. Επιπλέον, παρουσιάζονται και αναλύονται έρευνες πάνω σε ασθένειες που είναι κληρονομικές, ώστε λαμβάνοντας υπόψιν τα δεδομένα αυτά, να γίνει εκτίμηση για το αν κάποιος ασθενής θα νοσήσει προβάλλοντας στατιστικά σε διαγράμματα. Τέλος σχεδιάζεται και υλοποιείται ένα σύστημα Ηλεκτρονικού Ιατρικού Φακέλου Ασθενούς στον τομέα της Ιατρικής Επιστήμης, καθορίζονται οι απαιτήσεις δεδομένων για τη σχεδίαση και δημιουργία μιας βάσης δεδομένων και παρουσιάζονται οι γλώσσες προγραμματισμού, οι τεχνολογίες καθώς και τα εργαλεία που χρησιμοποιούνται και εφαρμόζονται.

Η εφαρμογή λειτουργεί σε διαδικτυακό περιβάλλον. Απαιτούνται όνομα χρήστη και κωδικός πρόσβασης για είσοδο και η ακεραιότητα της εφαρμογής επιτυγχάνεται με την χρήση session cookies.

ABSTRACT

It is a fact that the Electronic Medical Record Patient is a contemporary implementation of informational system in the health sector, and in practice it is of utmost importance to collect the largest possible amount of information for a clearest insight of the patient's condition.
This paper deals with the expansion of the functions and potential of a traditional medical record. Considering the above information, an EMRP is being designed and developed using modern technologies, databases and web tools. The importance of this application consists in the fact that the data of the medical record are stored in a database providing an easy accessibility at any time the Doctor need them for processing and coming to the right conclusions with regard past history-examinations and future decisions-policies. At the same time it provides the ability to pump statistical information concerning the registered patients, and also regarding the inheritance of some diseases.
Organizing the chapters of the current project, it is primarily and briefly presented the approach of the issue from a medical and technological aspect. Thus, this paper first of all describes the role of the Medical Record of the Patient continues with a brief historical review and emphasizes the importance of the electronic record. Next, there is a reference on the current standards for the Electronic Medical Records and the Information Healthcare Systems on the market nowadays. Furthermore, there are presented researches on diseases which are inherited, so as considering these data, the doctor can assess the possibility of a patient to become ill, providing statistics in charts. Finally an Electronic Health Record System of Patient is designed and implemented in the field of Medical Science, the data requirements for the design and creation of a database are determined and finally the programming languages, the technologies and the tools used and applied are presented.
The application is designed to work in the online web environment. It demands username and password to login and the integrity of the application is achieved by using session cookies.

ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

1 ΕΙΣΑΓΩΓΗ 11
1.1 Γενικά 11
1.2 Αντικείμενο Διπλωματικής 11
1.3 Συνοπτική Περιγραφή 12
2 ΙΑΤΡΙΚΟΣ ΦΑΚΕΛΟΣ ΑΣΘΕΝΗ 14
2.1 Εισαγωγή 14
2.2 Ιατρική Πληροφορία 14
2.3 Ιατρικός Φάκελος 15
2.4 Σκοποί Ιατρικού Φακέλου 16
2.5 Ιστορική Αναδρομή 16
2.6 Σήμερα 18
2.7 Ο Ηλεκτρονικός Ιατρικός Φάκελος στην Ελλάδα 19
2.8 Paper-based 20
2.9 Η έννοια του Ηλεκτρονικού Ιατρικού Φακέλου Ασθενούς 21
3 ΙΣΤΟΡΙΚΟ ΤΟΥ ΑΡΡΩΣΤΟΥ 23
3.1 Εισαγωγή 23
3.2 Λήψη Ιατρικού Ιστορικού 23
3.3 Προσέγγιση Αρρώστου 24
3.4 Τυποποίηση Πληροφοριών Ιστορικού Ασθενούς 24
3.4.1 Τα στοιχεία ταυτότητας 24
3.4.2 Φαρμακευτική αγωγή 24
3.4.3 Αλλεργίες 24
3.4.4 Κοινωνικό ιστορικό-Συνήθειες 25
3.4.5 Ατομικό αναμνηστικό-Νοσήματα/Χειρουργικές επεμβάσεις 25
3.4.6 Οικογενειακό ιστορικό 25
3.4.7 Κύριο ενόχλημα – Συμπτώματα εισόδου 25
3.4.8 Παρούσα νόσος-Αρχική αξιολόγηση 26
3.4.9 Κλινική εξέταση 26
3.4.10 Διάγνωση 26
4 ΠΡΟΤΥΠΑ ΗΛΕΚΤΡΟΝΙΚΟΥ ΙΑΤΡΙΚΟΥ ΦΑΚΕΛΟΥ 27
4.1 Εισαγωγή 27
4.2 CEN/TC 251 27

4.3 ISO/TC 215 ... 27
4.4 HL7 ... 28
5 ΠΛΗΡΟΦΟΡΙΑΚΑ ΣΥΣΤΗΜΑΤΑ ΥΓΕΙΑΣ ... 30
5.1 Εισαγωγή ... 30
5.2 Στόχοι Πληροφοριακών Συστημάτων Υγείας ... 30
5.3 Ιστορία και Εξέλιξη ... 30
5.4 Πάροχοι Ηλεκτρονικών Υπηρεσιών Υγείας ... 31
5.4.1 Microsoft HealthVault ... 31
5.4.2 Dossia ... 31
5.4.3 World Medical Card ... 31
5.5 Πληροφοριακά Συστήματα Υπηρεσιών Υγείας στην Ελλάδα ... 32
5.5.1 MEDILIG ... 32
5.5.2 Πανάκεια ... 32
5.5.3 Professional clinic ... 32
5.5.4 DoctorsNet ... 33
5.5.5 Σειρά Συστημάτων SmartHealth της Apollo Information Technologies ... 34
5.5.6 TMS Iasis ... 34
5.5.7 Vida24 ... 36
6 ΚΛΗΡΟΝΟΜΙΚΟΤΗΤΑ ΑΣΘΕΝΙΩΝ ... 37
6.1 Εισαγωγή ... 37
6.2 Σακχαρώδης Διαβήτης τύπου 2 ... 37
6.3 Θυρεοειδής ... 37
6.4 Οστεοπόρωση ... 38
6.5 Μεσογειακή Αναιμία ... 39
7 ΑΝΑΠΤΥΞΗ ΠΛΗΡΟΦΟΡΙΑΚΟΥ ΣΥΣΤΗΜΑΤΟΣ ... 41
7.1 Εισαγωγή ... 41
7.2 Ανάλυση Προδιαγραφών - Παραδοχές ... 41
7.3 Σχεδιασμός – Διαγράμματα UML ... 42
7.3.1 Διαγράμματα Περιπτώσεων Χρήσης ... 43
7.3.2 Διαγράμματα Δομής Υλοποίησης ... 48
7.4 Υλοποίηση ΠΣ ... 50
7.4.1 Εισαγωγή ... 50

7.4.2 Η γλώσσα προγραμματισμού c# και το περιβάλλον .NET 50
7.4.3 Το περιβάλλον ανάπτυξης Microsoft Visual Studio 50
7.4.4 ASP .net 50
7.4.5 ASP .net MVC 51
7.4.6 Πλεονεκτήματα MVC-based Web εφαρμογής 51
7.4.7 Entity Framework 52
7.4.8 Δομή Κώδικα 53
7.4.9 Bootstrap 53
7.4.10 Linq 53
7.4.11 Αρχιτεκτονική Υλοποίησης - GUI 54
8 ΒΑΣΗ ΔΕΔΟΜΕΝΩΝ 57
8.1 Εισαγωγή 57
8.2 Γενική Περιγραφή ΒΔ 57
8.3 Ανάλυση Απαιτήσεων ΒΔ 58
8.4 Εννοιολογικό Μοντέλο Οντοτήτων-Σχέσεων (E-R) 59
8.4.1 Γενικά 59
8.4.2 Υλοποίηση Διαγράμματος E-R στην ΒΔ 60
8.5 Σχεσιακό Μοντέλο Δεδομένων 60
8.5.1 Γενικά 60
8.5.2 Υλοποίηση Σχεσιακού Μοντέλου της ΒΔ 61
8.6 Παρουσίαση Αντικειμένων ΒΔ 62
8.6.1 Tables 62
8.6.2 Views 68
8.6.3 Procedures 69
9 ΠΑΡΟΥΣΙΑΣΗ ΕΦΑΡΜΟΓΗΣ 79
9.1 Σύνδεση Χειριστή 79
9.2 Λίστα Ασθενών και Καταχώρηση Νέου 82
9.3 Καρτέλες Ασθενούς 88
9.4 Λίστα Επισκέψεων και Προσθήκη Νέας 94
9.5 Καρτέλες Επίσκεψης 97
9.6 Σχέσεις Συγγένειας 99
9.7 Στατιστικά για τα Καταχωρημένα Άτομα 100
10 ΣΥΜΠΕΡΑΣΜΑΤΑ 102

10.1 Σύνοψη και Παρουσίαση Συμπερασμάτων 102
10.2 Μελλοντικές Επεκτάσεις 102
11 ΕΓΚΑΤΑΣΤΑΣΗ ΣΥΣΤΗΜΑΤΟΣ 104
12 ΕΓΧΕΙΡΙΔΙΟ ΧΡΗΣΗΣ ΕΦΑΡΜΟΓΗΣ 105
12.1 Σύνδεση Χειριστή 105
12.2 Λίστα Ασθενών και Καταχώρηση Νέου 110
12.3 Καρτέλες Ασθενούς 114
12.4 Λίστα Επισκέψεων και Προσθήκη Νέας 119
12.5 Καρτέλες Επίσκεψης 122
12.6 Σχέσεις Συγγένειας 126
12.7 Λίστα Συμπτωμάτων και Προσθήκη Νέου 127
12.8 Λίστα Νοσημάτων και Προσθήκη Νέου 127
12.9 Στατιστικά για τα Καταχωρημένα Άτομα 128
13 ΠΑΡΑΡΤΗΜΑΤΑ 133
13.1 Παράρτημα Α’ 133
13.2 Παράρτημα Β’ 169
13.3 Παράρτημα Γ’ 171
14 ΒΙΒΛΙΟΓΡΑΦΙΑ 173

1 ΕΙΣΑΓΩΓΗ

1.1 Γενικά

Η ραγδαία εξέλιξη της τεχνολογίας σε συνδυασμό με την ανάγκη για ευκολία πρόσβασης στην πληροφορία, οδήγησε στη δημιουργία Πληροφοριακών Συστημάτων. Τα τελευταία χρόνια παρατηρείται στο χώρο της υγείας, η απαίτηση για αναβάθμιση στην ποιότητα των προσφερόμενων υπηρεσιών. Για το σκοπό αυτό σημαντική βοήθεια έρχονται να προσφέρουν οι τεχνολογίες τηλεπικοινωνιών και πληροφορικής.

Η παρούσα εργασία ασχολείται με τη μελέτη, το σχεδιασμό και την υλοποίηση ενός Ηλεκτρονικό Ιατρικό Φάκελο Ασθενούς (Η.Ι.Φ.Α), ώστε να παρουσιαστούν οι επεκτάσεις των δυνατοτήτων και λειτουργιών του παραδοσιακού ιατρικού φακέλου.

Η εφαρμογή που υλοποιήθηκε είναι διαδικτυακή. Περιλαμβάνει ένα μεγάλο σύνολο από στοιχεία του ασθενούς σύμφωνα με την επισκεψιμότητά του και τα επεισόδια που έχουν δημιουργηθεί κατά την διάρκεια αυτών. Επιπλέον περιλαμβάνει πεδία για συγκέντρωση των δημογραφικών στοιχείων του ασθενούς καθώς και τα ιατρικά δεδομένα που τον αφορούν σύμφωνα με το ατομικό του ιστορικό. Τέλος παρέχει στο ιατρικό προσωπικό την δυνατότητα άντλησης πληροφοριών στατιστικής φύσεως.

Είναι κοινή διαπίστωση ότι ο όγκος των πληροφοριών που σχετίζονται με την φροντίδα και την ιατρική παρακολούθηση του ασθενούς έχει αυξηθεί τα τελευταία χρόνια, το οποίο δικαιολογείται αρκετά λόγω της ενσωμάτωσης αυξημένου αριθμού εργαστηριακών και παρακληνικών εξετάσεων στους φακέλους των ασθενών.

Λόγω της αυξανόμενης ανάγκης για την παροχή καλύτερων υπηρεσιών υγείας με μικρότερο κόστος, η παροχή υπηρεσιών υγείας δεν αφορά μόνο τα λογιστήρια και την κάλυψη διοικητικών αναγκών. Έτσι εισήχθη η έννοια του Ηλεκτρονικού Ιατρικού Φακέλου Ασθενούς για την τήρηση των ιατρικών στοιχείων. Η αποθήκευση και ανάκτηση των δεδομένων γίνεται γρήγορα και με ασφάλεια. Επιπλέον, καθίσταται δυνατή η επεξεργασία των δεδομένων και η άμεση μεταφορά τους με ηλεκτρονικά μέσα σε οποιαδήποτε απόσταση.

Με βάση τα παραπάνω η εφαρμογή που αναπτύχθηκε είναι φιλική προς το χρήστη και ικανοποιεί τις προϋποθέσεις ασφάλειας δεδομένων. Η σημαντικότητα της εφαρμογής έγκειται στο γεγονός ότι τα στοιχεία αποθηκεύονται με ασφάλεια σε μία βάση δεδομένων και είναι ανά πάσα στιγμή στην διάθεση του Ιατρού. Μπορεί γρήγορα να ανατρέξει στα ιστορικά επισκέψεων των ασθενών του, να κάνει προσθήκες αλλά και να εξάγει συμπεράσματα ως προς το σύνολο των καταχωρήσεών του. Επίσης δόθηκε μεγάλη έμφαση στη δημιουργία ενός εύχρηστου περιβάλλοντος διαχείρισης το οποίο μπορεί να χειριστεί ο χρήστης-γιατρός χωρίς να απαιτούνται γνώσεις πληροφορικής.

1.2 Αντικείμενο Διπλωματικής

Στα πλαίσια δημιουργίας της παρούσας Διπλωματικής Εργασίας, γίνεται προσπάθεια αξιοποίησης των γνώσεων που αποκτήθηκαν κατά την διάρκεια παρακολούθησης του Προγράμματος Μεταπτυχιακών Σπουδών. Επιχειρείται να γίνει ανάλυση, σχεδίαση και ανάπτυξη ενός συστήματος, το οποίο να δείχνει τη λειτουργία του Ηλεκτρονικού Ιατρικού Φακέλου Ασθενούς, να παρουσιαστεί η αλληλεπίδραση του χρήστη με την εφαρμογή και γενικότερα να προταθεί μία πλήρης εικόνα για τις καινοτομίες που μπορούν να υλοποιηθούν στα Πληροφοριακά Συστήματα Υγείας του σήμερα.

Επιπροσθέτως, πέρα από την υλοποίηση της εφαρμογής με την χρήση σύγχρονων τεχνολογιών, μεγάλη βαρύτητα δόθηκε στο καθορισμό του θεωρητικού υπόβαθρου αναφορικά με την ιατρική αλλά και την τεχνολογική πλευρά του θέματος για την αποθήκευση και επεξεργασία των πολύτιμων ιατρικών δεδομένων.
Μέχρι σήμερα δεν υπάρχει μία ξεκάθαρη άποψη για τα συστήματα Ηλεκτρονικών Ιατρικών Φακέλων Ασθενών (ΗΙΦΑ). Άλλοτε αποδίδεται ως αντίγραφο του χειρόγραφου φακέλου με διαδικασίες αυτόματης ανάγνωσης, άλλοτε ως αυτοματοποιημένος εργαστηριακός φάκελος και άλλοτε ως Ηλεκτρονικός Φάκελος Υγείας.
Ο ιατρικός φάκελος αποτελεί πολύτιμο εργαλείο που μπορεί να έχει στα χέρια του ο κάθε γιατρός και συνεπώς κατέχει μια σπουδαία θέση κατά τη διάρκεια της διάγνωσης για την θεραπεία του ασθενούς.
Ο τύπος του ΗΙΦΑ μπορεί να είναι είτε κλασσικός περιέχοντας απλά μόνο τη στοιχειώδη κλινική πληροφορία, είτε μοντέρνος και να περιέχει επιπλέον την κατανεμημένη στα επιμέρους υποσυστήματα, πληροφορία για τις ιατρικές απεικονίσεις, να παράγει μηνύματα και να διασυνδέεται με άλλες μονάδες υγείας.
Κύριο μέλημα της εργασίας ήταν να εντοπιστούν καταρχήν τα θεωρητικά για τον ιδανικό ΗΙΦΑ ο οποίος παρέχει τη δυνατότητα σχεδιασμού ιατρικών συμπερασμάτων από τα δεδομένα του, με τη χρήση αλγορίθμων εξόρυξης δεδομένων και με τη ταυτόχρονη μετάφραση όλων αυτών των κλινικών δεδομένων.
Συνεπώς, στο πλαίσιο εκπόνησης της εργασίας, υλοποιήθηκε σύστημα ΗΙΦΑ με τα δημογραφικά στοιχεία του ασθενή, πληροφορίες για την προέλευση, την πρόοδο και την κατάληξη μιας ασθένειας, τις διαγνώσεις συνοδευμένες από σαφείς λεπτομέρειες των εκάστοτε νοσηλειών, όπως συνταγογραφία και τιμές εργαστηριακών εξετάσεων. Η καταγραφή και η παρακολούθηση της πορείας του ασθενούς συνδυάζεται και με την δυνατότητα να δημιουργούνται συγγένειες πρώτου βαθμού μεταξύ των καταχωρημένων ασθενών.
Ο φάκελος λειτουργεί γενικά και ως ιστορικός φάκελος της κατάστασης της υγείας ενός συγκεκριμένου ατόμου, δίνοντας έτσι στο θεράποντα ιατρό κάθε φορά τη δυνατότητα να γνωρίζει στοιχεία από προηγούμενες ή παράλληλες θεραπευτικές αγωγές και να πράττει ανάλογα. Επιπρόσθετα, λαμβάνοντας υπόψιν το οικογενειακό ιστορικό μπορεί να εκτιμήσει αν υπάρχει κίνδυνος για εμφάνιση κληρονομικών ασθενειών, ώστε να γίνουν εγκαίρως προληπτικές εξετάσεις και ταχύτερες διαγνώσεις.

1.3 Συνοπτική Περιγραφή

Η δομή της εργασίας είναι σε ενότητες με σκοπό να παρουσιαστεί αρχικά με συνοπτικό τρόπο το θεωρητικό υπόβαθρο αναφορικά με την ιατρική αλλά και την τεχνολογική πλευρά του θέματος για την αποθήκευση και επεξεργασία ιατρικών δεδομένων και στη συνέχεια να γίνει υλοποίηση ενός συστήματος, το οποίο να δείχνει τη λειτουργία του Ηλεκτρονικού Ιατρικού Φακέλου Ασθενούς.
Αρχικά περιγράφεται ο ρόλος του Ιατρικού Φακέλου Ασθενούς, γίνεται μία σύντομη ιστορική αναδρομή και τονίζεται η σημασία εφαρμογής του σε ηλεκτρονική μορφή. Στην συνέχεια, γίνεται αναφορά στα πρότυπα που υπάρχουν για τον ηλεκτρονικό ιατρικό φάκελο και για τα Πληροφοριακά Συστήματα παροχής Υπηρεσιών Υγειάς που υπάρχουν στην αγορά. Επιπλέον, παρουσιάζονται και αναλύονται έρευνες πάνω σε ασθένειες που είναι κληρονομικές, ώστε λαμβάνοντας υπόψιν τα δεδομένα αυτά, να γίνει εκτίμηση για το αν κάποιος ασθενής θα νοσήσει

προβάλλοντας στατιστικά σε διαγράμματα. Τέλος σχεδιάζεται και υλοποιείται ένα σύστημα Ηλεκτρονικού Ιατρικού Φακέλου Ασθενούς στον τομέα της Ιατρικής Επιστήμης, καθορίζονται οι απαιτήσεις δεδομένων για τη σχεδίαση και δημιουργία μιας βάσης δεδομένων και παρουσιάζονται οι γλώσσες προγραμματισμού, οι τεχνολογίες καθώς και τα εργαλεία που χρησιμοποιούνται και εφαρμόζονται.

2 ΙΑΤΡΙΚΟΣ ΦΑΚΕΛΟΣ ΑΣΘΕΝΗ

2.1 Εισαγωγή

Είναι γεγονός ότι μέχρι σήμερα, το ιατρικό ιστορικό των ασθενών στα νοσοκομεία και ιατρεία της Ελλάδος καταγράφεται χειρόγραφα στις καρτέλες ασθενών και στις γνωματεύσεις των ιατρών. Αυτό έχει ως αποτέλεσμα να δημιουργηθεί μία ασυνέχεια στη παρακολούθηση του ιατρικού ιστορικού από τους γιατρούς διαφορετικών ειδικοτήτων στη διάρκεια ζωής ενός ατόμου, ενώ ταυτόχρονα το κράτος δεν μπορεί να έχει σαφή εικόνα για τους δείκτες υγείας του πληθυσμού του, προκειμένου να προβλέψει κόστη δαπανών υγείας.

Η συλλογή ηλεκτρονικών πληροφοριών που αφορούν την υγεία ενός συγκεκριμένου ατόμου θα μπορούσε να γίνεται συστηματικά, ώστε στη συνέχεια να μοιράζεται μέσω πληροφοριακών συστημάτων και δικτύων, είτε ακόμη να δημιουργηθεί ο ηλεκτρονικός Ιατρικός φάκελος του κάθε ασθενή από ένα οργανισμό, ο οποίος προσφέρει Ιατροφαρμακευτική φροντίδα, όπως είναι ένα νοσοκομείο ή το γραφείο ενός Ιατρού που συνήθως είναι μέρος ενός αυτόνομου συστήματος πληροφοριών Υγείας.

2.2 Ιατρική Πληροφορία

Γενικά κάθε πληροφορία ως οργανωμένο στοιχείο ή γνώση που παρέχει μία βάση για τη λήψη αποφάσεων, καθορίζεται κυρίως δια μέσου των λειτουργιών της. Στην περίπτωσή μας, όταν ένας γιατρός αποφασίζει για τον ασθενή, θα πρέπει αρχικά να μπορεί να μελετήσει το ιστορικό του, όπως ευρήματα από εργαστηριακές και απεικονιστικές εξετάσεις, υποθέσεις, διαγνώσεις και προηγούμενες νοσηλείες.

Οι γιατροί πρέπει να έχουν στην διάθεσή τους όλες τις απαραίτητες πληροφορίες που απαιτούνται για τη λήψη σωστών αποφάσεων, ώστε να αποφεύγονται τα ιατρικά σφάλματα, που έχουν όχι μόνο μοιραία αποτελέσματα για την υγεία των ασθενών αλλά συνεπάγονται και σημαντική οικονομική επιβάρυνση τόσο για τους ασθενείς όσο και για το σύστημα υγείας.

Η καλή διαχείριση της ιατρικής πληροφορίας μπορεί να βελτιώσει την ποιότητα, την αποτελεσματικότητα και την αποδοτικότητα των υπηρεσιών υγείας. Για να γίνει κάτι τέτοιο απαιτείται πρόσβαση των γιατρών στο κλινικό ιστορικό του ασθενούς με χρήση κοινών αρχείων, συμμετοχή στη λήψη αποφάσεων και μεγαλύτερη αυτοδιαχείριση.

Επιπλέον η διακίνηση της ιατρικής πληροφορίας είναι μία πολυπαραγοντική διαδικασία όχι μόνο γιατί σε αυτήν εμπίπτουν θέματα εμπιστευτικότητας των στοιχείων των ασθενών, αλλά και γιατί σχετίζεται με την έλλειψη κατάλληλων οργανωτικών υποδομών στα νοσοκομεία που να εξασφαλίζουν τη χρήση προτύπων και κατευθυντήριων οδηγιών κλινικής πρακτικής.

Σημαντικό επίσης πρόβλημα αποτελεί το ότι αρκετές φορές ο ασθενής είναι ο μοναδικός που γνωρίζει ποιοι ιατροί συμμετείχαν στην πρόοδο της θεραπείας του, αλλά και πού βρίσκονται σχετικές πληροφορίες για αυτή. Πολλές φορές δεν δίνει πληροφορίες από φόβο, ντροπή ή άγνοια. Συνεπώς ο ασθενής μόνο προσεγγιστικά μπορεί να θεωρηθεί ως πηγή πληροφόρησης.

Για τους ανωτέρω λόγους, υπάρχει η ανάγκη να δημιουργηθεί μία πληροφοριακή δομή που να παρέχει τη μεγαλύτερη δυνατή πληροφορία για τα παραπάνω, ενώ μεγαλύτερη αξιοπιστία στην αξιολόγηση των κλινικών δεδομένων επιτυγχάνεται με την συστηματική καταγραφή και ανάλυση της παραγόμενης ιατρικής πληροφορίας. Το σημαντικότερο στοιχείο που αποτελεί και τη βάση της διαχείρισης της ιατρικής πληροφορίας στο χώρο και στο χρόνο λοιπόν, είναι ο Ιατρικός Φάκελος.

2.3 Ιατρικός Φάκελος

Σύμφωνα με την Ευρωπαϊκή Επιτροπή Προτυποποίησης, Ιατρικός Φάκελος είναι η αποθήκη όλων των πληροφοριών που αφορούν στο ιατρικό ιστορικό του ασθενούς, έτσι ώστε να αποτελεί τη βάση της διάγνωσης και της θεραπευτικής αντιμετώπισης του ασθενούς αλλά και τη βάση επιδημιολογικών ερευνών. Επιπλέον παρέχει πληροφορίες διοικητικής, οικονομικής και στατιστικής φύσεως καθώς και ποιοτικού ελέγχου .

Συνεπώς, ο Ιατρικός Φάκελος είναι η συστηματοποιημένη συλλογή του ιστορικού και της κατάστασης υγείας ενός ασθενούς. Συνήθως δημιουργείται, διατηρείται και συντηρείται από έναν ιατρό ή μία Μονάδα Υγείας ή άλλον επαγγελματία φροντίδας υγείας.

Σύμφωνα με τη διεθνή βιβλιογραφία, οι ιατρικοί φάκελοι ταξινομούνται σε σχέση με τα ακόλουθα στοιχεία:

- Το περιεχόμενο: Φάκελος ενδονοσοκομειακών ασθενών, Φάκελος εξωνοσοκομειακών ασθενών, Φάκελος Φροντίδας Υγείας.
- Τη δομή: Φάκελος προσανατολισμένος στο πρόβλημα, Φάκελος προσανατολισμένος στο χρόνο, Φάκελος προσανατολισμένος στην εργασία, Φάκελος προσανατολισμένος στην αντιμετώπιση του ασθενή.
- Το σκοπό: Νοσηλευτικός φάκελος, Ακτινολογικός φάκελος, Φαρμακευτικός φάκελος.
- Το μέσο που χρησιμοποιείται για την καταγραφή: Χειρόγραφος φάκελος, Ηλεκτρονικός φάκελος, Φάκελος Πολυμέσων, Φάκελος ασθενή σε μικροφίλμμ.

Ανεξάρτητα από τη μορφή που έχει, κάθε ιατρικός φάκελος θα πρέπει να περιέχει όλα τα δεδομένα και τη πληροφορία που σχετίζεται με την κατάσταση υγείας του ασθενή. Η πληροφορία αυτή αφορά στο ιστορικό, στη κλινική εξέταση, στη διάγνωση, στα αποτελέσματα εργαστηριακών και διαγνωστικών ή παρακλινικών εξετάσεων, στις απεικονιστικές εξετάσεις κ.α..

Με άλλα λόγια ένας τέτοιος φάκελος χρειάζεται να συνδυάζει μία πλειάδα από διαφορετικού τύπου πληροφορίες. Αυτές οι πληροφορίες είναι:

- Δημογραφικά στοιχεία
- Ιατρικό ιστορικό
- Κλινικά δεδομένα φυσικής εξέτασης – διαγνώσεις και σημεία
- Νοσηλείες – Εγχειρήσεις
- Ιατροφαρμακευτική περίθαλψη
- Εργαστηριακές εξετάσεις (ανάλυση αίματος ,ούρων, κλπ.)
- Καταγραφές βιοδυναμικών (ηλεκτροκαρδιογράφημα, ηλεκτρομυογράφημα, κλπ.)
- Ιατρικές πράξεις
- Παραπεμπτικά - Γνωματεύσεις
- Διαγνωστικές εξετάσεις και ιατρικές εικόνες (Ακτινογραφίες, μαγνητικές τομογραφίες, αξονικές τομογραφίες, κλπ.)
- Διαχειριστικά – οικονομικά στοιχεία ιατρικών πράξεων και νοσηλειών
- Πιθανά αρχεία παλιών ιατρικών φακέλων

Οι αντίστοιχες εξετάσεις συνοδεύουν τον φάκελο του ασθενούς συνήθως υπό την μορφή με την οποία δημιουργούνται στα αντίστοιχα εργαστήρια.

Αποτέλεσμα των μορφών αυτών είναι η παραγωγή ενός μεγάλου όγκου ιατρικού φακέλου, με μεγάλη πιθανότητα απώλειας δεδομένων, με μεγάλη δυσκολία ανάκτησης πληροφορίας, με ασύγχρονο συσχετισμό του ιστορικού με τις εξετάσεις και την κλινική εξέταση.

2.4 Σκοποί Ιατρικού Φακέλου

Παραδοσιακά ο ιατρικός φάκελος εξυπηρετεί του παρακάτω σκοπούς:

- Αποτελεί ένα μέσο επικοινωνίας ανάμεσα στο ιατρικό και παραϊατρικό προσωπικό που ασχολείται με τον συγκεκριμένο ασθενή. Οδηγίες θεραπείας, διαγνώσεις, παραπεμπτικά με ειδικές οδηγίες, καταγραφή πορείας νόσου κ.λπ., δρομολογούνται στους διαφόρους εμπλεκόμενους, που δεν έχουν τη δυνατότητα της μεταξύ τους άμεσης επικοινωνίας, μέσω του ιατρικού φακέλου.
- Κατά την περίοδο αντιμετώπισης του προβλήματος, ο ιατρικός φάκελος αποτελεί το σημείο αναφοράς στο οποίο ανατρέχει κάποιος για να έχει μία εικόνα της κατάστασης του ασθενή. Οι εμπλεκόμενοι σε ένα ιατρικό επεισόδιο γνωρίζουν ότι για να δουν τα αποτελέσματα μμίας εξέτασης, πρέπει να ανατρέξουν στον ιατρικό φάκελο του ασθενή.
- Ανεπίσημα, ο ιατρικός φάκελος χρησιμεύει και ως «χώρος εργασίας» όπου καταγράφονται ιδέες και εντυπώσεις για το πρόβλημα του ασθενή και την πορεία της αντιμετώπισης του προβλήματος. Είναι ο χώρος όπου κάποιος μπορεί να πληροφορηθεί την εξέλιξη του περιστατικού ως μία αφήγηση τρίτων.
- Με την ολοκλήρωση ενός επεισοδίου, ο ιατρικός φάκελος είναι το μέρος που φυλάσσονται όλα τα κλινικά δεδομένα για μελλοντική χρήση, είτε αυτή αφορά περαιτέρω θεραπεία του ασθενή, είτε αφορά την έρευνα (κλινική έρευνα, επιδημιολογικές μελέτες, εκτίμηση της ποιότητας των προσφερόμενων υπηρεσιών και έρευνα αγοράς φαρμάκων).

Στα παραπάνω, θα προσθέταμε ότι ο ιατρικός φάκελος μπορεί να χρησιμεύσει για τον μετέπειτα έλεγχο των διαδικασιών που ακολουθήθηκαν κατά τη διάρκεια της θεραπείας του ασθενή, π.χ. στην περίπτωση υποψίας ιατρικού λάθους. Επίσης, μπορεί να χρησιμεύσει ως μέσο διασταύρωσης των οικονομικών στοιχείων που αφορούν το επεισόδιο. Για παράδειγμα, οι περισσότερες ασφαλιστικές εταιρείες, απαιτούν στοιχεία του φακέλου ώστε να αποφανθούν για την κάλυψη συγκεκριμένων εξετάσεων, ελέγχοντας την αναγκαιότητα πραγματοποίησης της εξέτασης.

Τέλος, δεν θα πρέπει να ξεχάσουμε να αναφερθούμε στις ανάγκες που προκύπτουν από τη σύγχρονη αντίληψη γύρω από τη διοίκηση και διαχείριση, οι οποίες απαιτούν τη χρήση όσο το δυνατόν περισσότερων πληροφοριών που θα λειτουργήσουν υποστηρικτικά σε αποφάσεις που αφορούν τη διαχείριση ενός οργανισμού παροχής ιατρικών υπηρεσιών (π.χ. νοσοκομείο), αλλά και ολόκληρου του συστήματος υγείας μίας περιοχής ή ενός κράτους.

2.5 Ιστορική Αναδρομή

Τον πέμπτο π.Χ. αιώνα, οι ιατρικές εκθέσεις επηρεάσθηκαν έντονα από τον Ιπποκράτη. Τα ιστορικά της εποχής περιείχαν γεγονότα που προηγούνταν της ασθένειας και όχι πραγματικά αιτιώδεις διασαφηνίσεις. Ο Ιπποκράτης περιέγραφε την πορεία μίας ασθένειας με καθαρά χρονολογική σειρά. Ένα τέτοιο ιστορικό ονομάζεται time-oriented medical record. Οι περιγραφές, αντικατοπτρίζουν κυρίως την ιστορία, όπως την αναφέρει ο ασθενής και οι συγγενείς του.

Μέχρι τις αρχές του 19ου αιώνα, οι γιατροί στήριζαν τις παρατηρήσεις τους σε ό,τι άκουγαν, έπιαναν, έβλεπαν. Το 1816, ο Laennec εφηύρε το στηθοσκόπιο. Αυτό το όργανο συνέβαλε σημαντικά στις διαθέσιμες διαγνωστικές τεχνικές. Όταν περισσότερα διαγνωστικά όργανα, όπως το οφθαλμοσκόπιο και το λαρυγγοσκόπιο, τέθηκαν στη διάθεση των γιατρών, αναπτύχθηκε μία ορολογία προκείμενου να εκφραστούν τα καινούρια ευρήματα αυτών των οργάνων. Η πρόοδος αυτής της νέας τεχνολογίας προκάλεσε την επέκταση του εύρους του ιστορικού του ασθενούς,

από την ιστορία που διηγείτο ο ίδιος ο ασθενής και οι συγγενείς του στα ευρήματα του γιατρού και του νοσοκόμου.
Λίγο μετά το 1880, ο Αμερικανός χειρουργός William Mayo σχημάτισε την πρώτη πολυκλινική, που έγινε η γνωστή τώρα Κλινική Mayo, στο Ρότσεστερ της Μινεσότα. Στην αρχή, εκεί, κάθε γιατρός κρατούσε ιατρικές σημειώσεις σε ένα προσωπικό βιβλίο χρεών και υποχρεώσεων. Το βιβλίο περιείχε χρονολογική έκθεση όλων των συναντήσεων με ασθενείς. Κατά συνέπεια, οι σημειώσεις που σχετίζονταν με έναν μόνο ασθενή μπορεί να απείχαν σελίδες, ανάλογα με την χρονική απόσταση των επισκέψεων. Έτσι, οι σκόρπιες σημειώσεις έκαναν πολύπλοκη την καλή, συνολική θεώρηση όλης της ιστορίας της νόσου ενός ασθενή. Επιπρόσθετα, μέρος των πληροφοριών σχετικά με τον ασθενή μπορεί να βρίσκονταν στα βιβλία άλλων γιατρών.
Το 1907, η κλινική Mayo υιοθέτησε έναν ξεχωριστό φάκελο για κάθε ασθενή. Από αυτήν την καινοτομία προέκυψε το ασθενοκεντρικό ιατρικό ιστορικό. Όμως, το γεγονός ότι όλες οι σημειώσεις φυλάσσονταν σε έναν μόνο φάκελο, δεν σήμαινε ότι τηρούνταν και τα κριτήρια στα οποία έπρεπε να ανταποκρίνεται το περιεχόμενο αυτών των ιστορικών.
Το 1920, η διοίκηση της κλινικής Mayo συμφώνησε μία ελάχιστη ποσότητα δεδομένων που όλοι οι γιατροί ήταν υποχρεωμένοι να καταγράφουν. Αυτό το σύνολο των δεδομένων αποτελεί λίγο-πολύ τον σκελετό για το σημερινό ιατρικό ιστορικό.
Παρά την πρωτοβουλία αυτή για την (πρώτο)τυποποίηση του ιστορικού των ασθενών, τα γραπτά τους περιεχόμενα ήταν συχνά μίγμα παραπόνων, αποτελεσμάτων εξετάσεων, σκέψεων, θεραπευτικών πλάνων και ευρημάτων. Αυτές οι άναρχες σημειώσεις δεν πρόσφεραν βαθιά γνώση, ιδιαίτερα στις περιπτώσεις ασθενών που εξετάζονταν για περισσότερες από μία ενοχλήσεις ή ασθένειες. Ο Weed άδραξε την ευκαιρία να βελτιώσει την οργάνωση του ιστορικού των ασθενών και στη δεκαετία του 1960 εισήγαγε το problem-oriented medical record. Σύμφωνα με αυτό, σε κάθε ασθενή αποδίδονταν ένα ή περισσότερα προβλήματα. Παρότι το problem-oriented ιστορικό του Weed έγινε εύκολα αποδεκτό σε επίπεδο ορθολογιστικό, αποδείχτηκε ότι στην πράξη απαιτούσε μεγάλη πειθαρχία προκειμένου να επιμείνει κανείς στη μέθοδο.
Η ιδέα του Ηλεκτρονικού φακέλου ξεκίνησε το 1969 από τον Dr. William Edward Hammond II ως το μέρος όπου αποθηκεύονται για πάντα όλες οι πληροφορίες για έναν ασθενή, προσφέροντας του έτσι τις καλύτερες υπηρεσίες, παρέχοντας δηλαδή τη δυνατότητα της γνώσης κάθε λεπτομέρειας του ιστορικού του ασθενή (εξετάσεις, διαγνώσεις, φάρμακα κτλ.) και συνεπώς τη συνολική αντίληψη των προβλημάτων υγείας. Το μέρος αυτό είναι οι ηλεκτρονικοί υπολογιστές αντί των χάρτινων χειρόγραφων φακέλων, μέσω των οποίων επιτυγχάνεται η συλλογή και η χρονική παρουσίαση των δεδομένων της κατάστασης υγείας των ασθενών ανά πάσα χρονική στιγμή.
Η υλοποίηση του Ιατρικού φακέλου πραγματοποιήθηκε με την κατασκευή μίας διασύνδεσης ανάμεσα σε ένα σκάνερ και έναν προσωπικό υπολογιστή (τύπου PDP 12), με ένα πρόγραμμα σε γλώσσα assembly που εκτύπωνε το ιατρικό ιστορικό άμεσα από τον ασθενή στο Health Department at Duke University. Από το 1973 το κλείσιμο ραντεβού και οι πληρωμές των εξωτερικών ασθενών λειτουργούσαν βάσει του πρώτου Ηλεκτρονικού Ιατρικού φακέλου (CPR). Αργότερα ομάδα από πέντε γιατρούς και φοιτητές κατασκεύασε το GEMISCH, δηλαδή μία command line γλώσσα που έτρεχε στα λειτουργικά συστήματα εκείνης της εποχής (RSX and VMS Operating Systems), βάσει του οποίου ειδικές εφαρμογές αντικαταστάθηκαν από γενικότερες εφαρμογές.

2.6 Σήμερα

Η μηχανογράφηση δεδομένων ιατρικής δεν ξεκίνησε από την ανάγκη αντιμετώπισης των προβλημάτων του paper-based ιατρικού φακέλου. Όπως σε όλες σχεδόν τις περιπτώσεις μηχανογράφησης, οι πρώτες διαδικασίες που μηχανογραφήθηκαν ήταν αυτές που είχαν να κάνουν με τα οικονομικά δεδομένα και την διαχείριση πόρων ενός συστήματος. Έτσι λοιπόν και στην περίπτωση του ιατρικού χώρου, οι πρώτες προσεγγίσεις είχαν να κάνουν με τη μηχανογράφηση των λογιστηρίων των οργανισμών υγείας και κατόπιν με τη διαχείριση πόρων (κλινών, ραντεβού, κ.λπ.).

Σε αντίθεση, τομείς όπως ο ιατρικός φάκελος, εμφανίζουν αρκετές δυσκολίες στη μηχανογράφησή τους:

1. Δεν υπάρχουν καλά ορισμένες διαδικασίες. Για παράδειγμα, δεν υπάρχει πρωτόκολλο περιγραφής των στοιχείων που πρέπει να καταγράφονται κατά την εξέταση ενός ασθενούς, μία και αυτό εξαρτάται πολύ από την πείρα του ιατρού, το ιατρικό πρόβλημα του ασθενή, την ειδικότητα του ιατρού, κ.λπ.
2. Δεν υπήρχε άμεσο όφελος για τους οργανισμούς, εκτός ίσως από την υποψία ότι θα βελτιωνόταν η ποιότητα των προσφερόμενων υπηρεσιών από τον οργανισμό, και αυτό, μάλιστα, σε μακροπρόθεσμο ορίζοντα.
3. Η εφαρμογή μηχανογράφησης δεν θα μπορούσε να εξαλείψει (ούτε καν να μειώσει αισθητά) την πιθανότητα ανθρώπινου λάθους, π.χ. στην περίπτωση μίας διάγνωσης.
4. Συνήθως οι απόπειρες μηχανογράφησης προς αυτή την κατεύθυνση, απαιτούσαν την καταγραφή μεγάλου όγκου πληροφοριών από το ιατρικό προσωπικό με αποτέλεσμα τη δημιουργία καθυστέρησης στην ολοκλήρωση των καθηκόντων τους και, κατά συνέπεια, την «εκ των έσω» απόρριψη τέτοιων συστημάτων.
5. Το ιατρικό προσωπικό συνήθως αντιμετώπιζε με δέος και φόβο την τεχνολογία.
6. Τέλος, δεν μπορούμε να αγνοήσουμε το γεγονός ότι οι πρώιμες προσπάθειες μηχανογράφησης είχαν «φτωχό» user-interface, κάτι που δυσχέραινε ιδιαίτερα τη χρησιμοποίησή της από ανθρώπους που αντιμετώπιζαν έτσι κι αλλιώς αυτή τους την υποχρέωση ως πάρεργο.

Τα παραπάνω στοιχεία οδήγησαν στη σημερινή πραγματικότητα όπου πολύ λίγοι φορείς υγείας μπορούν να υποστηρίξουν ότι διαθέτουν ηλεκτρονικό φάκελο ασθενή. Σχεδόν σε όλους, υπάρχουν πληροφοριακά συστήματα, τα οποία όμως σε πολύ μικρό βαθμό αφορούν κλινικά στοιχεία για τους ασθενείς. Μπορούμε να πούμε ότι είναι διαχειριστικό-κεντρικά πληροφοριακά συστήματα.

Ο ηλεκτρονικός φάκελος ασθενή μπορεί να είναι «κλασσικός», περιέχοντας στοιχειώδη κλινική πληροφορία και «μοντέρνος» περιέχοντας επιπλέον κατανεμημένη πληροφορία για ιατρικές απεικονίσεις, ηχογραφήσεις, video, παραγωγή μηνυμάτων και με τη δυνατότητα διασύνδεσης με άλλες μονάδες υγείας. Ένας ιδανικός φάκελος παρέχει τη δυνατότητα σχεδιασμού ιατρικών συμπερασμάτων από τα δεδομένα του, με τη χρήση αλγόριθμων εξόρυξης δεδομένων μεταφράζοντας δεδομένα με επεξεργασία της φυσικής γλώσσας ενός κειμένου, αποτελώντας τον Ηλεκτρονικό Φάκελο Πολυμέσων.

Περιφερειακά σε αυτή την προσέγγιση, υπάρχουν τμήματα ιατρικής πληροφορίας που καταγράφονται ηλεκτρονικά ή, ακόμα συχνότερα, παράγονται με τη βοήθεια μηχανογραφικών λύσεων, όπως για παράδειγμα τα εργαστήρια ενός νοσοκομείου. Σπάνια όμως τέτοιου είδους πληροφορίες συνθέτουν μία συνολική εικόνα για τον ασθενή, μία και αποτελούν ένα μικρό συνήθως μέρος της. Είναι χαρακτηριστικό ότι τις περισσότερες φορές η πληροφορία είναι

διασκορπισμένη όχι μόνο σε διαφορετικούς οργανισμούς υγείας, αλλά ακόμα και στον ίδιο τον οργανισμό που συνήθως χρησιμοποιεί διαφορετικά πληροφοριακά συστήματα.
Πάντως είτε ο φάκελος είναι κλασσικός είτε πολυμεσικός, θα πρέπει να επιτρέπει σε κάθε χρονική στιγμή την ανάκτηση των δεδομένων που αφορούν τον ασθενή είτε ανά μονάδα υγείας, είτε ανά υγειονομική περιφέρεια, είτε σε Εθνικό δίκτυο.
Είναι γεγονός ότι δεν υπήρξε εκτενής υλοποίηση αυτών των συστημάτων. Βασικοί λόγοι ήταν το κόστος για μικρούς οργανισμούς, καθώς επίσης και η δυσκολία στη χρήση τέτοιων ψηφιακών μέσων από τους ιατρούς και τους ασθενείς. Επιπλέον, δεν υπήρχε ένα σταθερό πρότυπο, το οποίο να επιτρέπει την μεταφορά των δεδομένων από τον ένα οργανισμό στον άλλον, κάτι που δημιούργησε αρνητικό κλίμα όσον αφορά την υιοθέτηση τέτοιων πρακτικών συντήρησης Ιατρικού ιστορικού.
Στις Η.Π.Α. έως πρόσφατα οι προσπάθειες επικεντρώνονταν στην ενοποίηση των κέντρων που υποστηρίζουν ΗΙΦΑ, μέσω προτυποποίησης και με την εδραίωση ενός κεντρικού μη κερδοσκοπικού οργανισμού, του «Healthcare Information and Management Systems Society (HIMSS)», αλλά δίχως τα επιθυμητά αποτελέσματα. Το 2009 με το «Health Information Technology for Economic and Clinical Health Act», αλλιώς «HITECH Act», στα πλαίσια του «American Recovery and Reinvestment Act of 2009», παρέχονται κίνητρα για την υιοθέτηση ενός πανεθνικού δικτύου για την παροχή Ηλεκτρονικού Ιατρικού Φακέλου και εξασφαλίζει την κινητικότητα των αρχείων σε όλες τις πολιτείες των Η.Π.Α. Την αντίστοιχη πορεία έχει ακολουθήσει και το Ηνωμένο Βασίλειο.
Στην Ευρωπαϊκή Ένωση, έως τώρα, καμία προσπάθεια δεν είχε γίνει· εκτός της Ευρωπαϊκής κάρτας Ασφάλισης Ασθενείας, η οποία δεν πλησιάζει τον ορισμό του Ηλεκτρονικού Μητρώου Υγείας. Η πρώτη ενοποιημένη και οργανωμένη προσπάθεια επήλθε με τη «Ψηφιακή Ατζέντα για την Ευρώπη 2011 - 2020» και συγκεκριμένα την οδηγία 2011/24/EU της Commission, όπου ορίζονται τα δικαιώματα των πολιτών για πανευρωπαϊκή, διασυνοριακή παροχή υπηρεσιών υγείας, κάτι που πρέπει να έχει υλοποιηθεί έως το 2013. Το άρθρο 14 ορίζει ένα δίκτυο «eHealth»(επίσημη ονομασία), το οποίο αποτελείται από τους οργανισμούς κάθε κράτους υπεύθυνα για την υλοποίηση της οδηγίας. Αυτό κατ' επέκταση οδήγησε στην ανάγκη δημιουργίας ενός συστήματος Ηλεκτρονικού Ιατρικού Μητρώου, κάτι που εντάχθηκε στην Ψηφιακή Ατζέντα. Μια αναφορά χρήζει και στο European Institute for Health Records, μία Μ.Κ.Ο. με σκοπό την υιοθέτηση υψηλής ποιότητας Ηλεκτρονικού Μητρώου Υγείας στην Ευρωπαϊκή Ένωση.

2.7 Ο Ηλεκτρονικός Ιατρικός Φάκελος στην Ελλάδα

Στην Ελλάδα άργησε κατά πολύ η εφαρμογή της Πληροφορικής στο δημόσιο τομέα ενώ στα δημόσια νοσοκομεία των μεγάλων αστικών κέντρων ξεκίνησε περίπου κατά το τέλος της δεκαετίας του 1980, με την χρήση προσωπικών υπολογιστών σε κάποια τμήματα κυρίως οικονομικά. Παράλληλα το τμήμα Πληροφορικής δεν είχε θεσμοθετηθεί στους περισσότερους οργανισμούς των νοσοκομείων, ενώ σε όποια υπήρχε, είχε ελάχιστο εξειδικευμένο προσωπικό.
Οι ιατρικοί φάκελοι στα περισσότερα Δημόσια νοσοκομεία ακόμα και σήμερα εξακολουθούν να είναι χειρόγραφοι, ογκώδεις, ασαφείς, δυσεύρετοι, δυσανάγνωστοι ενώ πολλές φορές χάνονται, φθείρονται και αλλοιώνονται. Η αναζήτηση ιστορικών και κλινικών δεδομένων είναι πολύ δύσκολη, ενώ η εξαγωγή στατιστικών συμπερασμάτων εντελώς αβέβαιη και πολύπλοκη. Ακόμα και στις ελάχιστες περιπτώσεις που υπάρχει ατομικός ηλεκτρονικός φάκελος, τα περιεχόμενα και τα δεδομένα δεν μπορούν να κυκλοφορήσουν με το εσωτερικό δίκτυο του ιδίου νοσοκομείου, με

κυριότερη αιτία το ότι ο ηλεκτρονικός φάκελος και το πληροφοριακό διαχειριστικό σύστημα δεν έχουν ούτε την κατάλληλη διασύνδεση ούτε την απαραίτητη διαλειτουργικότητα.

2.8 Paper-based

Δεν υπάρχει αμφιβολία ότι για δεκαετίες, η λύση του paper-based ιατρικού φακέλου έχει χρησιμοποιηθεί με σχετική επιτυχία. Η μέθοδος που χρησιμοποιεί ως βάση το χαρτί, υλικό με το οποίο είναι εξοικειωμένοι οι περισσότεροι άνθρωποι και σίγουρα οι ασχολούμενοι με τα ιατρικά πράγματα, παρουσιάζει ως βασικό πλεονέκτημα αφενός την πιο πάνω εξοικείωση και αφετέρου το αυτόνομο της μεθόδου (π.χ. δεν είναι απαραίτητη η ύπαρξη υπολογιστή, ρεύματος, πρίζας, για να ανακτήσει κανείς το περιεχόμενο ενός φακέλου).

Το χαρτί, ωστόσο, από μόνο του ως υλικό παρουσιάζει κάποια σημαντικότατα μειονεκτήματα:

- Μπορεί να καταστραφεί εύκολα, ενώ είναι αρκετά επίπονη και χρονοβόρα η διαδικασία της δημιουργίας αντιγράφων ασφαλείας.
- Φθείρεται με τη χρήση ή το χρόνο, οπότε έχει περιορισμένη διάρκεια ζωής.
- Είναι διαθέσιμο σε ένα μόνο μέρος την ίδια στιγμή.

Παρόλα αυτά, τα σημαντικότερα προβλήματα που εμφανίζονται από την χρήση ενός paper-based ιατρικού φακέλου δεν οφείλονται στο βασικό χρησιμοποιούμενο υλικό, αλλά στο τι αυτό συνεπάγεται. Οι ιατρικοί φάκελοι απαιτούν ιδιαίτερα πολύ χρόνο για να βρεθούν, κυρίως σε μεγάλους οργανισμούς (νοσοκομεία κ.λπ.), ενώ συνήθως το περιεχόμενό τους είναι διασκορπισμένο σε διαφορετικά σημεία (γραφεία ιατρών, διαγνωστικά και εξεταστικά κέντρα κ.λπ.).

Ένα άλλο σημαντικό μειονέκτημα είναι ότι η πληθώρα ιατρικών φακέλων, τόσο σε ένα ιατρείο όσο και σε έναν οργανισμό, λειτουργεί εις βάρος του σε ό,τι αφορά το κόστος, καθώς απαιτείται μεγάλος αριθμός σε ανθρώπινο δυναμικό για τη σωστή αποθήκευση και ταξινόμηση χιλιάδων πιθανά φακέλων. Αυτός είναι και ο σημαντικότερος λόγος για τον οποίο πάρα πολλά νοσηλευτικά ιδρύματα και οργανισμοί διατηρούν αρχείο για ένα μικρό σχετικά χρονικό διάστημα (σε πολλά νοσοκομεία τα αρχεία φυλάσσονται για περίπου μία πενταετία).

Είναι σημαντικό να τονίσουμε ότι στην περίπτωση του paper-based ιατρικού φακέλου δεν είναι δυνατό να μιλάμε για την άμεση χρησιμοποίηση του περιεχομένου πληθώρας ιατρικών φακέλων για έρευνα, μιας και κάτι τέτοιο απαιτεί όχι μόνο ιδιαίτερη προσπάθεια, αλλά κυρίως χρόνο.

Τέλος, θα πρέπει να αναφερθούμε στις ανάγκες που προκύπτουν από τη σύγχρονη αντίληψη γύρω από τη διοίκηση και τη διαχείριση, οι οποίες απαιτούν τη χρήση όσο το δυνατόν περισσότερων πληροφοριών που θα λειτουργήσουν υποστηρικτικά σε αποφάσεις που αφορούν στη διαχείριση ενός οργανισμού παροχής ιατρικών υπηρεσιών (νοσοκομειακό ίδρυμα κ.λπ.), αλλά και ολόκληρου του συστήματος υγείας μμιας περιοχής ή ακόμα και ενός κράτους.

Συνοπτικά ο κλασικός μη ηλεκτρονικός φάκελος:

- Μεγάλη δυσκολία στη λήψη στατιστικών δεδομένων και ανάλυσης δεδομένων.
- Δυσκολία ανεύρεσης πληροφορίας από τους φακέλους
- Μεγάλο κόστος για τη συντήρηση του-απαιτείται προσωπικό για την οργάνωση και αρχειοθέτηση του
- Διασκορπισμένα στοιχεία του ιστορικού σε διάφορα ιατρικά κέντρα

2.9 Η έννοια του Ηλεκτρονικού Ιατρικού Φακέλου Ασθενούς

Ο Ηλεκτρονικός Ιατρικός Φάκελος είναι η αποθήκη όλων των πληροφοριών που αφορούν στο ιατρικό ιστορικό του ασθενούς. Αποτελεί επομένως την βάση της διάγνωσης και της θεραπευτικής αντιμετώπισης του ασθενούς αλλά και τη βάση επιδημιολογικών ερευνών. Επιπλέον, παρέχει πληροφορίες διοικητικής, οικονομικής και στατιστικής φύσεως, καθώς και ποιοτικού ελέγχου.

Ο θεράπων ιατρός, με το πάτημα ενός κουμπιού, θα γνωρίζει αναλυτικά όλα όσα πρέπει να ξέρει για τον ασθενή του, όπου κι αν βρίσκεται αυτός, καθώς θα είναι σε θέση να βλέπει συγκεντρωτικά όλο το ιατρικό ιστορικό του, τη σωστή πληροφορία τη στιγμή που τη χρειάζεται, ώστε να μπορεί εύκολα να εποπτεύσει το περιστατικό.

Ο ηλεκτρονικός φάκελος υγείας αποθηκεύει ψηφιακά ένα υποσύνολο δεδομένων ή όλα τα δεδομένα σχετικά με τις ιατρικές πράξεις που έγιναν κατά τη διάρκεια της ζωής ενός ατόμου με σκοπό την υποστήριξη της ποιοτικής, προσβάσιμης και αποτελεσματικής συνέχειας στην παροχή υπηρεσιών υγείας. Ένα σύστημα ηλεκτρονικού φάκελου υγείας είναι το πληροφοριακό σύστημα που διαχειρίζεται και δουλεύει σένα σετ από ηλεκτρονικούς φακέλους υγείας και βοηθά στην πρόσβαση και στη κοινή χρησιμοποίηση της υπάρχουσας πληροφορίας με φιλικό τρόπο. Για να μπορεί να χρησιμοποιηθεί σήμερα, θα πρέπει να είναι ασφαλές σύστημα και να εξασφαλίζει την εμπιστευτική υφή των δεδομένων.

Μερικές από τις ανάγκες των ιατρών που πρέπει να ικανοποιεί ένας ηλεκτρονικός ιατρικός φάκελος είναι:

- Εύκολη διαχείριση φακέλων ασθενών
- Ευχρηστία αποθήκευσης, ενημέρωσης και πλοήγησης στις ιατρικές πληροφορίες
- Παρουσίαση δεδομένων με απλό και κατανοητό τρόπο
- Αναζήτηση πληροφοριών γρήγορα και εύκολα
- Διευκόλυνση στην διεξαγωγή μελετών και ερευνών
- Ανάλυση δεδομένων και παραγωγή στατιστικών αποτελεσμάτων
- Επεκτασιμότητα
- Ευρεία χρήση προτύπων

Ο ηλεκτρονικός φάκελος ασθενή αποτελεί ένα σύνολο αρχείων με δομημένη μορφή και μπορεί να περιέχει πολυμεσικά δεδομένα, όπως ακτινογραφίες ή ηλεκτροκαρδιογραφήματα, καθώς και δεδομένα με τη μορφή κειμένου, όπως διαγνώσεις, και μπορεί να είναι κατανεμημένος σε πολλά πληροφοριακά συστήματα υγείας ή συγκεντρωτικός. Οι ηλεκτρονικοί φάκελοι ασθενών περιέχουν τα δημογραφικά στοιχεία των ασθενών (όνομα, επώνυμο, φύλο…) και κάθε πληροφορία που αφορά στην υγεία των τελευταίων.

Τα δεδομένα αυτά θα πρέπει να μπορούν να επεξεργαστούν αυτοματοποιημένα αλλά και από τον άνθρωπο ενώ θα πρέπει να είναι σαφή και να περιέχουν κάθε πληροφορία που σχετίζεται με ένα ή περισσότερα γεγονότα που αφορούν στην υγεία του ασθενούς. Τα περιεχόμενα των ηλεκτρονικών φακέλων ασθενών μπορούν να εμπλουτίζονται π.χ. με διαγράμματα που περιγράφουν την αύξηση του σωματικού βάρους ή της αρτηριακής πίεσης σε βάθος χρόνου ή στατιστικά στοιχεία για τη συχνότητα των επεισοδίων άσθματος ενός ασθενή.

Το κράτος από την άλλη μεριά θα μπορεί να εξάγει στατιστικά στοιχεία για την υγεία του πληθυσμού και να εφαρμόσει πολιτικές πρόληψης. Με αυτόν τον τρόπο η πρωτοβάθμια φροντίδα υγείας μπορεί να δρα προληπτικά, αλλά και στοχευμένα, βελτιώνοντας τη δημόσια υγεία και μειώνοντας τη μελλοντική ιατρική/φαρμακευτική δαπάνη που θα απαιτηθεί αν δεν ληφθούν εγκαίρως μέτρα.

Τέλος, τα ιατρικά ευρήματα καταγράφονται ηλεκτρονικά και η διάγνωση του γιατρού τεκμηριώνεται. Η ιατρική ευθύνη συγκεκριμενοποιείται σε περιπτώσεις που προκύπτουν σημαντικά ιατρικά σφάλματα. Με αυτό τον τρόπο, ο γιατρός είναι υπόλογος έναντι του ασθενούς του και των ελεγκτικών μηχανισμών. Επιπρόσθετα, το ιατρικό ιστορικό δεν είναι δυνατόν να αλλοιωθεί ή να καταστραφεί.

3 ΙΣΤΟΡΙΚΟ ΤΟΥ ΑΡΡΩΣΤΟΥ

3.1 Εισαγωγή

Η αφήγηση των ενοχλημάτων του αρρώστου, καθώς επίσης και η αναφορά των τυχόν προβλημάτων υγείας, τα οποία απασχόλησαν στο παρελθόν τον ίδιο τον άρρωστο ή το άμεσο συγγενικό του περιβάλλον, αποτελούν το ιστορικό του αρρώστου.

Το ιστορικό αποτελεί, για όλα σχεδόν τα νοσήματα, τον ακρογωνιαίο λίθο της διάγνωσης. Πολλά μάλιστα νοσήματα, στα οποία δεν υπάρχουν ευρήματα από την φυσική εξέταση, διαγιγνώσκονται μόνο από το ιστορικό. Ένα πλήρες ιστορικό εγγυάται για μία σωστή διάγνωση και μεγάλο ποσοστό διαγνώσεων είναι σχεδόν βέβαιες και μόνο με τη λήψη ιστορικού.

Η λήψη καλού ιστορικού περιέχει την τεχνική, την ιατρική γνώση, την εμπειρία, την γνώση της ψυχολογίας του αρρώστου και τη θεώρησή του ως ένα ψυχοσωματικό σύνολο.

Η λήψη του ιστορικού του ασθενούς δεν είναι εύκολη διαδικασία. Ο ιατρός είναι πιθανό να αντιμετωπίσει ορισμένες δυσκολίες, συχνότερες των οποίων είναι:

1. Η απροθυμία του ασθενούς να αποκαλύψει κάποιες πληροφορίες. Παραδείγματα αποτελούν στοιχεία της σεξουαλικής συμπεριφοράς ή της καπνιστικής συνήθειας σε άτομα που είναι πιθανό να πάσχουν από ένα σεξουαλικώς μεταδιδόμενο νόσημα ή καρκίνο του πνεύμονα, αντίστοιχα.
2. Διαφορετική εκτίμηση μεταξύ ασθενούς και ιατρού ως προς τη σημασία των λέξεων.
3. Μη ακριβής προσδιορισμός του χρόνου έναρξης της νόσου.
4. Υπερτίμηση ή υποτίμηση των ενοχλημάτων από την πλευρά του ασθενούς.
5. Συνεχείς μεταβολές του ιστορικού εξαιτίας των όσων αναφέρονται από τον ασθενή.
6. Τα συμπτώματα αποτελούν νευρωσικές εκδηλώσεις: ενοχλήματα ασυνήθιστης έντασης, μη υποχώρηση άλγους παρά τη χορήγηση φαρμάκων, μη ικανοποίηση από όλους τους προηγούμενους ιατρούς, ιδιαίτερα εριστικός ασθενής.

3.2 Λήψη Ιατρικού Ιστορικού

Ο θεράπων ιατρός συγκεντρώνει στον μεγαλύτερο δυνατό βαθμό, όλα τα απαιτούμενα στοιχεία για την υγεία του ασθενούς του, την φαρμακευτική του αγωγή, τυχόν χειρουργικές επεμβάσεις που στις οποίες έχει υποβληθεί και περιγράφει την παρούσα κλινική εικόνα. Δεδομένα του νοσολογικού ιστορικού του ιδίου και της οικογενείας του είναι απαραίτητα προκειμένου να αποσαφηνίζεται τυχόν ύπαρξη κληρονομικής νόσου.

Το σύνολο των δεδομένων είναι απόρρητα και φυλάσσονται στην καρτέλα του πελάτη/ασθενούς.

Τα στοιχεία που λαμβάνονται με το Ιατρικό ιστορικό είναι καθοριστικά για τον καθορισμό της ιατρικής παρέμβασης καθώς και οι προτεραιότητες που τίθενται στην αντιμετώπιση πολλαπλών προβλημάτων υγείας.

Το ιατρικό ιστορικό είναι η συλλογή των στοιχείων που λαμβάνεται από τη συνέντευξη ενός ασθενή. Ακούγεται αρκετά απλό, όμως η λήψη ενός ιατρικού ιστορικού είναι πραγματικά μια τέχνη. Απαιτεί προσπάθειες από ομάδες τεχνικών και παθολόγων. Ο γιατρός που παίρνει το ιστορικό του ασθενή πρέπει να είναι συνομιλητικός, συμπονετικός και δημιουργικός. Οι τύποι ερωτήσεων που υποβάλλονται είναι αρκετά προσωπικοί. Πολλοί ασθενείς περιγράφουν σύντομα το πρόβλημά τους, πιστεύοντας ότι ο γιατρός αντιλήφθηκε το λόγο επίσκεψής τους. Τότε είναι που επιβάλλεται να χρησιμοποιήσει ο γιατρός όλη του την τέχνη προκειμένου να του αποσπάσει όσο το δυνατόν περισσότερες έγκυρες πληροφορίες. Ορισμένες απαντήσεις μπορούν να οδηγήσουν σε συνέπειες ικανές να αλλάξουν τη ζωή κάποιου (π.χ. χειρουργική επέμβαση).

Η τεχνική της λήψης ιστορικού περιλαμβάνει τρία κύρια στάδια:

- Την προσέγγιση του αρρώστου
- Την αφήγηση της ιστορίας της νόσου από τον ασθενή
- Την υποβολή ερωτήσεων στον άρρωστο για περισσότερες πληροφορίες

Ένας καλός τρόπος να πάρουμε πληροφορίες από τον ασθενή είναι να αναπτύξουμε μια συζήτηση και να συγκεντρώσουμε τις απαραίτητες πληροφορίες για ένα ολοκληρωμένο ιατρικό ιστορικό. Τέλος, η κλινική κρίση θα καθορίσει την ποσότητα πληροφοριών που απαιτείται. Η επανεξέταση αυτού του ιστορικού μπορεί να αποκαλύψει προβληματικές πληροφορίες.

3.3 Προσέγγιση Αρρώστου

Η πρώτη προσέγγιση στον άρρωστο έχει βασική σημασία για την συνέχεια και την έκβαση της λήψης ιστορικού. Ο γιατρός πρέπει να αποκτήσει την εμπιστοσύνη του ατόμου που εξετάζει. Η έκφραση του προσώπου του πρέπει να χαρακτηρίζεται από φιλικότητα, συμπάθεια, προσοχή, σοβαρότητα και ενδιαφέρον για τον άρρωστο. Ο γιατρός πρέπει να βλέπει τον άρρωστο στα μάτια, με έκφραση συμπάθειας και προσοχής. Πρέπει να αποφεύγει εκφράσεις, μορφασμούς και επιφωνήματα.

3.4 Τυποποίηση Πληροφοριών Ιστορικού Ασθενούς

Οι πληροφορίες που αναζητούνται από την λήψη του ιστορικού σε συνδυασμό με την φυσική εξέταση που θα ακολουθήσει και με τα αποτελέσματα των παρακλινικών εξετάσεων θέτουν την ορθή διάγνωση.

Υπάρχει μία τυποποίηση για την σειρά που πρέπει να αναζητηθούν οι πληροφορίες του ιστορικού. Στην πραγματικότητα, ενώ φαινομενικά η συζήτηση είναι ελεύθερη, ουσιαστικά κατευθύνεται από τον γιατρό.

3.4.1 Τα στοιχεία ταυτότητας

Τα στοιχεία ταυτότητας του αρρώστου, τα οποία απαραίτητα πρέπει να γνωρίζει ο γιατρός, είναι: το ονοματεπώνυμο, η ηλικία, ο τόπος γεννήσεως, το επάγγελμα και η διεύθυνση του αρρώστου.

3.4.2 Φαρμακευτική αγωγή

Γίνεται καταγραφή όλων των φαρμάκων που λαμβάνει ο ασθενής. Σημειώνεται η ονομασία, δοσολογία, έναρξη χορήγησης, αιτιολογία χορήγησης. Επίσης αναφέρεται το κατά πόσο υπάρχει συμμόρφωση με την αγωγή.

Αν δεν είστε βέβαιοι για την ονομασία κάποιου φαρμάκου ζητήστε από τον ασθενή το βιβλιάριο υγείας ή τα κουτιά από τα φάρμακα. Καταγράψτε επίσης τα μη συνταγογραφούμενα φάρμακα (πχ αναλγητικά, βιταμίνες, υπακτικά κλπ.), τα φυτικά σκευάσματα ή οποιαδήποτε άλλα σκευάσματα με θεραπευτική δράση λαμβάνει ο ασθενής πχ κολλύρια, αλοιφές, εισπνεόμενα, αντισυλληπτικά . Ανάλογα με το ιστορικό μπορεί να χρειαστεί να καταγράψετε και φάρμακα τα οποία ο ασθενής έχει διακόψει πχ λόγω μιας παρενέργειας ή λόγω μη αποτελεσματικότητας.

3.4.3 Αλλεργίες

Το ιστορικό των αλλεργιών πρέπει να καταγράφεται ξεχωριστά από τα φάρμακα. Καταχωρείτε την περιγραφή της αλλεργικής αντίδρασης καθώς και τα πιθανά αίτια και τον τρόπο διάγνωσης.

Αν ο ασθενής αναφέρει αναφυλακτική αντίδραση (δηλαδή δυνητικά θανατηφόρα αλλεργική αντίδραση) επισημάνετε ιδιαίτερα τα πιθανά αίτια πχ τσίμπημα μέλισσας, πενικιλίνη κλπ.
Τέλος για κάθε αλλεργία σημειώνεται και ο χαρακτηρισμός για το αν είναι βεβαιωμένη ή πιθανολογούμενη.

3.4.4 Κοινωνικό ιστορικό-Συνήθειες

Ο τρόπος που ζει ο άρρωστος προκαλεί, αρκετά συχνά, προβλήματα στην υγεία του. Άτομα που εργάζονται κάτω από συνθήκες μεγάλης ψυχολογικής φόρτισης, εμφανίζουν σε αυξημένη συχνότητα καρδιαγγειακά επεισόδια, ενώ άτομα που εργάζονται σε περιβάλλον που έχει χημικά απόβλητα ή σκόνες, εμφανίζουν σε αυξημένη συχνότητα χρόνιες πνευμονικές παθήσεις. Η κατάχρηση καπνού προδιαθέτει σε παθήσεις του καρδιαγγειακού και του αναπνευστικού συστήματος, ενώ η κατάχρηση αλκοόλης συνδέεται με συχνότερη εμφάνιση κίρρωσης του ήπατος, πεπτικού έλκους και παγκρεατίτιδας.

3.4.5 Ατομικό αναμνηστικό-Νοσήματα/Χειρουργικές επεμβάσεις

Το ατομικό αναμνηστικό περιλαμβάνει την καταγραφή των προηγούμενων νοσημάτων του αρρώστου. Τα νοσήματα του παρελθόντος είναι πιθανό να έχουν σχέση με την παρούσα νόσο του αρρώστου, αλλά είναι επίσης πιθανό να μην έχουν καμία σχέση με αυτή.
Ο γιατρός είναι υποχρεωμένος να διερευνήσει το ενδεχόμενο η παρούσα νόσος να οφείλεται σε νοσήματα του παρελθόντος.

3.4.6 Οικογενειακό ιστορικό

Υπάρχουν νοσήματα τα οποία είτε μεταβιβάζονται κληρονομικά, είτε έχουν οικογενή χαρακτήρα, δηλαδή εμφανίζονται σε μεγάλη συχνότητα στα μέλη της ίδιας οικογένειας.
Η καταγραφή των νοσημάτων των συγγενών του αρρώστου, είναι δυνατόν να βοηθήσει στη διάγνωση της παρούσας νόσου.
Καταγράψτε την ηλικία και την κατάσταση υγείας ή την ηλικία και την αιτία θανάτου όλων των άμεσων συγγενών.

3.4.7 Κύριο ενόχλημα – Συμπτώματα εισόδου

Κύρια ενοχλήματα θεωρούνται εκείνα που οδήγησαν τον άρρωστο στο γιατρό ή στο νοσοκομείο. Πολλοί προτιμούν να καταγράφουν εδώ την περιγραφή των συμπτωμάτων με τον ίδιο τρόπο ακριβώς (τα ίδια λόγια) που τα περιγράφουν οι ίδιοι οι άρρωστοι. Τα κύρια ενοχλήματα καταγράφονται επιγραμματικά και αναλύονται στη συνέχεια, ώστε να διαμορφώσουν την παρούσα νόσο του αρρώστου.
Η επτάδα των συμπτωμάτων:

1. Εντόπιση: Πού είναι; Ακτινοβολεί; Επεκτείνεται;
2. Ποιότητα: Πώς είναι; Πώς το αισθάνεσθε;
3. Ποσότητα/Σοβαρότητα: Πόσο σοβαρό, έντονο είναι (πόνος 1-10)
4. Χρονολογία: Πότε άρχισε; Πόσο διαρκεί; Πόσο συχνά συμβαίνει;
5. Συνθήκες πρόκλησης: Περιβαλλοντικοί παράγοντες, προσωπικές δραστηριότητες, συναισθηματικές αντιδράσεις, άσκηση, συγκινησιακή φόρτιση
6. Παράγοντες που το ανακουφίζουν/επιδεινώνουν: Τί το βελτιώνει ή τί το επιδεινώνει;
7. Συνοδά συμπτώματα: μήπως έχετε παρατηρήσει άλλα συνοδά συμπτώματα;

3.4.8 Παρούσα νόσος-Αρχική αξιολόγηση

Παρούσα νόσο είναι το σύνολο των συμπτωμάτων του αρρώστου, τα οποία πρέπει να αναλυθούν από τον γιατρό και να σχηματίσουν μια συγκεκριμένη εικόνα.

Οι ερωτήσεις που θα υποβληθούν στον άρρωστο, αποβλέπουν στην ανάλυση των χαρακτήρων των συμπτωμάτων, ώστε αυτά να γίνουν σαφή και ξεκάθαρα. Εδώ καταγράφεται με χρονολογική σειρά το ιστορικό της παρούσας νόσου. Είναι ευνόητο ότι έχει γίνει μία αρχική αξιολόγηση πραγματοποιώντας φυσική εξέταση και έχουμε ήδη διαμορφώσει μία άποψη για την πιθανή διάγνωση.

3.4.9 Κλινική εξέταση

Εδώ καταγράφονται τα ευρήματα της κλινικής εξέτασης την συγκεκριμένη στιγμή και γίνεται ανασκόπηση κατά συστήματα, τα οποία είναι τα παρακάτω:

- Κεφαλή-Τράχηλος
- Άκρα
- Αναπνευστικό
- Κυκλοφορικό
- Πεπτικό
- Αιμοποιητικό-Λεμφικό
- Κινητικό
- Νευρικό
- Ουροποιογεννητικό

3.4.10 Διάγνωση

Γράφεται η διάγνωση στην οποία καταλήγουμε. Πρέπει να είναι διάγνωση συγκεκριμένη, τοπογραφική, παθολογοφυσιολογική και αιτιολογική.

4 ΠΡΟΤΥΠΑ ΗΛΕΚΤΡΟΝΙΚΟΥ ΙΑΤΡΙΚΟΥ ΦΑΚΕΛΟΥ

4.1 Εισαγωγή

Καταρχήν, γνωρίζουμε ότι υπάρχουν τρείς κυρίως οργανισμοί με παγκόσμια πρότυπα, οι οποίοι σχετίζονται άμεσα με τον ηλεκτρονικό ιατρικό φάκελο (ΗΙΦΑ). Πρόκειται για τους:

• CEN (Committee European Normalization – The European Standards Organization)

• ISO (International Standards Organization)

• HL7 (Health Level 7) που εδρεύει στις Η.Π.Α. και έχει παραρτήματα σε περισσότερες από 27 χώρες στον κόσμο.

Στις Η.Π.Α υπάρχουν αρκετοί οργανισμοί που ασχολούνται με την εξέλιξη του ηλεκτρονικού ιατρικού φακέλου (ΗΙΦΑ) και τα πρότυπα για την αναπαράσταση της γνωστικής πληροφορίας, από τους οποίους, οι πιο σημαντικοί είναι:

• ASTM . Ο ASTM είναι περισσότερο ενεργός στον τομέα των standards περιεχομένου του ΗΙΦΑ,

• OMG HDTF (Object Management Group Health Domain Task Force).Ο HDTF έχει κάνει εκπληκτική συνεισφορά στην εξέλιξη των προδιαγραφών «ανοικτών υπηρεσιών» όπως COAS (Clinical Observation Access Service), PIDS (Person Identification Service), TQS/LQS (Terminology/Lexicon Query Service) και RAD (Resource Access Service). Το DICOM (Digital Imaging and Communications) είναι ένα πρότυπο που εξελίχθηκε από το Αμερικανικό Κολλέγιο Ραδιολογίας και καθορίζει το σχήμα και τη διάταξη των μηνυμάτων καθώς και τα πρότυπα της επικοινωνίας για διαγνωστικές και θεραπευτικές εικόνες.

Για την μεταφορά, παρουσίαση, διασύνδεση και αποθήκευση δεδομένων υγείας έχουν προταθεί και ήδη υπάρχουν πολλά πρότυπα τα οποία χρησιμοποιούνται από διάφορους οργανισμούς. Επιπλέον, υπάρχουν διάφορα πρότυπα για την κατηγοριοποίηση ασθενειών, ιών και ιατρικών όρων.

4.2 CEN/TC 251

Η Ευρωπαϊκή Επιτροπή Τυποποίησης (Committee European Normalization – The European Standards Organization - CEN) έχει δημοσιεύσει ένα PreStandard για την αρχιτεκτονική ηλεκτρονικού ιατρικού φακέλου (Electronic HealthCare Record) με την ονομασία ENV 13606, το οποίο δημοσιεύθηκε το 1999/2000. Το PreStandard σχηματίστηκε ολοκληρωτικά σύμφωνα με το πρώτο CEN EHR πρότυπο, ENV 12265, το οποίο δημοσιεύθηκε το 1995. Βασίστηκε εξολοκλήρου στο project GEHR (Good Electronic Health Record) αλλά ποτέ δεν υλοποιήθηκε. Το ENV 13606 ορίζει γενικές δομές πληροφορίας και χαρακτηριστικά κοινά σε κάθε ηλεκτρονικό ιατρικό φάκελο, δηλαδή ένα λογικό μοντέλο, χωρίς να καθορίζει ακριβώς τι ιατρική πληροφορία θα περιέχει ή πως θα υλοποιηθεί. Το ENV 13606 είναι το μόνο πρότυπο ειδικά για ηλεκτρονικό ιατρικό φάκελο στον κόσμο και δεν έχει υλοποιηθεί σε κάποιο σύστημα, αποτελεί όμως αναφορά.

4.3 ISO/TC 215

Ο οργανισμός ISO έχει ιδρύσει την Τεχνική Επιτροπή 215 (TC215) με στόχο την προτυποποίηση στον τομέα της ιατρικής πληροφορικής. Τα πρότυπα (standards) κατά ISO/TC 215 είναι η παγκόσμια κορυφή για τον ΗΙΦΑ, όπως και για άλλα standards που αφορούν στην Ιατρική Πληροφορική.

Κάποιοι οργανισμοί έκαναν χρήση ήδη συγκεκριμένων προτυποποιήσεων διεθνών οργανισμών ,όπως είναι το ISO. Μερικοί από τους οργανισμούς που το έκαναν αυτό είναι οι DICOM ,IEEE, CEN, HL7. Είναι γνωστό άλλωστε πως οι τρείς τελευταίοι οργανισμοί έχουν ειδική συμφωνία με τον ISO που εξουσιοδοτούν τις υπάρχουσες προτυποποιήσεις προκειμένου να γίνουν πρότυπα κατά ISO.
Η πληροφορία του ασθενή είναι διαθέσιμη όποτε και όταν χρειαστεί μέσω του ΗΙΦΑ, που παρέχει ποιότητα στο τομέα της υγείας, διευκολύνει την αποτελεσματικότητα των κλινικών παρατηρήσεων και βελτιώνει την ασφάλεια των ασθενών.
Επί του παρόντος, υπάρχουν δύο διαφορετικές όψεις από το σκοπό του ΗΙΦΑ κατά (ISO/TS 18308:2004), που ισχύουν από τον Οκτώβριο του 2002, μετά από τη σύσκεψη που έγινε στη Βαλτιμόρη των Η.Π.Α. Η πρώτη όψη καλείται "ουσιώδης ηλεκτρονικός ιατρικός φάκελος" (Core EHR) ενώ η δεύτερη καλείται "εκτεταμένος ηλεκτρονικός ιατρικός φάκελος" (Extended EHR).
Τα θεμελιώδη χαρακτηριστικά του "ουσιώδη ηλεκτρονικού ιατρικού φακέλου" είναι αυτά που ασχολούνται με το απλό αντικείμενο της ιατρικής φροντίδας, έχει πρωταρχικό σκοπό να παρέχει μία καταγεγραμμένη εγγραφή για παροντική ή μελλοντική χρήση από το ιατρικό προσωπικό.
Ο "εκτεταμένος ηλεκτρονικός ιατρικός φάκελος" δεν περιλαμβάνει μόνο κλινικές πληροφορίες αλλά και ουσιώδη χαρακτηριστικά που ονομάζονται τοπία ιατρικής πληροφορικής. Περισσότερα από τα περιεχόμενα του εκτεταμένου ΗΙΦΑ καθορίζονται από δεύτερης σημασίας χρήση όπως πληρωμή, πολιτική και σχεδιασμός, στατιστική ανάλυση, επικύρωση, διοίκηση υπηρεσιών υγείας, έλεγχος ποιότητας και έρευνα για την ανάλυση και τον υπολογισμό των νέων διαγνωστικών μορφών, καθώς και επιδημιολογικές μελέτες.

4.4 HL7

Το πρωτόκολλο HL7 − από τον ομώνυμο μη κερδοσκοπικό οργανισμό − ορίζει το περιεχόμενο και τη δομή των ιατρικών δεδομένων και δεδομένων ασφάλισης υγείας, τα οποία ανταλλάσσονται μεταξύ συστημάτων υγείας ή και οργανισμών. Σημαντικό είναι πως το πρωτόκολλο αυτό δεν είναι ανοιχτό, έτσι κάθε χρήστης του πρωτοκόλλου πρέπει να πληρώνει ετήσια συνδρομή κάτι που θα αλλάξει στο κοντινό μέλλον.
Το Health Level Seven (HL7) είναι ένα σύνολο από ανοιχτά πρότυπα, που επιτρέπει σε ετερογενή ιατρικά πληροφοριακά συστήματα να επικοινωνούν μεταξύ τους. Το HL7 προτυποποιεί τα πρωτόκολλα και τις δομές για την ανταλλαγή μηνυμάτων ιατρικού ενδιαφέροντος στο επίπεδο της εφαρμογής, το 7ο Επίπεδο του μοντέλου ISO, δηλαδή είναι ανεξάρτητο από συγκεκριμένες πλατφόρμες και τεχνολογίες.
Ο HL7 ιδρύθηκε το 1987 με σκοπό να αναπτύξει πρότυπα για την ηλεκτρονική ανταλλαγή ιατρικών, οικονομικών και διαχειριστικών πληροφοριών μεταξύ ανεξάρτητων υπολογιστικών συστημάτων προσανατολισμένων στην υγεία, όπως πληροφοριακά συστήματα νοσοκομείων, συστήματα κλινικών εργαστηρίων, συστήματα επιχειρήσεων και συστήματα φαρμακείων.
Το HL7 είναι το πλέον ευρέως χρησιμοποιημένο πρότυπο ανταλλαγής πληροφοριών μέσω μηνυμάτων σε κλινικό περιβάλλον. Χρησιμοποιείται σε όλες τις ηπείρους.
Σχεδόν όλα τα ευφυή διαγνωστικά μηχανήματα (ιατροτεχνολογικός εξοπλισμός) υποστηρίζουν το HL7 και σχεδόν όλα τα ιατρικά πληροφοριακά συστήματα υψηλού επιπέδου είναι σε θέση να στείλουν και να λάβουν τα κατάλληλα HL7 μηνύματα, χρησιμοποιώντας τους κανόνες ανταλλαγής μηνυμάτων του πρωτοκόλλου του HL7.

Το HL7 είναι ξεκάθαρα το πιο ώριμο πρότυπο ανταλλαγής πληροφοριών μέσω μηνυμάτων. Η έρευνα από την ακαδημαϊκή κοινότητα και την βιομηχανία και τις εταιρίες συμβούλων οδήγησε σ' αυτό το πρότυπο, την κυριότητα του οποίου την κατέχει ο μη κερδοσκοπικός οργανισμός Health Level Seven ο οποίος έχει τοπικά υποκαταστήματα σε όλες σχεδόν τις χώρες της Ευρώπης, στις Ηνωμένες Πολιτείες της Αμερικής, στην Αυστραλία / Νέα Ζηλανδία, την Ασία και στη ζώνη του Ειρηνικού. Το πρότυπο HL7 έχει αναγνωριστεί από πολλά εθνικά ιδρύματα προτυποποίησης, όπως ο ANSI (USA) και ο DIN(Γερμανία).
Στην Ελλάδα ιδρύθηκε και λειτουργεί από το 2003 το παράρτημα (μη κερδοσκοπικού χαρακτήρα) του διεθνούς οργανισμού Health Level Seven (HL7) με την επωνυμία "HL7 Hellas" www.hl7.org.gr. Ο ιδρυτικός πυρήνας περιλαμβάνει δεκαπέντε (15) διακεκριμένα ονόματα φορέων τόσο από τον Πανεπιστημιακό όσο και από τον χώρο των εταιριών Ιατρικής Πληροφορικής και Τεχνολογίας. Η συμβολή του προτύπου HL7 στην διατηρησιμότητα και διαλειτουργικότητα των πληροφορικών συστημάτων υγείας είναι τεράστια.

5 ΠΛΗΡΟΦΟΡΙΑΚΑ ΣΥΣΤΗΜΑΤΑ ΥΓΕΙΑΣ

5.1 Εισαγωγή

Η δημιουργία υποδομής στο χώρο της υγείας ενός πληροφοριακού συστήματος, απαιτεί την συνεργασία μεθόδων παροχής υπηρεσιών του παρελθόντος και τεχνολογιών του σήμερα. Τα παραπάνω προϋποθέτουν την θέσπιση και εφαρμογή κωδικών και προτύπων, τα οποία θα καθορίζουν έναν συγκεκριμένο τρόπο συλλογής, συνεργασίας και παρουσίασης των ιατρικών δεδομένων από διαφορετικά πληροφοριακά συστήματα. Απαραίτητη λοιπόν θεωρείται η ύπαρξη ενός δικτύου ώστε να συνεργάζονται όλα αυτά τα συστήματα.

Τα Πληροφοριακά Συστήματα Υγείας είναι πληροφοριακά συστήματα που χρησιμοποιούνται μέσα σε ένα οργανισμό παροχής υπηρεσιών υγείας για να διευκολύνουν την επικοινωνία, να ολοκληρώνουν την πληροφορία, βοηθούν στην διατήρηση αρχείων.

5.2 Στόχοι Πληροφοριακών Συστημάτων Υγείας

Επιγραμματικά θα αναφέρουμε παρακάτω μερικούς από τους κυριότερους στόχους των πληροφοριακών συστημάτων στον χώρο της υγείας:

- Ψηφιακή διαχείριση και επεξεργασία ιατρικής πληροφορίας για την προαγωγή της υγείας και τη διαχείριση του ασθενούς, αλλά και για την υποστήριξη κλινικής έρευνας, βασικής έρευνας και εκπαίδευσης
- Διασύνδεση τμημάτων και φορέων παροχής υπηρεσιών υγείας
- Να υποστηρίζουν τον γιατρό, τη νοσοκόμα και το διοικητικό προσωπικό αλλά κυρίως να ενισχύουν τον ασθενή και τον πολίτη σε θέματα υγείας.
- Να διαχειριστούν νέα είδη πληροφορίας (π.χ. γονιδιακή) και να χρησιμοποιήσουν νέα είδη τεχνολογίας (π.χ. συσκευές που φέρονται στο σώμα)

5.3 Ιστορία και Εξέλιξη

Τα πληροφοριακά συστήματα υγείας αποτελούν ένα σύνολο από υλικό και λογισμικό που σκοπό έχει την υποβοήθηση του ιατρικού και παραϊατρικού προσωπικού, τη διαχείριση των οικονομικών και λογιστικών λειτουργιών σε έναν οργανισμό υγείας και γενικότερα τη βελτίωση των παρεχόμενων υπηρεσιών υγείας. Μπορούν να αποτελούνται από ενσύρματες ή ασύρματες συσκευές, από μικρά ή μεγαλύτερα δίκτυα υπολογιστών, να έχουν πρόσβαση στο διαδίκτυο, να συνδέονται με άλλα πληροφοριακά συστήματα, να προσφέρουν υπηρεσίες τηλεϊατρικής ή υποστήριξης αποφάσεων κ.ά..

Είναι σαφές πως τα πληροφοριακά συστήματα υγείας αποτελούν σημαντικό εφόδιο ενός οργανισμού υγείας. Αυτοματοποιούν πολλές από τις διαδικασίες των παραδοσιακών συστημάτων αρχειοθέτησης και λογιστικής καταγραφής ενώ ταυτόχρονα μπορούν να διαχειριστούν τα ραντεβού με τους ιατρούς, την παραγγελία και τα αποτελέσματα των εργαστηριακών εξετάσεων και γενικότερα όλες τις ενέργειες που χρειάζονται από την εισαγωγή ενός ασθενούς έως το εξιτήριό του.

Από τα σημαντικότερα υποσυστήματα των πληροφοριακών συστημάτων υγείας είναι τα συστήματα διαχείρισης ηλεκτρονικών φακέλων ασθενών. Τα υποσυστήματα αυτά αποθηκεύουν, αρχειοθετούν και επεξεργάζονται κάθε πληροφορία που αφορά στην υγεία του ασθενούς. Τα συστήματα διαχείρισης ηλεκτρονικών φακέλων ασθενών είναι άμεσα συνδεδεμένα με την έννοια της ιδιωτικότητας καθώς διαχειρίζονται ευαίσθητα δεδομένα, όπως τα αποτελέσματα ιατρικών

εξετάσεων, το ιατρικό ιστορικό του ασθενούς, τις διαγνώσεις και τα φάρμακα που χορηγούνται σε έναν ασθενή. Όλες αυτές οι πληροφορίες θα πρέπει να παρουσιάζονται στο ιατρικό και παραϊατρικό προσωπικό με τρόπο σαφή, ολοκληρωμένο και δομημένο έτσι ώστε να μπορούν να εξαχθούν συμπεράσματα για την κατάσταση της υγείας του ασθενούς. Επίσης, η επεξεργασία πολλών ηλεκτρονικών φακέλων ασθενών μπορεί να εξάγει συμπεράσματα για επιδημιολογικές μελέτες, στατιστικά στοιχεία για την υγεία των ασθενών σε μία περιοχή ή για την αποτελεσματικότητα μιας θεραπευτικής αγωγής. Ταυτόχρονα, τα συστήματα αυτά θα πρέπει να προσφέρουν υψηλό βαθμό ασφάλειας έτσι ώστε η πρόσβαση στα δεδομένα να επιτρέπεται μόνο σε εξουσιοδοτημένα άτομα και να προστατεύεται η ιδιωτικότητα των ασθενών.

5.4 Πάροχοι Ηλεκτρονικών Υπηρεσιών Υγείας

Υπάρχουν διάφοροι πάροχοι ηλεκτρονικών υπηρεσιών για ασθενείς, γιατρούς και ιατρικά ιδρύματα. Με επίκεντρο τις ΗΠΑ όπου η μεγαλύτερη βιομηχανία είναι αυτή της Υγείας θα αναφερθούν μερικά παραδείγματα.

5.4.1 Microsoft HealthVault

Το HealthVault από την Microsoft είναι μια διαδικτυακή πλατφόρμα για την διαχείριση και αποθήκευση δεδομένων ασθενών, καθώς και για την επικοινωνία τους με ιδρύματα και ιατρούς. Είναι διαθέσιμη στους πολίτες του Ηνωμένου Βασιλείου και στις ΗΠΑ. Επιπλέον, υποστηρίζονται συσκευές οι οποίες μπορούν να επικοινωνούν με τη πλατφόρμα και να προσθέτουν δεδομένα για τους χρήστες. Μπορεί να επιτραπέι να έχουν πρόσβαση πολλαπλά άτομα και η εξουσιοδότηση μπορεί να δοθεί κάνοντας χρήση των λογαριασμών Facebook, Windows Live.

5.4.2 Dossia

Το Dossia λειτουργεί με ανοιχτό λογισμικό και δίνεται πλήρης έλεγχος στο χρήστη πάνω στα δεδομένα του, τα οποία μπορεί να τα εξάγει. Η δημιουργία της πλατφόρμας αυτής είναι αποτέλεσμα της συνεργασίας διάφορων επιχειρηματικών κολοσσών των ΗΠΑ για την παροχή Ηλεκτρονικού Ιατρικού Φακέλου στους εργαζόμενους τους. Το σύστημα δίνει τη δυνατότητα στους χρήστες να συγκεντρώσουν ψηφιακά αντίγραφα των ιατρικών δεδομένων τους και να δημιουργήσουν τα δικά τους φορητά Ιατρικά έγγραφα.

5.4.3 World Medical Card

Η World Medical Card είναι ένα προϊόν της Νορβηγικής εταιρείας World Medical Technologies. Το World Medical Card παρέχει τρεις βασικές υπηρεσίες:

- Ένα προφίλ υγείας web-based, που επιτρέπει στο χρήστη να προσθέσετε και να επεξεργαστείτε τα προσωπικά δεδομένα υγείας (onWeb)
- Μία εφαρμογή για κινητά που δίνει πρόσβαση στα ίδια δεδομένα, μεταφρασμένο σε μια γλώσσα της επιλογής (onMobile)
- Μια σφραγισμένη κάρτα η οποία περιλαμβάνει μια συνοπτική αναφορά των πληροφοριών για την υγεία του κατόχου(onCard)

Ο κύριος σκοπός του προϊόντος είναι να παρέχει την δυνατότητα στο χρήστη να έχει πρόσβαση και να διαχειρίζεται ζωτικής σημασίας προσωπικά στοιχεία, όπως:

- Διαγνώσεις
- Ιατρικό Ιστορικό

- Φαρμακευτικές Αγωγές και ανεπιθύμητες ενέργειες του φαρμάκου
- Αλλεργίες και αντιδράσεις σε φάρμακα
- Εμβολιασμούς
- Επαφές για επείγουσες περιστάσεις
- Σημαντικά προσωπικά έγγραφα (από φόρτωμα αρχείου)

5.5 Πληροφοριακά Συστήματα Υπηρεσιών Υγείας στην Ελλάδα

Είναι γεγονός ότι ο τομέας υγείας στην χώρα μας ιδιωτικός και δημόσιος υστερεί εντυπωσιακά στην εφαρμογή των Τεχνολογιών της Πληροφορικής και των Επικοινωνιών. Παρόλο αυτά τα τελευταία χρόνια παρατηρείται μία σημαντική κινητικότητα, με αποτέλεσμα να μπορούμε να κάνουμε αναφορά στις παρακάτω υπηρεσίες.

5.5.1 MEDILIG

Το MEDILIG χρησιμοποιεί ανοιχτή σχεσιακή βάση δεδομένων και βασίζεται σε τεχνολογία Microsoft με ανοιχτό κώδικα.

Τα λειτουργικά χαρακτηριστικά που προσφέρει είναι: Καταχώρηση δημογραφικών στοιχείων, πιστοποίηση ασθενούς, εισαγωγές-επεμβάσεις-επισκέψεις ασθενούς, ιστορικό, αποθήκευση και διαχείριση εργαστηριακών αποτελεσμάτων, απεικονιστικά, πολυμεσικά αρχεία, συνταγογράφηση, καταγραφή κλινικής εξέτασης κλπ.

5.5.2 Πανάκεια

Είναι ένα σύστημα διαχείρισης πληροφοριών κατά τη νοσηλεία του ασθενούς (bed management system), προσφέροντας στον επαγγελματία υγείας τη σωστή πληροφορία, τη στιγμή που τη χρειάζεται, με ένα τρόπο που να μπορεί εύκολα να εποπτεύσει το περιστατικό νοσηλείας. Είναι πλήρως προσαρμοσμένο στις ροές πληροφορίας και στις διαδικασίες των κλινικών, εξασφαλίζοντας σε αυτές ένα paperless περιβάλλον. Ταυτόχρονα δίνει τη δυνατότητα στον θεράποντα ιατρό να έχει ολοκληρωμένη εικόνα για τον ασθενή του, καθώς μπορεί και συγκεντρώνει πληροφορίες από όλα τα νοσοκομεία της Κρήτης.

5.5.3 Professional clinic

Το Professional clinic είναι ένα από τα πιο προηγμένα ιατρικά προγράμματα διαχείρισης ασθενών. Είναι εύχρηστο, απλό και ολοκληρωμένο εργαλείο το οποίο έχει δημιουργηθεί με σκοπό να οργανώσει και να διαχειρίζεται όλες τις ανάγκες ενός ιατρείου, αφού είναι άκρως παραμετροποιήσιμο. Χρησιμοποιώντας όλες τις τελευταίες τεχνολογικές δυνατότητες αλλά διατηρώντας συγχρόνως ένα απλό και φιλικό στο χειρισμό περιβάλλον.

Τυχόν αλλαγές που επιθυμεί ο χειριστής γίνονται άμεσα και γρήγορα, χωρίς να είναι απαραίτητη η επίσκεψη τεχνικού, μέσω διαδικτυακών αναβαθμίσεων. Επιπλέον το Professional Clinic αποτελεί ένα από τα πιο πρωτοποριακά προγράμματα του κλάδου καθώς προφέρει τη δυνατότητα πρόσβασης στις πληροφορίες του ιατρείου, ακόμη και σε απομακρυσμένες θέσεις, όπου υπάρχει πρόσβαση στο διαδίκτυο.

5.5.4 DoctorsNet

Το πρόγραμμα DoctorsNet καλύπτει το σύνολο των αναγκών ενός ιατρείου. Είναι απλό στη χρήση και μπορεί να παραμετροποιηθεί για να καλύψει σχεδόν όλες τις ανάγκες ενός σύγχρονου ιατρείου.

Τα βασικά χαρακτηριστικά του προγράμματος είναι :

- Διαχείριση Ιατρικού Φακέλου Υγείας Ασθενή

Η εφαρμογή τηρεί τα βασικά δημογραφικά και ασφαλιστικά στοιχεία, τα πλήρη ιατρικά – νοσηλευτικά δεδομένα του Ασθενή: Ατομικό και Κληρονομικό Ιστορικό, Φάρμακα, Χειρουργεία, Διαγνώσεις. Τηρεί τα αποτελέσματα όλων των κλινικών, διαγνωστικών και απεικονιστικών εξετάσεων ανά επίσκεψη, καλύπτοντας όλες τις απαιτήσεις τήρησης του ιατρικού απορρήτου.

- Διαχείριση Φαρμάκων

Περιέχει όλα τα φάρμακα με τις τιμές λιανικής πώλησης. Ο γιατρός έχει τη δυνατότητα να καταχωρεί τα φάρμακα σε κάθε επίσκεψη και να τα εκτυπώνει.

- Μικροβιολογικές Εξετάσεις

Το τμήμα των Μικροβιολογικών Εξετάσεων περιέχει όλες τις μικροβιολογικές εξετάσεις με τις φυσιολογικές τιμές τους καθώς επίσης και ένα λειτουργικό και απλό περιβάλλον εργασίας που καλύπτει όλες τις ανάγκες ενός Μικροβιολόγου.

Παρέχεται η δυνατότητα προσαρμογής των εκτυπωτικών του προγράμματος στις ιδιαίτερες ανάγκες σας.

Παρέχεται η δυνατότητα σύνδεσης με διάφορους τύπους αναλυτών (αιματολογικούς, βιοχημικούς κλπ.) έτσι ώστε τα αποτελέσματα των διαφόρων εξετάσεων να εισάγονται αυτόματα στο πρόγραμμα.

- Διαχείριση ραντεβού ιατρείου

Προσφέρεται η δυνατότητα διαχείρισης ραντεβού του Outlook μέσα από το πρόγραμμα DoctorsNet.

- Υποστηρίζει το νέο ηλεκτρονικό αριθμό του ΙΚΑ (ΑΜΚΑ)

Υπάρχει η δυνατότητα καταχώρησης του νέου αριθμού κοινωνικής ασφάλισης ΑΜΚΑ.

- Αυτόματη ενημέρωση Φαρμάκων

Ενημερώνεται αυτόματα από το Internet για τα νέα φάρμακα και τις νέες τιμές πώλησης.

- Διαχείριση Χειρουργείων

Παρέχεται η δυνατότητα καταχώρησης των χειρουργείων των ασθενών. Το τμήμα των χειρουργείων τηρεί τα εξής στοιχεία για κάθε χειρουργείο: ημερομηνία εισόδου, ημερομηνία εξόδου, διάγνωση εισόδου, ιστορικό, είδος χειρουργείου, συστήσας ιατρός, θεράπων ιατρός, αναισθησιολόγος.

- Ενημέρωση ασθενών μέσω SMS

Παρέχεται η δυνατότητα ενημέρωσης των ασθενών μέσω SMS στις εξής περιπτώσεις:

Ενημέρωση των ασθενών για παραλαβή εργαστηριακών αποτελεσμάτων

Ενημέρωση των ασθενών για προγραμματισμένο ραντεβού

Ευχές στην ονομαστική εορτή και στα γενέθλια

Ενημέρωση για ιατρικά θέματα

- Δέχεται σύνδεση με οποιαδήποτε ψηφιακή συσκευή στοιχείων.
- Συνεχής ενημέρωση από το ίντερνετ για νέες εκδόσεις και βελτιώσεις.

5.5.5 Σειρά Συστημάτων SmartHealth της Apollo Information Technologies

Η Apollo παρέχει σύγχρονες λύσεις στον χώρο των επιχειρήσεων, μέσα από την ανάπτυξη, εφαρμογή και υποστήριξη ολοκληρωμένων πληροφοριακών συστημάτων, καθώς επίσης και συμβουλευτικές υπηρεσίες και εφαρμογές στο επίπεδο του e-business.

Η σειρά δίνει ιδιαίτερο βάρος στην βοήθεια λήψης αποφάσεων στο διοικητικό-οικονομικό τομέα του φορέα, στην διασύνδεση των φορέων υγείας & των ιατρών σε επίπεδο επιστημονικό & επαγγελματικό, στην δημιουργία κέντρων πρόληψης & ενημέρωσης του πληθυσμού και στην δημιουργία κέντρων αναφοράς διαγνωστικού & θεραπευτικού χαρακτήρα.

Η σειρά SmartHealth υποστηρίζει την :

- Κάρτα Υγείας με ευρεία διάδοση & πολυσυλλεκτικό αντικείμενο υπηρεσιών.
- Τράπεζα Ιατρικού Φακέλου.
- Εγκαθίδρυση ενιαίου δικτύου ανταλλαγής ιατρικών δεδομένων από τους ιατρούς & τους φορείς που εμπλέκονται στο σύστημα
- Εγκαθίδρυση δικτύου ιατρικής διασύνδεσης σε επίπεδο επιστημονικής υποστήριξης
- Δημιουργία κέντρου αναφοράς διαγνωστικού & θεραπευτικού χαρακτήρα.
- Δημιουργία τμήματος επιστημονικής υποστήριξης & συνεχιζόμενης κατάρτισης του ιατρικού & νοσηλευτικού προσωπικού.
- Δημιουργία ειδικού τμήματος παρηγορητικής υποστήριξης ασθενών με χρόνια νοσήματα.

5.5.6 TMS Iasis

Το πρόγραμμα Ιατρικού Λογισμικού TMS Iasis και οι παραλλαγές του ανά ιατρική ειδικότητα, συνιστούν το βασικό άξονα δράσης της εταιρίας. Η εφαρμογή έχει εξ΄ολοκλήρου σχεδιαστεί και αναπτυχθεί από την TMS με στόχο την ολοκληρωμένη κάλυψη των αναγκών του επιστήμονα υγείας, βάσει των ιδιαίτερων αναγκών του και αποσκοπεί στην πλήρη διαχείριση των πληροφοριών και διαδικασιών που αποτελούν αναπόσπαστα στοιχεία της κάθε ειδικότητας.

Κύριες λειτουργίες που προσφέρει το προϊόν είναι:

ΚΑΡΤΕΛΑ ΑΣΘΕΝΩΝ

- Εύκολη αναζήτηση βάσει πεδίων και ειδικών πεδίων
- Αναλυτικά δημογραφικά και ασφαλιστικά στοιχεία ασθενών
- Αναλυτικά στοιχεία θεράποντος και συσταθέντος ιατρού
- Προσθήκη ειδικών πεδίων
- Παρακολούθηση ασθενούς (περιλαμβάνει ιστορικό επισκέψεων και εξετάσεων)

ΙΑΤΡΙΚΗ ΕΞΕΤΑΣΗ

- Αναλυτικές κλινικές εξετάσεις, συμπτώματα, ευρήματα, διάγνωση, θεραπεία
- Αρχεία διαγνωστικών εκθέσεων πραγματοποιθέντων απεικονιστικών εξετάσεων
- Βιομετρικά στοιχεία
- Φαρμακευτική αγωγή

ΙΣΤΟΡΙΚΟ ΑΣΘΕΝΗ – ΠΑΘΟΛΟΓΙΚΟ ΙΣΤΟΡΙΚΟ

- Ατομικό ιστορικό
- Οικογενειακό ιστορικό
- Ιστορικό επισκέψεων, εξετάσεων και διαγνώσεων ανά ειδικότητα
- Προσωπικό και οικογενειακό ιστορικό

- Ατομικό αναμνηστικό
- Κλινική εξέταση
- Νευρικό σύστημα
- Πορεία νόσου

ΕΡΓΑΣΤΗΡΙΑΚΕΣ ΕΞΕΤΑΣΕΙΣ

- Διαχείριση εργαστηριακών εξετάσεων
- Παραμετροποίηση εργαστηριακών εξετάσεων
- Πορεία εργαστηριακών εξετάσεων

ΔΙΑΧΕΙΡΙΣΗ ΟΠΤΙΚΟΥ ΥΛΙΚΟΥ

- Εισαγωγή Εικόνας – Φωτογραφίας – Video από εξωτερικές συσκευές και δυνατότητα αποθήκευσης και διαχείρισης αυτών στην καρτέλα ασθενούς
- Αρχειοθέτηση εικόνας, απεικονιστικού υλικού
- Εκτύπωση της εικόνας σε απαντητικό διαγνωστικό έντυπο

TMS CAPTURE

- Μια εξαιρετικά χρήσιμη εφαρμογή σε έρευνες και παρουσιάσεις. Προσφέρει:
- Λήψη αναλογικής – ψηφιακής εικόνας και video από ιατρικά μηχανήματα
- Επεξεργασία, διαχείριση και οργάνωση ανά περιστατικό και ανά ασθενή
- Χαρακτηρισμός με λέξεις-κλειδιά, ομάδες και υποομάδες

ΕΚΤΥΠΩΣΕΙΣ

- Αναλυτικές εκτυπώσεις διαγνώσεων, εξετάσεων, βεβαιώσεων
- Εύκολη τροποποίηση της μορφής των εκτυπώσεων
- Οι ιατρικές καρτέλες των διαφόρων τμημάτων μπορούν δυναμικά να αλλαχθούν και να προσαρμοστούν στις εκάστοτε ανάγκες

ΓΡΑΜΜΑΤΕΙΑ – ΔΙΑΧΕΙΡΙΣΗ

- Εύκολη παρακολούθηση των ραντεβού που είναι σε αναμονή επίσκεψης, εξέλιξης, ακυρωμένα, σε λίστα αναμονής, ολοκληρωμένα
- Δημιουργία ελεύθερων κειμένων, ιατρικών εκθέσεων, πρότυπων κειμένων καθώς και κειμένων ασθενούς
- Εύκολος έλεγχος οικονομικών και δημογραφικών στοιχείων πελάτη
- Γρήγορη παρακολούθηση
- Έλεγχος προηγούμενων επισκέψεων και ιατρικών αποτελεσμάτων
- Σύστημα διαχείρισης Παραπεμπτικών Ταμείων
- Επικοινωνία μεταξύ των χρηστών
- Λίστες υπενθυμίσεων
- Λίστες επανεξετάσεων

ΒΟΗΘΗΤΙΚΕΣ ΕΦΑΡΜΟΓΕΣ

- Scheduled Backup
- Δυνατότητα Custom Backup
- Manual Restore
- Διαχείριση χρηστών

ΟΙΚΟΝΟΜΙΚΗ ΔΙΑΧΕΙΡΙΣΗ

- Έκδοση παραστατικών

- Προβολή παραστατικών
- Εξετάσεις ανά ταμείο
- Υπόλοιπα ασθενών
- Κατάσταση ημερήσιου ταμείου
- Οικονομική διαχείριση ιατρείου (ανά ασφαλιστικό ταμείο, ανά ασθενή)
- Διαχείριση εξόδων – εσόδων (σύνδεση Φ.ΜΧ)

5.5.7 Vida24

Το Vida 24 της Vidavo είναι ένα ολοκληρωμένο σύστημα τηλεμετρίας, καταγραφής και παρακολούθησης βιολογικών σημάτων και παραμέτρων υγείας, που υποστηρίζει την διαχείριση χρόνιων ασθενειών, αλλά και την πρόληψη μέσω της βελτίωσης δεικτών υγείας που μπορεί να συμβάλλουν στην εκδήλωση χρόνιων νοσημάτων. Αποτελείται από μία διαδικτυακή πλατφόρμα, στην οποία διασυνδέεται ο χρήστης και μπορεί να έχει πρόσβαση σε αναλυτικό ατομικό και οικογενειακό ιατρικό ιστορικό των χρηστών υπηρεσιών υγείας, στο ιστορικό των ιατρικών και εργαστηριακών εξετάσεων και σε πολλές άλλες πληροφορίες και εργαλεία. Το σύστημα αυτό διασυνδέεται με ασύρματες ιατρικές συσκευές(καρδιογράφο, σπιρόμετρο, οξύμετρο, πιεσόμετρο, γλυκοζόμετρο, συσκευή μέτρησης τριγλυκεριδίων και χοληστερόλης, ζυγαριά) και λαμβάνει βιολογικά σήματα σε πραγματικό χρόνο, τα οποία αποθηκεύονται στο cloud και είναι προσβάσιμα την ίδια στιγμή μέσω της διαδικτυακής πλατφόρμας. Η εφαρμογή διατίθεται διαδικτυακά, όπως και σε tablet και κινητό, ενώ χρησιμοποιείται σε περισσότερα από 150 σημεία παγκοσμίως. Είναι το αποτέλεσμα 15ετούς έρευνας και έχει τις πιστοποιήσεις CE, ISO , όπως επίσης ακολουθεί το πρότυπο HL 7.

6 ΚΛΗΡΟΝΟΜΙΚΟΤΗΤΑ ΑΣΘΕΝΙΩΝ

6.1 Εισαγωγή

Δεν είναι μόνο τα εξωτερικά χαρακτηριστικά και ίσως η συμπεριφορά ή ο χαρακτήρας αυτά που κληρονομούμε από τους γονείς μας. Είναι και μια σειρά άλλα, μεταξύ των οποίων η σωματική και διανοητική μας υγεία.

Να μην ξεχνάμε ότι η πρόληψη είναι η καλύτερη θεραπεία!

Λαμβάνοντας υπόψιν τα παραπάνω και καθώς στην συμπλήρωση του ιατρικού φακέλου του ασθενούς ενημερώνουμε το οικογενειακό του ιστορικό σημειώνοντας τις ασθένειες που έχουν παρουσιάσει οι συγγενείς, ξέροντας ποιες από αυτές παρουσιάζουν κληρονομικότητα μπορούμε να παρουσιάσουμε τα άτομα τα οποία λόγω ¨κακού¨ ιστορικού έχουν πιθανότητες να εμφανίσουν στο μέλλον την συγκεκριμένη ασθένεια. Με αυτό τον τρόπο μπορούν να δρομολογηθούν άμεσα προληπτικές εστιασμένες εξετάσεις και περαιτέρω διερεύνηση ιδιαίτερων περιστατικών. Παρακάτω ακολουθεί περιγραφή των δεδομένων που ισχύουν για τέσσερις σοβαρές ασθένειες.

6.2 Σακχαρώδης Διαβήτης τύπου 2

Ο Σακχαρώδης Διαβήτης τύπου 2, ο οποίος αποτελεί το 90% των περιστατικών με διαβήτη σχετίζεται με κληρονομικότητα. Ομάδα υψηλού κινδύνου είναι άτομα που δε γυμνάζονται τακτικά αλλά ζουν καθιστική ζωή, που δεν ακολουθούν σωστή και μετρημένη διατροφή, που έχουν περιττό σωματικό βάρος και ειδικά κεντρική παχυσαρκία. Είναι γεγονός ότι τον παρουσιάζει το 10% των Ελλήνων γύρω στα 60 και μερικοί πολύ νωρίτερα. Αντίθετα, για το Διαβήτη τύπου 1 δεν αναφέρεται κληρονομικότητα.

Η επίδραση των γονιδίων στην ανάπτυξη του Διαβήτη δεν έχει γίνει πλήρως κατανοητή, διαφέρει όμως σύμφωνα με τον Τύπο. Στον Διαβήτη Τύπου 2, διάφορα γονίδια έχουν ανακαλυφθεί, που είτε επηρεάζουν την έκκριση της ινσουλίνης, είτε αυξάνουν την αντίσταση σε αυτήν. Υπολογίζεται ότι αν έχει διαβήτη ο ένας γονιός, το παιδί έχει 40% πιθανότητες να τον παρουσιάσει. Αν έχουν και οι δύο, οι πιθανότητες να αναπτύξει την ίδια νόσο είναι 80%.

Η μεγαλύτερη επίδραση στην ανάπτυξη του Διαβήτη Τύπου 2 φαίνεται ότι επέρχεται από την παρουσία του γονιδίου transcription factor 7–like 2 (TCF7L2). Τα γονίδια που γνωρίζουμε μέχρι τώρα αποτελούν το 10% της κληρονομικής επίδρασης. Δηλαδή έχουμε πολύ λίγα στοιχεία για να μπορέσουμε με μια εξέταση γονιδίων να βρούμε ποιος κινδυνεύει πραγματικά να αναπτύξει Διαβήτη Τύπου 2.

Ο Σακχαρώδης Διαβήτης είναι πολυγονιδιακή πάθηση και δεν υπάρχει γενετικό τεστ. Γι' αυτό το λόγο ο προληπτικός έλεγχος είναι σημαντικός. Αν έχουν διαβήτη οι γονείς σας, μπορείτε να κάνετε στα 20 ή στα 30 σας μία συνηθισμένη εξέταση σακχάρου, η οποία κατά κανόνα θα δείξει το σάκχαρό σας φυσιολογικό. Έπειτα όμως θα πρέπει να κάνετε και μία εξέταση «καμπύλης σακχάρου», που ίσως φανερώσει αντίσταση στην ινσουλίνη - αυτό δείχνει από νωρίς ότι έχετε προδιάθεση να εκδηλώσετε αργότερα διαβήτη Τύπου 2.

6.3 Θυρεοειδής

Στην εμφάνιση της νόσου της θυρεοειδίτιδας του Hashimoto, θεωρείται ότι εμπλέκονται γενετικοί παράγοντες.

Αυτό που μέχρι σήμερα έχει αποδειχθεί είναι ότι γονιδιακοί πολυμορφισμοί, δηλαδή παραλλαγές σε συγκεκριμένα γονίδια, σχετίζονται με την ανάπτυξη της νόσου του Hashimoto. Αυτά είναι τα

γονίδια CTLA4 (cytotoxic T lymphocyte-associated antigen-4) και PTPN2 (protein tyrosine phosphatase-22). Τα προϊόντα των γονιδίων αυτών είναι πρωτεϊνες, που σχετίζονται με τη λειτουργία των Τ -κυττάρων, δηλαδή με το μηχανισμό άμυνας του οργανισμού.
Και άλλα αυτοάνοσα νοσήματα, όπως ο διαβήτης τύπου 1, συνδέονται επίσης με πολυμορφισμούς σε αυτά τα γονίδια.
Το συμπέρασμα είναι για την εμφάνιση των αυτοάνοσων διαταραχών του θυρεοειδούς δεν ευθύνονται μόνο τα γονίδια. Υπάρχει όμως αυξημένη γενετική προδιάθεση όταν κάποιος συγγενής, ακόμη και όχι πρώτου βαθμού, πάσχει από τη νόσο.
Επιπλέον η θυρεοειδίτιδα Hashimoto είναι 5-8 φορές πιο συχνή στις γυναίκες. Ως εκ τούτου, όλοι αυτοί που έχουν Hashimoto ή / και υποθυρεοειδισμό θα πρέπει να ενθαρρύνουν τις κόρες, τις αδελφές, τις μητέρες, τις θείες και τις γιαγιάδες να εξεταστούν, ειδικά αν βρίσκονται στην εφηβεία, την περίοδο της εγκυμοσύνης και την περίοδο της περιεμμηνόπαυσης, τις τρεις πιο «επικίνδυνες» ηλικίες για έρθουν οι ανωμαλίες του θυρεοειδούς στην επιφάνεια.
Το γεγονός ότι η κατάσταση είναι πιο συχνή στις γυναίκες, δε σημαίνει βέβαια ότι οι άνδρες δεν επηρεάζονται και δεν πρέπει να εξετάζονται!
Ο εντοπισμός ενός κληρονομικού συνδρόμου επιτρέπει την παρέμβαση πριν από την εμφάνιση του καρκίνου. Σε πολλές περιπτώσεις ασθενείς με οικογενή μεταλλάξη μπορούν να υποβληθούν σε θυρεοειδεκτομή σε πολύ νεαρή ηλικία. Μια τέτοια μετάλλαξη μπορεί να ανιχνευθεί με ένα απλό τεστ αίματος. Η ύπαρξη ενός συγγενή πρώτου βαθμού με θυρεοειδίτιδα Hashimoto αυξάνει τον κίνδυνο εμφάνισης της νόσου.
Σε κάθε περίπτωση χρειάζεται γνώση του ιατρικού ιστορικού και ευαισθητοποίηση. Τα κλινικά συμπτώματα θα πρέπει γίνονται γρήγορα αντιληπτά και να μην αγνοούνται, ώστε να υπάρχει έγκαιρη διάγνωση των παθήσεων του θυρεοειδούς.

6.4 Οστεοπόρωση

Η πάθηση αυτή απειλεί έτσι κι αλλιώς τις μισές από τις 60χρονες Ελληνίδες και το 12% των συνομηλίκων τους ανδρών. Οι ειδικοί ξέρουν ότι σε αυτό το πρόβλημα παίζει σημαντικό ρόλο η κληρονομικότητα.
Η εξασθένηση των οστών που χαρακτηρίζει την οστεοπόρωση οδηγεί στην απότομη ή προοδευτική παραμόρφωση των οστών και σε κατάγματα. Η οστεοπόρωση είναι σήμερα πολύ συχνή, λόγω της αύξησης του μέσου όρου ζωής αλλά και των κακών συνηθειών που έχουν επικρατήσει από την αστικοποίηση του πληθυσμού. Ιδιαίτερα συχνή είναι στις γυναίκες μετά την έλευση της εμμηνόπαυσης. Έχει διαπιστωθεί ότι όταν η μητέρα ή ο πατέρας πάσχει από οστεοπόρωση, τα παιδιά έχουν πολλές πιθανότητες να παρουσιάσουν και αυτά την ίδια πάθηση. Οι γυναίκες με οικογενειακό ιστορικό θα πρέπει να φροντίζουν από νωρίς να διατηρήσουν τη μέγιστη οστική πυκνότητα. Οι ειδικοί προειδοποιούν ότι οι γυναίκες κινδυνεύουν να χάσουν μέχρι και το 20% της οστικής τους μάζας σε διάστημα 5-7 ετών από τη στιγμή που θα μπουν στην εμμηνόπαυση.
Η οστεοπόρωση είναι μία πολύ σοβαρή πάθηση, η οποία απαιτεί προληπτικό έλεγχο για την έγκαιρη διάγνωση και αντιμετώπισή της, διότι προκαλεί συμπτώματα και παραμορφώσεις που δημιουργούν πολλά προβλήματα στους ασθενείς.
Τα συμπτώματα και οι παραμορφώσεις προέρχονται κυρίως από τη σπονδυλική στήλη και είναι η απώλεια αναστήματος, η κύφωση, η προβολή της κοιλιάς, οι συγκάμψεις των άκρων και ο μόνιμος πόνος στην πλάτη και τη μέση.

Όσο νωρίτερα γίνεται η ανίχνευση της οστεοπόρωσης τόσο περιορίζονται και οι συνέπειές της. Με βάση αυτό το σκεπτικό συνιστάται προληπτικός έλεγχος για οστεοπόρωση τουλάχιστον από την εποχή έναρξης της εμμηνόπαυσης.
Στην απλή διερεύνηση ενός γενικότερου πληθυσμού, όπως είναι οι γυναίκες που μπαίνουν στην εμμηνόπαυση, αρκεί μία μέτρηση της οστικής μάζας με την τεχνική της διπλής απορροφησιομετρίας (DXA) για να διαγνωστεί η οστεοπόρωση. Η μέτρηση πρέπει να γίνεται σε δύο τουλάχιστον σημεία (σπονδυλική στήλη και ισχίο) τουλάχιστον την πρώτη φορά που εξετάζεται μία γυναίκα.
Η μέτρηση αυτή είναι μία απλή εξέταση εκτίμησης της περιεκτικότητας σε άλατα ασβεστίου του σκελετού μας και με αυτήν αξιολογείται η κατάσταση των οστών και ειδικότερα ο κίνδυνος εμφάνισης οστεοπορωτικών καταγμάτων.
Η γενική συμβουλή είναι να υποβάλλονται σε μέτρηση της οστικής μάζας όλες οι γυναίκες λίγο πριν από την αναμενόμενη εμμηνόπαυση (λ.χ. στο 45ο έτος της ηλικίας τους).
Εδώ θα πρέπει να σημειωθεί πως το οικογενειακό ιστορικό οστεοπόρωσης μπορεί να οδηγήσει στην απόφαση για έλεγχο σε νεότερη ηλικία (ακόμα και πριν από το 40ό έτος της ηλικίας). Υπάρχει, όπως προαναφέρθηκε, ποσοστό περίπου 15% των νέων γυναικών που παρ' όλη τη φυσιολογική ορμονική κατάσταση, διαπιστώνεται στη μέτρηση ότι έχουν χαμηλή οστική μάζα.
Βασική αιτία για την κατάσταση αυτή (η οποία δεν πρέπει να θεωρείται απώλεια οστικής μάζας σε νεαρή ηλικία αλλά ελαττωματική ανάπτυξη της οστικής μάζας), είναι το κληρονομικό ιστορικό οστεοπόρωσης. Πράγματι υπάρχουν πολλές παραλλαγές (πολυμορφισμοί) γονιδίων που είναι υπεύθυνα για τη λειτουργία των οστών, οι οποίες έχουν κατηγορηθεί για την κατάσταση αυτή.
Επειδή ο έλεγχος των γονιδίων είναι μία πολύπλοκη και ίσως δαπανηρή διαδικασία, προτείνεται ένας απλούστερος τρόπος ανίχνευσης της κληρονομικής προδιάθεσης για οστεοπόρωση.
Η ύπαρξη οστεοπόρωσης (ή καλύτερα καταγμάτων που να οφείλονται σε οστεοπόρωση) σε έναν από τους γονείς ή σε μία αδελφή κοκ αυξάνει κατά περίπου 10 φορές την πιθανότητα να βρεθεί σε μία μέτρηση χαμηλή οστική μάζα. Αυτό ισχύει ιδιαίτερα όταν η φυσιογνωμία του ανθρώπου (φαινότυπος) είναι πανομοιότυπη με το νεαρής ηλικίας άτομο που ενδιαφέρεται να μάθει αν θα πάσχει και το ίδιο στο μέλλον από οστεοπόρωση.
Ο κίνδυνος ένα νέο άτομο να πάσχει είναι περισσότερο αυξημένος εάν συνυπάρχουν οι ακόλουθες αιτίες:

1) Υπάρχει χαμηλό σωματικό βάρος
2) Η ημερήσια πρόσληψη ασβεστίου είναι μικρότερη των 300 mg
3) Καπνίζει (περισσότερα από 10 τσιγάρα καθημερινά)

6.5 Μεσογειακή Αναιμία

Ο όρος θαλασσαιμία ή μεσογειακή αναιμία (γνωστή παλαιότερα ως αναιμία του Cooley) αναφέρεται σε μια ομάδα γενετικών αιματολογικών διαταραχών που επηρεάζουν την παραγωγή αιμοσφαιρίνης στον οργανισμό, ένα βασικό συστατικό του αίματος, της πρωτεΐνης δλδ που μεταφέρει το οξυγόνο από τους πνεύμονες προς τους ιστούς και τα διάφορα όργανα του σώματος.
Πρόκειται για μια από τις πιο κοινές μονογονιδιακές διαταραχές στον κόσμο, που πλήττει 15 εκατομμύρια άτομα σε όλον τον κόσμο.
Η θαλασσαιμία εμφανίζεται συχνότερα σε άτομα των Μεσογειακών χωρών (Ιταλία, Ελλάδα, Μέση Ανατολή), αλλά και της Νότιας Ασίας και της Αφρικής. Λόγω της συνεχούς μετανάστευσης

πληθυσμών από τη μία χώρα στην άλλη, σήμερα δεν υπάρχει ουσιαστικά καμία χώρα στον κόσμο χωρίς θαλασσαιμία.
Ορισμένα άτομα είναι φορείς του γονιδίου της θαλασσαιμίας, αλλά δεν εμφανίζουν κάποιο σημαντικό πρόβλημα υγείας που να σχετίζεται με τη διαταραχή αυτή, πλην μιας ελαφράς αναιμίας.
Τα άτομα αυτά έχουν το αποκαλούμενο στίγμα μεσογειακής αναιμίας (ελάσσων β θαλασσαιμία).
Η θαλασσαιμία είναι ένα κληρονομικό χαρακτηριστικό του αίματος. Ελαττώνει την ποσότητα αιμοσφαιρίνης που μπορεί το σώμα σας να πάρει έτσι ώστε οδηγεί στην αναιμία.
Παρακάτω ακολουθεί μία σύντομη ανάλυση των περιπτώσεων που ένα ζευγάρι φέρνει στο κόσμο ένα παιδί.

- Δύο «φυσιολογικοί» γονείς.

Με κανένα τρόπο δεν μπορούν να μεταδώσουν το στίγμα της θαλασσαιμίας ή τη μείζονα θαλασσαιμία στα παιδιά τους.

- Ένας από τους δύο γονείς έχει το στίγμα της θαλασσαιμίας και ο άλλος είναι «φυσιολογικός».

Όλα τα παιδιά θα πρέπει να κληρονομήσουν ένα φυσιολογικό γονίδιο ή ένα γονίδιο θαλασσαιμίας από ένα φορέα-γονιό. Για το κάθε παιδί υπάρχει μια στις δύο (50%)πιθανότητα κληρονόμησης του γονιδίου θαλασσαιμίας από τον γονιό-φορέα: αν συμβεί αυτό τότε το παιδί θα φέρει το στίγμα της θαλασσαιμίας. Επίσης, υπάρχει μια στις δύο (50%) πιθανότητες κληρονόμησης του φυσιολογικού γονιδίου από τον γονιό-φορέα αν συμβεί αυτό τότε το παιδί θα είναι εντελώς φυσιολογικό.

- Και οι δύο γονείς είναι φορείς του στίγματος β-θαλασσαιμίας είναι δηλ. «ένα ζευγάρι σε κίνδυνο».

Τα ζευγάρια φορέων του στίγματος έχουν 1 πιθανότητα στις 4(25%) σε κάθε εγκυμοσύνη να κάνουν παιδί που θα είναι ασθενής της Μείζονος θαλασσαιμίας:1 πιθανότητα στις 2(50%) να κάνουν παιδί φορέα του στίγματος θαλασσαιμίας και 1 πιθανότητα στις 4(25%) ότι το παιδί που θα κάνουν θα κληρονομήσει ένα φυσιολογικό γονίδιο από τον κάθε γονιό και ότι έτσι θα είναι εντελώς φυσιολογικό.

7 ΑΝΑΠΤΥΞΗ ΠΛΗΡΟΦΟΡΙΑΚΟΥ ΣΥΣΤΗΜΑΤΟΣ

7.1 Εισαγωγή

Ως διεργασία ανάπτυξης λογισμικού ονομάζεται το σύνολο δραστηριοτήτων που αποβλέπουν στην ανάπτυξη ενός προϊόντος λογισμικού. Όλες οι διεργασίες ανάπτυξης λογισμικού περιλαμβάνουν τις ακόλουθες δραστηριότητες:

- ανάλυση και καθορισμός απαιτήσεων
- σχεδιασμός
- υλοποίηση
- επαλήθευση και επικύρωση
- εξέλιξη

Στην πρώτη φάση έχουμε την ανάλυση απαιτήσεων που είναι μια διαδικασία κατάρτισης μιας λίστας, όπου αναφέρονται οι προδιαγραφές που πρέπει να πληροί το προϊόν (η λύση) πληροφορικής που πρόκειται να δημιουργηθεί ή να εγκατασταθεί. Το προϊόν μπορεί να είναι μια νέα εφαρμογή που αναπτύσσεται κατά παραγγελία, η εγκατάσταση ενός έτοιμου προγράμματος, η δημιουργία ενός δικτυακού τόπου, ηλεκτρονικού καταστήματος κ.λπ. Οι προδιαγραφές που προσδιορίζονται μπορεί να είναι τεχνολογικές, επιχειρηματικές, λειτουργικές, να σχετίζονται με τη μορφή, το κόστος, τη διάρκεια, το χρόνο απόσβεσης κ.ά. Η λίστα που καταρτίζεται χρησιμεύει τόσο σε αυτούς που θα εγκαταστήσουν ή θα αναπτύξουν τη λύση λογισμικού όσο και σ' εκείνους που θα τη χρησιμοποιήσουν (χρήστες, πελάτες) και εφαρμόζεται σε μεγάλα αλλά και μικρά έργα πληροφορικής. Πραγματοποιείται μελέτη σκοπιμότητας, γίνεται εκτίμηση αν οι ανάγκες των χρηστών μπορούν να ικανοποιηθούν με τρέχουσες τεχνολογίες, αν είναι συμφέρον από επιχειρηματική άποψη, αν το απαιτούμενο κόστος είναι εντός προϋπολογισμού.

Στο στάδιο του σχεδιασμού, ορίζεται το πώς οι παραπάνω απαιτήσεις μπορούν να πραγματοποιηθούν. Καθορίζονται με βάση τον προσδιορισμό των απαιτήσεων ποιες θα είναι οι λειτουργίες του συστήματος, οι διεπαφές του και οι περιορισμοί που θα επιλεχθούν, ώστε να γίνει η ανάπτυξη.

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 που τέθηκαν στο σχεδιασμό.

Στη φάση της επαλήθευσης και της επικύρωσης, το σύστημα με την παρούσα δομή του εκτελείται και τα δεδομένα δοκιμών που παράγονται ελέγχονται ώστε να πραγματοποιηθεί επαλήθευση για αν τελικά αναπτύχθηκε σωστά το λογισμικό με βάση πάντα αυτά που έχουν καθοριστεί στην ανάλυση των προδιαγραφών και το σχεδιασμό υλοποίησης. Επιπλέον γίνεται και επικύρωση ότι όντως έχει αναπτυχθεί λογισμικό σύμφωνα με τις απαιτήσεις του χρήστη και του πελάτη.

Τέλος έχουμε και το στάδιο της εξέλιξης που έχει στόχο την συνεργασία με τον πελάτη και τον χρήστη προκειμένου να υλοποιηθούν οι τυχόν διορθώσεις που μπορεί να απαιτούνται και να ενσωματωθούν οι νέες δυνατότητες που προτείνονται ώστε να εξασφαλίζεται η πλήρης λειτουργικότητα του τελικού προϊόντος.

7.2 Ανάλυση Προδιαγραφών - Παραδοχές

Στο συγκεκριμένο σύστημα ορίζονται οι παρακάτω απαιτήσεις:

- Οι χρήστες του συστήματος θα είναι οι γιατροί ενός εξωτερικού ιδιωτικού ιατρείου.

- Αρχικά ο χρήστης θα πρέπει να δημιουργήσει λογαριασμό δηλώνοντας ένα έγκυρο mail ως username και ένα password. Στην συνέχεια θα μπορεί να πραγματοποιήσει είσοδο, κάνοντας χρήση των σωστών διαπιστευτηρίων του.
- Οι χρήστες του συστήματος θα μπορούν να καταχωρήσουν ασθενείς τα οποία τους επισκέπτονται για εξέταση, δημιουργώντας νέα εγγραφή η οποία αποτελεί τον ιατρικό του φάκελο.
- Οι χρήστες του συστήματος θα μπορούν να διαχειρίζονται τον ιατρικό φάκελο ενός ασθενούς. Ο ιατρικός φάκελος θα περιλαμβάνει όλα τα δημογραφικά δεδομένα του ασθενούς, την φωτογραφία του, όπως επίσης τις αλλεργίες του, τις φαρμακευτικές του αγωγές, τα νοσήματα και τις χειρουργικές επεμβάσεις του, το οικογενειακό του ιστορικό. Τέλος, στον ιατρικό φάκελο ενός ασθενούς θα υπάρχει η δυνατότητα προσθήκης αρχείων από ιατρικές εξετάσεις που έχει προσκομίσει ο ασθενής.
- Θα υπάρχει και η δυνατότητα για κάθε ασθενή κατά την επεξεργασία της καρτέλας που αφορά το οικογενειακό του ιστορικό, να δηλώσει ως συγγενείς πρώτου βαθμού, άτομα τα οποία έχουν ήδη επισκεφτεί και καταχωρηθεί από τον ίδιο γιατρό.
- Για κάθε ασθενή μπορούν να προστεθούν επισκέψεις, οι οποίες έχουν καρτέλες με πληροφορίες αναφορικά με την παρούσα νόσο και τα συμπτώματα που παρουσιάζει ο ασθενής, που ουσιαστικά είναι και η αιτία της επίσκεψής του.
- Στην συνέχεια καταχωρείτε από τον γιατρό η διαφορική και η τελική διάγνωση για κάθε επίσκεψη.
- Κάθε επίσκεψη θα χαρακτηρίζεται από τον ασθενή, τον γιατρό, την ημερομηνία και ώρα της επίσκεψης, τον λόγο επίσκεψης και την διάγνωση που προέκυψε.
- Το σύστημα για κάθε επίσκεψη θα δίνει την δυνατότητα στον χρήστη να εξάγει σε εκτυπώσιμη μορφή το Φύλλο Επισκέψεως που έχει δημιουργηθεί για κάποιον ασθενή.
- Το σύστημα θα παρέχει στατιστικά δεδομένα για το σύνολο των ασθενών που έχουν καταγραφεί από τον γιατρό αναφορικά με το φύλο και την ηλικία τους καθώς και σε συνδυασμό με τα δεδομένα που επικρατούν αυτήν της περίοδο στην ιατρική, θα παρουσιάζει την πιθανότητα εμφάνισης κάποιων ασθενειών στους ασθενής του με βάση το οικογενειακό τους ιστορικό.

7.3 Σχεδιασμός – Διαγράμματα UML

Η ενοποιημένη γλώσσα σχεδιασμού (unified modeling language) (UML) είναι μια γραφική γλώσσα για την οπτική παράσταση, τη διαμόρφωση προδιαγραφών και την τεκμηρίωση λογισμικών. Το σχέδιο είναι μια απλοποιημένη παράσταση της πραγματικότητας. 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.

Σχεδιάζουμε για να μπορέσουμε να καταλάβουμε το σύστημα που αναπτύσσουμε. Έτσι δημιουργώντας ένα σχέδιο επιτυγχάνουμε τέσσερεις στόχους:

- παριστάνουμε οπτικά το σύστημα που έχουμε ή θέλουμε να κατασκευάσουμε,
- προσδιορίζουμε τη δομή και τη συμπεριφορά του συστήματος,
- δημιουργούμε ένα πρότυπο για να βασίσουμε την κατασκευή του συστήματος,
- τεκμηριώνουμε τις αποφάσεις που λάβαμε.

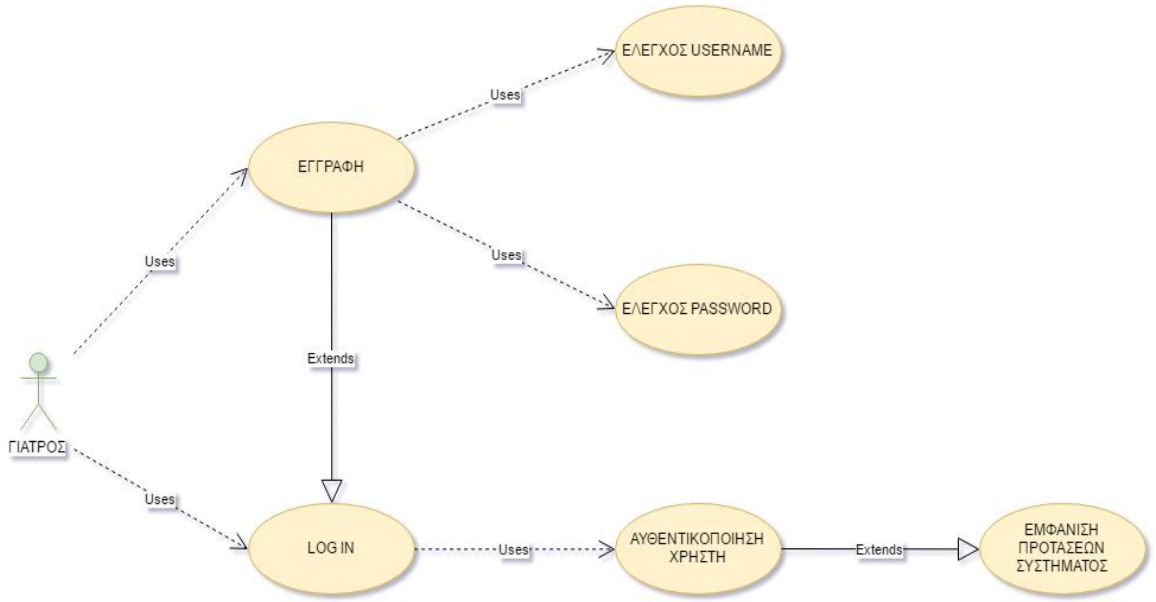
Η UML είναι και ορίζει μόνο τη γλώσσα μοντελοποίησης και όχι τη διαδικασία, ούτε την μεθοδολογία. Η UML περιλαμβάνει τρία βασικά στοιχεία:

- Οντότητες
- Σχέσεις
- Διαγράμματα

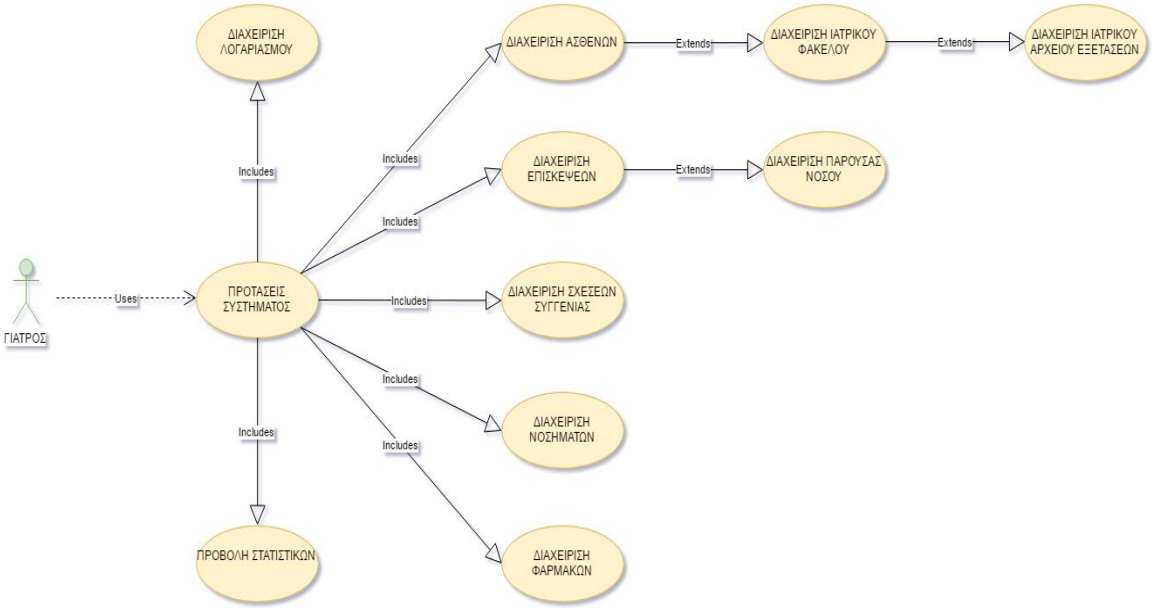
Για την αναπαράσταση των διαγραμμάτων που ακολουθεί, χρησιμοποιήθηκε το Drawio της Google.

7.3.1 Διαγράμματα Περιπτώσεων Χρήσης

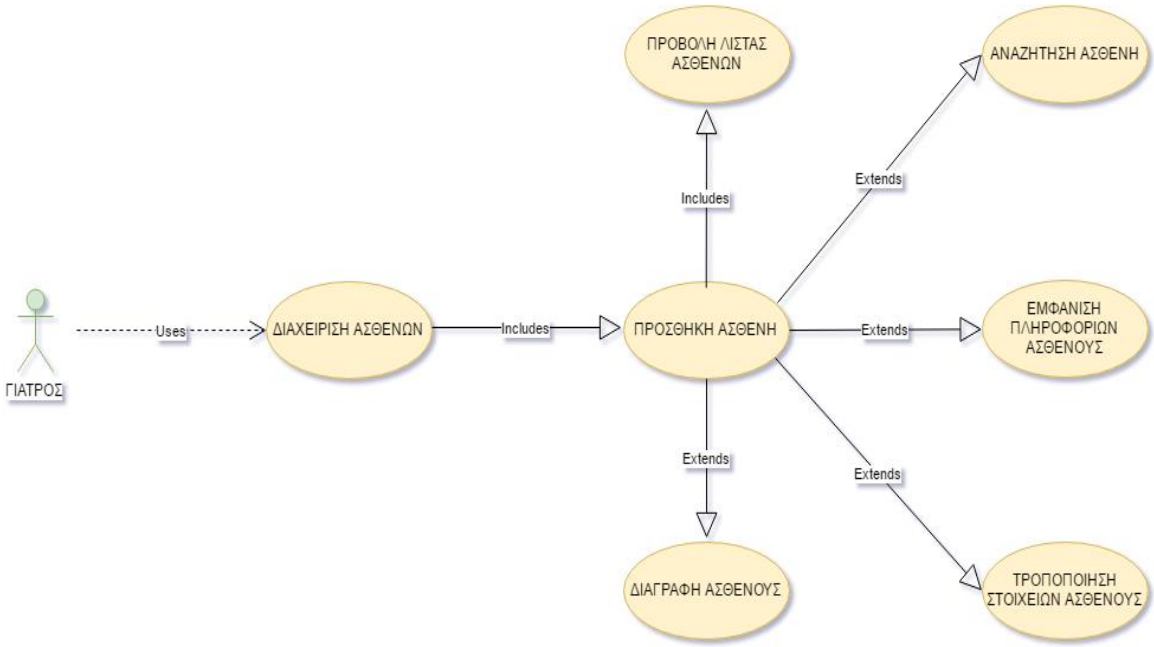
Τα διαγράμματα περιπτώσεων χρήσης χρησιμοποιούνται για τη μοντελοποίηση του πλαισίου λειτουργίας της εφαρμογής και την καταγραφή των προδιαγραφών της. Περιλαμβάνουν τις λειτουργίες/περιπτώσεις χρήσης, τους χρήστες του συστήματος (actors) καθώς επίσης και πιθανές σχέσεις γενίκευσης. Ουσιαστικά έχουμε μία συγκεντρωτική εμφάνιση των απαιτήσεων ενός συστήματος, και προσδιορίζεται μία εποπτική εικόνα του συστήματος.



Στο παραπάνω Use Case διάγραμμα φαίνεται η διαδικασία που ο γιατρός δεν έχει λογαριασμό για να κάνει είσοδο και κάνει εγγραφή με έγκυρα διαπιστευτήρια και στη συνέχεια μπορεί να τα χρησιμοποιήσει για να συνδεθεί μετά από επιτυχημένη αυθεντικοποίηση.



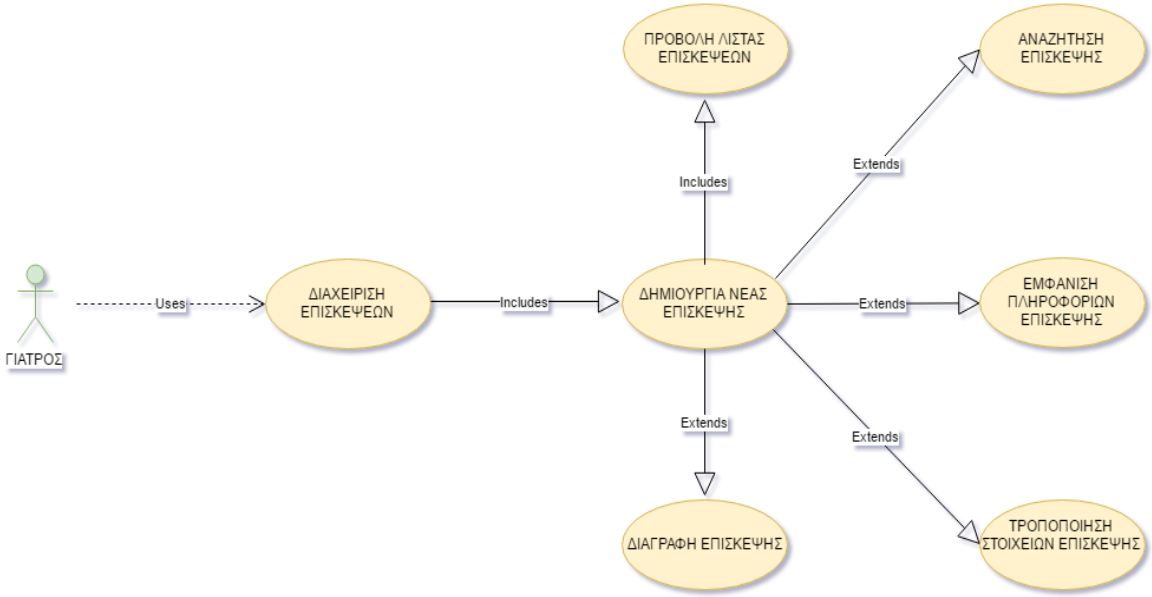
Στο παραπάνω Use Case διάγραμμα φαίνονται οι προτάσεις του συστήματος αφού πρώτα πραγματοποιήσει ο γιατρός επιτυχημένη σύνδεση στην εφαρμογή του.



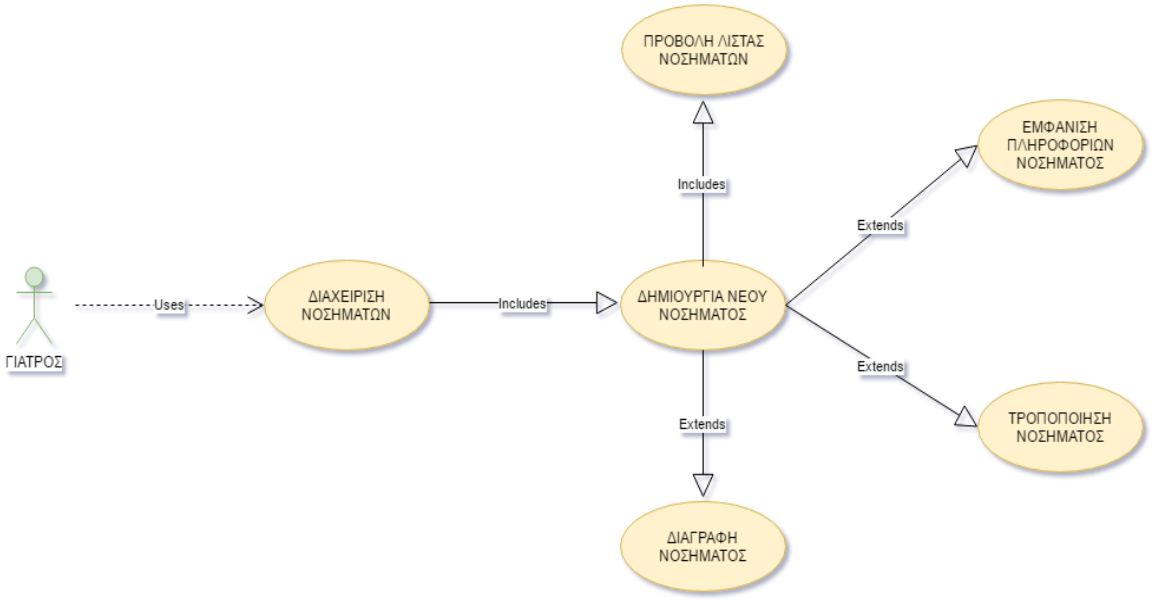
Στο παραπάνω Use Case διάγραμμα φαίνονται οι επιλογές που έχει ο γιατρός σχετικά με την διαχείριση των ασθενών που τον επισκέπτονται για εξέταση.

Στο παραπάνω Use Case διάγραμμα φαίνεται η δυνατότητα που δίνεται στον χειριστή να αλλάξει τον κωδικό πρόσβασής του.

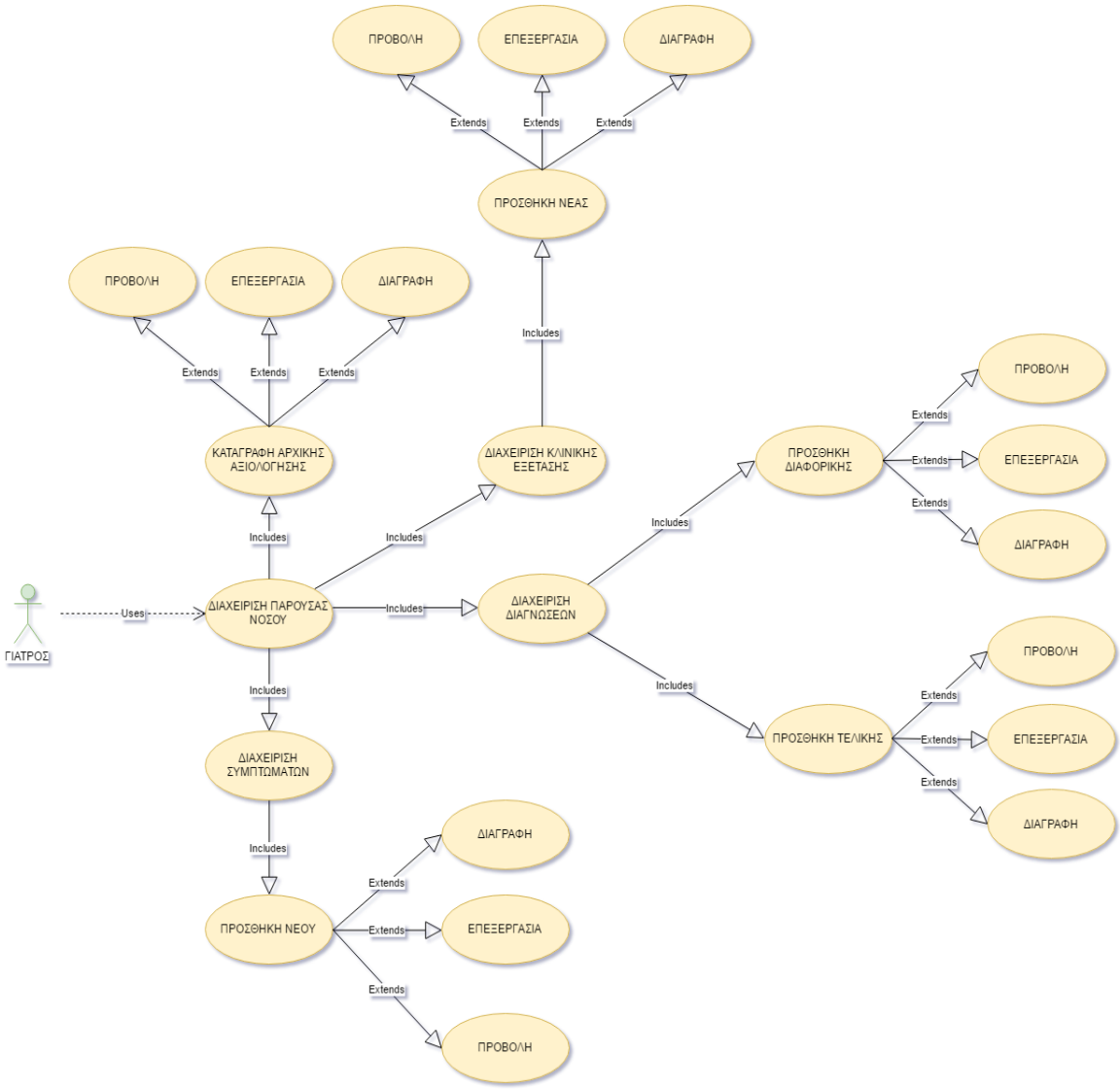
Στο παραπάνω Use Case διάγραμμα φαίνεται αναλυτικά η λειτουργία της διαχείρισης του ιατρικού φακέλου για κάθε ασθενή. Στην αναπαράσταση αναγνωρίζονται οι διαδικασίες για την καταγραφή των αλλεργιών, των φαρμάκων, του κοινωνικού και οικογενειακού ιστορικού καθώς και του ατομικού αναμνηστικού.



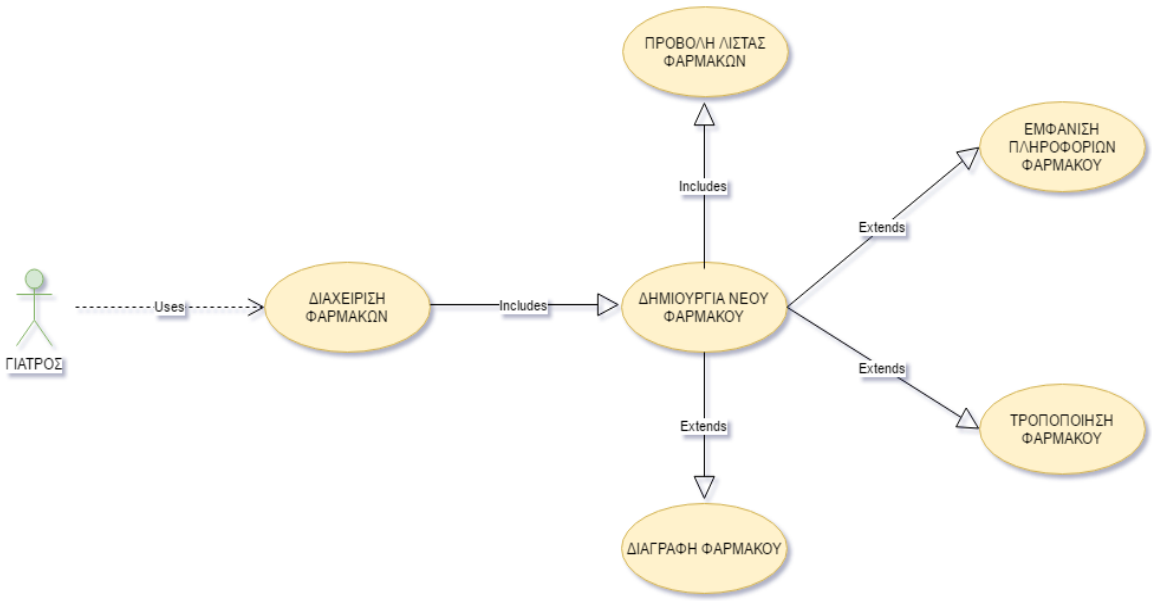
Στο παραπάνω Use Case διάγραμμα φαίνεται η λειτουργία που έχει ο γιατρός αναφορικά με την διαχείριση των πληροφοριών που καταγράφει για τις επισκέψεις που πραγματοποιεί ο κάθε ασθενής του.



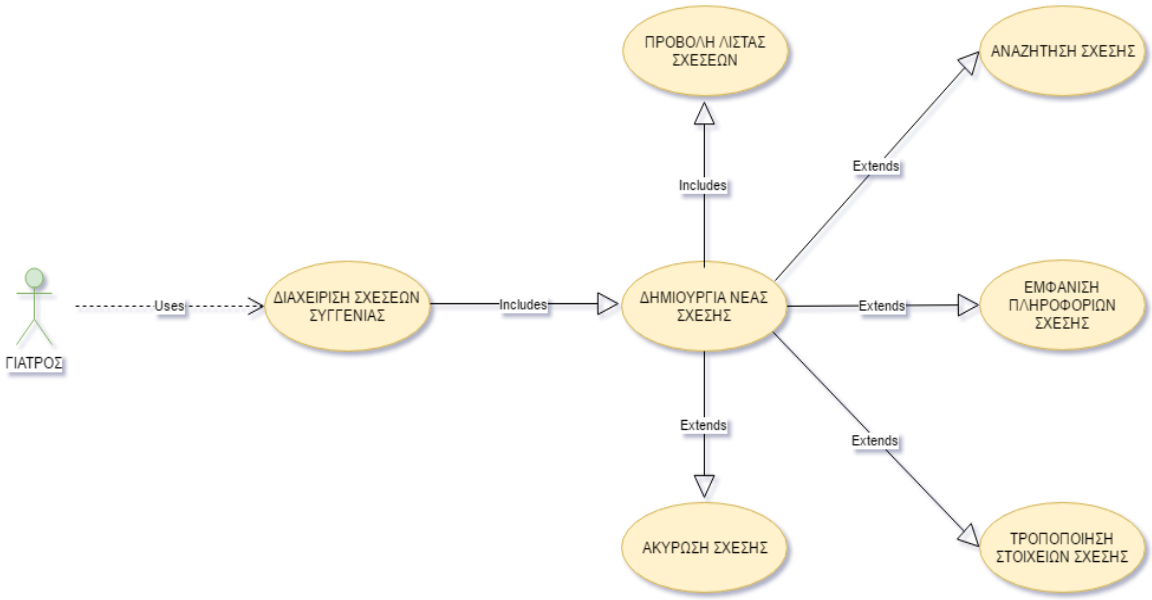
Στο παραπάνω Use Case διάγραμμα φαίνεται η λειτουργία διαχείρισης των νοσημάτων.



Στο παραπάνω Use Case διάγραμμα φαίνεται αναφορικά με την επίσκεψη η λειτουργίες καταγραφής των χαρακτηριστικών που πλαισιώνουν κάθε επίσκεψη.



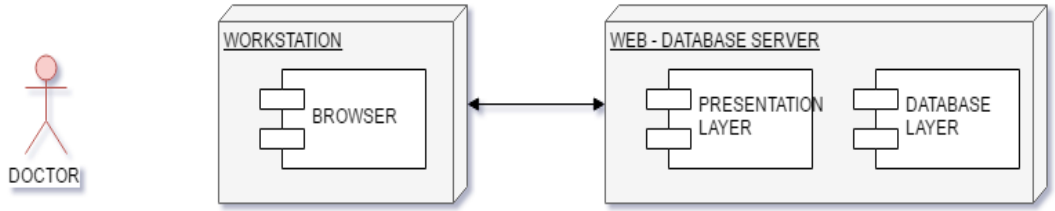
Στο παραπάνω Use Case διάγραμμα φαίνεται η επιλογή που δίνεται στο γιατρό να διαχειριστεί κατευθείαν τα φάρμακα τα οποία είναι στην φαρμακευτική αγωγή του ασθενή.



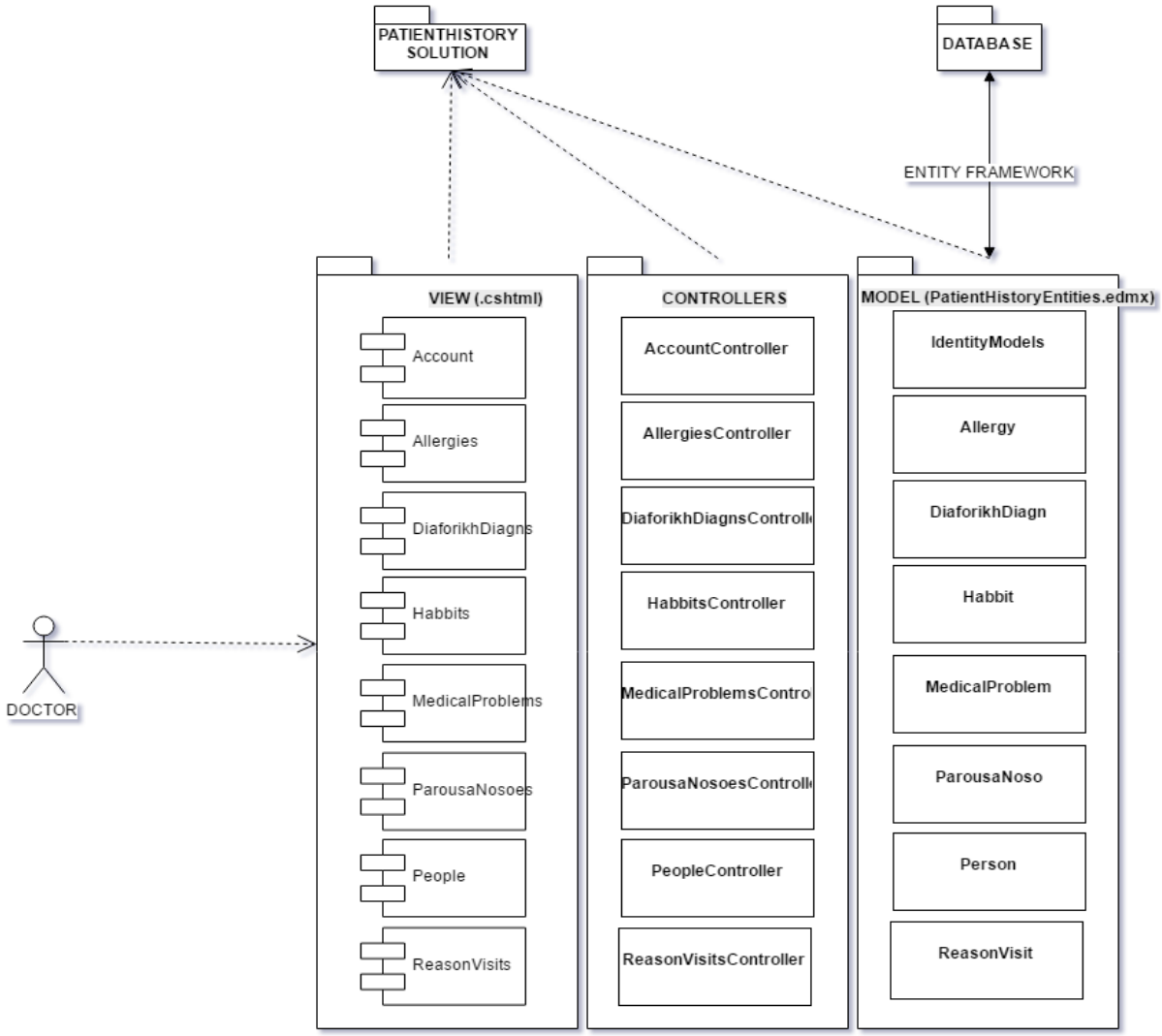
Στο παραπάνω Use Case διάγραμμα φαίνεται η λειτουργία της εφαρμογής να δημιουργεί σχέσεις συγγένειες μεταξύ των καταχωρημένων ασθενών.

7.3.2 Διαγράμματα Δομής Υλοποίησης

Τα διαγράμματα δομής υλοποίησης είναι τα διάγραμμα εξαρτημάτων και τα διάγραμμα ανάπτυξης. Το διάγραμμα εξαρτημάτων δείχνει τα συστατικά μέρη του κώδικα και την φυσική τους δομή. Παρουσιάζει την οργάνωση και τις εξαρτήσεις των μελών ενός συνόλου συνιστωσών. Το διάγραμμα ανάπτυξης περιγράφουν τον τρόπο με τον οποίο διατάσσεται το λογισμικό και το υλικό για την τελική λειτουργία του συστήματος.



Έτσι στο παραπάνω Deployment διάγραμμα, το πρόγραμμα περιήγησης είναι μια UI εφαρμογή που κατοικεί στον υπολογιστή του σταθμού εργασίας του χρήστη-γιατρού και επικοινωνεί με την βάση δεδομένων που είναι και αυτή εγκατεστημένη στην προκειμένη περίπτωση στον ίδιο μηχάνημα με τον web server που φιλοξενείτε το web υλικό.



Στο παραπάνω Component διάγραμμα φαίνονται τα συστατικά μέρη του κώδικα και η φυσική τους δομή.

7.4 Υλοποίηση ΠΣ

7.4.1 Εισαγωγή

Σε αυτήν την ενότητα θα γίνει περιγραφή της γλώσσας προγραμματισμού που χρησιμοποιήθηκε, στα εργαλεία, τις τεχνολογίες που εφαρμόστηκαν και ανάλυση στην δομή του κώδικα που αναπτύχθηκε.

7.4.2 Η γλώσσα προγραμματισμού c# και το περιβάλλον .NET

Η C# (CSharp) είναι μια αντικειμενοστραφής γλώσσα προγραμματισμού η οποία δημιουργήθηκε από την εταιρεία λογισμικού Microsoft στα πλαίσια της προώθησης του περιβάλλοντος .NET. Η C# αναπτύχθηκε ώστε να υποστηρίζει το περιβάλλον .NET.

Η χρήση της γλώσσας προγραμματισμού C# επελέγη κυρίως λόγω της εύκολης εκμάθησής της και της ευρείας χρήσης της σε εφαρμογές για το λειτουργικό σύστημα Windows. Η χρήση του περιβάλλοντος .NET απλοποίησε αρκετές από τις διαδικασίες ανάπτυξης της εφαρμογής και βοήθησε στην γρήγορη κατασκευή ενός γραφικού περιβάλλοντος.

Η έκδοση που χρησιμοποιήθηκε η ανάπτυξη της εφαρμογής είναι η .NET Framework Version 4.5.

7.4.3 Το περιβάλλον ανάπτυξης Microsoft Visual Studio

Το περιβάλλον δημιουργίας του συστήματος είναι το Microsoft Visual Studio Έκδοση Professional 2013 Version 12 Update 5. Το συγκεκριμένο εργαλείο είναι ένα ενοποιημένο περιβάλλον ανάπτυξης λογισμικού (IDE–Integrated Development Environment) που κατασκευάζει και διαθέτει η εταιρεία Microsoft. Χρησιμοποιείται για την ανάπτυξη κάθε είδους προγραμμάτων ηλεκτρονικών υπολογιστών Microsoft Windows, καθώς και ιστοσελίδων, διαδικτυακών εφαρμογών, υπηρεσιών Web, εφαρμογών κινητών συσκευών κ.α..

Το συγκεκριμένο project για τις ανάγκες της εργασίας το κάνω p=δημοσίευση με τον IIS Express που είναι ενσωματωμένος στον Visual Studio και του Url είναι σε localhost στην πόρτα 55430 (http://localhost:55430/).

7.4.4 ASP .net

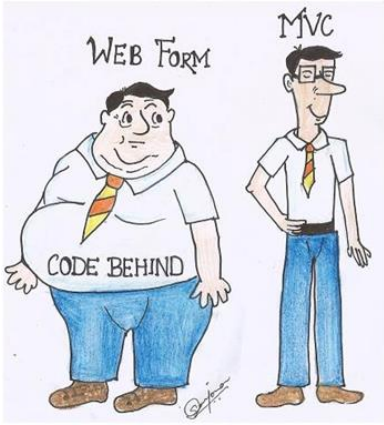
Το Asp.net είναι ένα προγραμματιστικό περιβάλλον της Microsoft που δημιουργήθηκε για διαδικτυακό προγραμματισμό για την δημιουργία δυναμικών ιστοσελίδων στο διαδίκτυο. Αναπτύχθηκε για να δώσει την δυνατότητα σε προγραμματιστές να δημιουργήσουν ιστοσελίδες, διαδικτυακές εφαρμογές και διαδικτυακές υπηρεσίες.

Με το ASP.NET, μπορούν να δημιουργηθούν γρήγορα web sites που βασίζονται σε HTML, CSS και JavaScript, προσθέτοντας στη συνέχεια και πιο σύνθετες δυνατότητες όπως APIs Web, Φόρμες πάνω σε δεδομένα ή και συναλλαγές σε πραγματικό χρόνο.

Το ASP.NET προσφέρει τρία πλαίσια τεχνολογιών για τη δημιουργία web εφαρμογών:

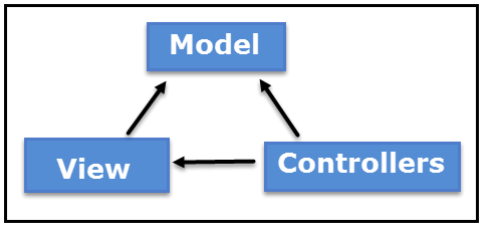
- Web Forms
- ASP.NET MVC
- ASP.NET Web Pages

Και τα τρία πλαίσια είναι σταθερά και ώριμα, και μπορούν να δημιουργήσουν μεγάλες εφαρμογές web. Δεν έχει σημασία τι πλαίσιο θα επιλεχθεί, καθώς όλα τα πλεονεκτήματα και τα χαρακτηριστικά του ASP.NET υπάρχουν παντού.
Το κάθε πλαίσιο στοχεύει σε ένα διαφορετικό στυλ ανάπτυξης. Η επιλογή εξαρτάται από ένα συνδυασμό των στοιχείων του προγραμματιστή (γνώσεις, δεξιότητες και εμπειρία ανάπτυξης) και τον τύπο της εφαρμογής που θα δημιουργηθεί. Και τα τρία πλαίσια όμως, θα πρέπει να υποστηριχθούν, να ενημερώνονται και να βελτιώνονται με τις μελλοντικές εκδόσεις του ASP.NET. Επιπλέον, τα τρία πλαίσια δεν είναι εντελώς ανεξάρτητα, και επιλέγοντας ένα δεν αποκλείεται η χρήση άλλου.



7.4.5 ASP .net MVC

Το ASP.NET MVC είναι ένα πλαίσιο για web εφαρμογή που αναπτύχθηκε από τη Microsoft, η οποία υλοποιεί το πρότυπο Model-View-Controller (MVC). Ισχυρό και ελαφρύ εργαλείο για την οικοδόμηση εξελιγμένων δυναμικών ιστοσελίδων που επιτρέπει σε έναn καθαρό διαχωρισμό των εργασιών και δίνει τον πλήρη έλεγχο σήμανσης για ευκίνητη και γρήγορη ανάπτυξη.



Βασισμένο σε ASP.NET, το MVC επιτρέπει στους προγραμματιστές λογισμικού να οικοδομήσουν μια web εφαρμογή, ως σύνθεση με τρεις ρόλους: Μοντέλο (Model), Προβολή (View) και Ελεγκτή (Controller). Κάθε ένα από αυτά τα συστατικά έχει κατασκευαστεί για να χειριστεί ειδικές αναπτυξιακές πτυχές της αίτησης.

- Ένα μοντέλο αναπαριστά την κατάσταση μιας συγκεκριμένης πτυχής της αίτησης και αντιστοιχεί σε όλα τα στοιχεία που σχετίζονται με τη λογική στην οποία εργάζεται ο χρήστης.
- Ένας ελεγκτής χειρίζεται τις αλληλεπιδράσεις και ενημερώνει το μοντέλο ώστε να αντικατοπτρίζει μια αλλαγή στην κατάσταση της αίτησης, και στη συνέχεια περνάει πληροφορίες στην προβολή.
- Μια προβολή δέχεται τις απαραίτητες πληροφορίες από τον ελεγκτή και καθιστά μια διεπαφή χρήστη για να εμφανίσει τις πληροφορίες.

7.4.6 Πλεονεκτήματα MVC-based Web εφαρμογής

Το πλαίσιο ASP.NET MVC προσφέρει τα εξής πλεονεκτήματα:

- Θα είναι πιο εύκολο να διαχειριστεί την πολυπλοκότητα διαχωρίζοντας τα καθήκοντα μιας εφαρμογής στο μοντέλο, την προβολή και τον ελεγκτή καθώς είναι ιδανικό για έργα με μεγάλη ομάδα web developers επιτρέποντας να εργαστούν μεμονωμένα.
- Δεν χρησιμοποιεί προβολές που βασίζονται σε διακομιστή. Αυτό κάνει το πλαίσιο MVC ιδανικό για τους προγραμματιστές που θέλουν πλήρη έλεγχο της συμπεριφοράς της αίτησης.
- Χρησιμοποιεί ένα μοτίβο πρόσοψης του ελεγκτή που επεξεργάζεται αιτήσεις εφαρμογών Web μέσω ενός ενιαίου ελεγκτή. Αυτό δίνει τη δυνατότητα να σχεδιαστεί μια εφαρμογή που υποστηρίζει μια πλούσια υποδομή δρομολόγησης.

- Παρέχει την καλύτερη υποστήριξη για την ανάπτυξη οδηγούμενη από τον έλεγχο (TDD).

Λειτουργεί καλά για Web εφαρμογές που υποστηρίζονται από τις μεγάλες ομάδες προγραμματιστών και για το Web σχεδιαστές που χρειάζονται ένα υψηλό βαθμό ελέγχου επί της συμπεριφοράς της εφαρμογής.

7.4.7 Entity Framework

Το Entity Framework (EF) είναι ένα ανοικτού κώδικα object-relational mapper (αντικειμενοστραφες) framework για ADO.NET, το οποίο ήταν ένα μέρος του .NET Framework, αλλά από την νέα έκδοση 6 διαχωρίζεται από το .NET Framework. Αυτό το σύνολο τεχνολογιών στο ADO.NET επιτρέπει στους .NET προγραμματιστές να συνεργαστούν με τη σχεσιακή βάση δεδομένων κάνοντας χρήση αντικειμένων που έχουν δημιουργήσει υποστηρίζοντας την ανάπτυξη εφαρμογών λογισμικού δεδομένων. Με αυτό τον τρόπο, εξαλείφεται η ανάγκη για το μεγαλύτερο μέρος του κωδικού πρόσβασης στα δεδομένα της βάσης που οι προγραμματιστές συνήθως χρειάζονται να γράψουν. Ένα τυπικό ORM εργαλείο όπως είναι το Entity, παράγει κλάσεις για την διασύνδεση με την Βάση Δεδομένων μίας εφαρμογής.

Το Entity Framework επιτρέπει τη δημιουργία ενός μοντέλου με το γράψιμο κώδικα ή χρησιμοποιώντας κουτιά και τις γραμμές στο Designer EF. Και οι δύο αυτές προσεγγίσεις μπορούν να χρησιμοποιηθούν για τη στόχευση σε μια υπάρχουσα βάση δεδομένων ή να δημιουργηθεί μια νέα από την αρχή. Με την μέθοδο Code First που η ανάπτυξη στοχεύει στη συνεργασία με τη βάση δεδομένων, ορίζονται αρχικά τα μοντέλα μας, χρησιμοποιώντας κλάσεις με τη μορφή των αντικειμένων και των ιδιοτήτων σε γλώσσα C#, χωρίς να χρειάζεται να ασχοληθούμε με τους πίνακες της βάσης δεδομένων και τις στήλες, όπου αποθηκεύονται τα δεδομένα. Τα Code-First APIs θα δημιουργήσουν τη βάση δεδομένων on the fly (αν δεν υπάρχει ήδη), κάνοντας τις απαραίτητες αντιστοιχίσεις.

Τέλος το Entity Framework είναι χρήσιμο και για την περίπτωση που υπάρχει ήδη το σχήμα της Βάσης Δεδομένων.

Με το πλαίσιο οντοτήτων, οι προγραμματιστές μπορούν να εργαστούν σε ένα υψηλότερο επίπεδο αφαίρεσης, όταν ασχολούνται με τα δεδομένα, και μπορούν να δημιουργήσουν και να διατηρήσουν τα δεδομένα με γνώμονα τις εφαρμογές τους, έχοντας λιγότερο κώδικα σε σχέση με τις παραδοσιακές εφαρμογές.

Στην περίπτωση της συγκεκριμένης εφαρμογής σχεδιάστηκε πρώτα το σχήμα της Βάσης Δεδομένων στον SQL Server και στην συνέχεια χρησιμοποιώντας την διαθέσιμη τεχνολογία δημιουργήσαμε στο project μας τις οντότητες.

Το κύριο συστατικό του Entity Framework, είναι το Entity Data Model (EDM), το οποίο αποτελείται από 3 μέρη. Αρχικά το Conceptual Model, που περιλαμβάνει τις κλάσεις μοντελοποίησης και τις μεταξύ τους σχέσεις. Το Storage Model που αποτελεί το μοντέλο σχεδίασης της Βάσης Δεδομένων με τα αντικείμενά της. Και τέλος το Mapping στο οποίο περιέχονται οι πληροφορίες απεικόνισης του Storage Model στο Conceptual Model.

Στο project της εφαρμογής μας, υπάρχει το PatientHistoryEntities.edmx με κώδικα που δημιουργήθηκε αρχικαάυτόματα από την υπάρχουσα βάση δεδομένων μας και στην συνέχεια παραμετροποιήθηκαν διάφορα αρχεία του με τα σημαντικότερα το PatientHistoryEntities.Context.cs και το PatientHistoryEntities.tt που περιέχει το σύνολο των Οντοτήτων που αντιστοιχίζονται σε πίνακες της Βάσης Δεδομένων. Η DBContext κλάση είναι η κύρια κλάση, η οποία ευθύνεται για την διαχείριση των δεδομένων, ως αντικείμενα.

7.4.8 Δομή Κώδικα

Το project που δημιούργησα στο Visual Studio είναι το PatientHistory το οποίο περιλαμβάνει όλα τα αρχεία πηγαίου κώδικα, εικόνες, αρχεία δεδομένων. Είναι ένα Asp .Net Web Application project με δομή MVC, δηλαδή γίνεται χρήση της αρχιτεκτονικής Model-View-Controller. Υπάρχει ένα solution που έχει όλες τις πληροφορίες κατάστασης.

Τα αρχεία του κώδικα είναι ομαδοποιημένα μέσα στο solution σε φακέλους, με πιο κύριους τους Models ,View, Controllers που περιέχουν τις οντότητες που συνδέονται με την βάση δεδομένων του συστήματος, την λογική της λειτουργικότητας που εφαρμόζουμε σύμφωνα με τις απαιτήσεις μας και τέλος το περιεχόμενο που εμφανίζεται στον χρήστη.

Στις επόμενες ενότητες θα γίνει αναλυτική εξέταση της δομής, των τεχνολογιών και των μεθοδολογιών του κώδικα που χρησιμοποιήθηκαν.

7.4.9 Bootstrap

Το Bootstrap είναι μια συλλογή εργαλείων ανοιχτού κώδικα (Ελεύθερο λογισμικό) για τη δημιουργία ιστοσελίδων και διαδικτυακών εφαρμογών. Περιέχει HTML και CSS για τις μορφές τυπογραφίας, κουμπιά πλοήγησης και άλλων στοιχείων του περιβάλλοντος, καθώς και προαιρετικές επεκτάσεις JavaScript. Είναι το πιο δημοφιλές πρόγραμμα στο GitHub [1] και έχει χρησιμοποιηθεί από τη NASA και το MSNBC, μεταξύ άλλων.

Το Bootstrap αναπτύχθηκε από τον Mark Otto και τον Jacob Thornton στο Twitter ως ένα πλαίσιο για την ενθάρρυνση της συνέπειας στα εσωτερικά εργαλεία. Πριν το Bootstrap, διάφορες βιβλιοθήκες χρησιμοποιήθηκαν για την ανάπτυξη της διεπαφής, η οποία οδήγησε σε αντιφάσεις και υψηλή φορολογική επιβάρυνση συντήρησης.

Το Bootstrap έχει σχετικά ελλιπή υποστήριξη για HTML5 και CSS αλλά είναι συμβατό με όλους τους φυλλομετρητές (browsers). Βασικές πληροφορίες συμβατότητας των ιστοσελίδων ή εφαρμογές είναι διαθέσιμες για όλες τις συσκευές και τα προγράμματα περιήγησης. Υπάρχει μια έννοια της μερικής συμβατότητας που κάνει τα βασικά στοιχεία μιας ιστοσελίδας που διατίθενται για όλες τις συσκευές και τα προγράμματα περιήγησης.

7.4.10 Linq

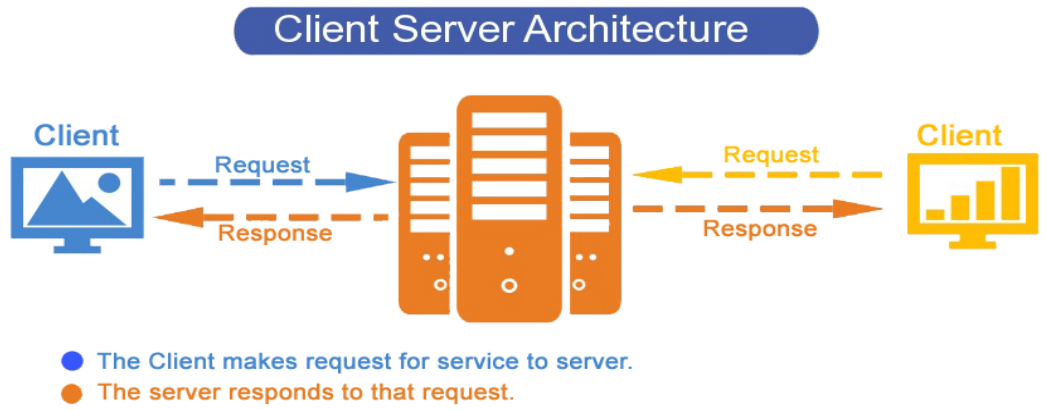
Η LINQ είναι μία γλώσσα παραγωγής ερωτημάτων (query language) της Microsoft, η οποία χρησιμοποιείται για την δημιουργία ερωτημάτων προς το μοντέλο των αντικειμένων και επιστρέφει Οντότητες. Ξέρουμε ότι, είναι πολύ εύκολο να βρει κανείς δεδομένα από SQL αντικείμενα απλά γράφοντας ένα ερώτημα, ενώ υπάρχει μία έντονη δυσκολία όταν θέλουμε να κάνουμε το ίδιο πράγμα σε ένα DataTable ή σε Λίστες. Σε γενικές γραμμές θα πρέπει με την χρήση ενός βρόχου να γίνει προσπέλαση σε κάθε στοιχείο για να βρεθεί η ακριβή αντιστοιχία. Αντιθέτως με την Linq παρέχεται ένας εύκολος τρόπος για να γράψουμε τα ερωτήματα που μπορούμε να τρέξουμε με αντικείμενα μνήμης. Μέσω της LINQ, μπορούμε να τρέχουμε Queries σε datasets όπως SQLite βάσεις, XML αρχεία ,Collections / Generics, Objects, Entities, Enumerables κ.α. Είναι ένα έξυπνο feature της C# που μας βοηθάει στην καλύτερη εκμετάλλευση δεδομένων σε μεγάλα datasets! Δεν χρειαζόμαστε middleware για την εκτέλεση τους! Όλα γίνονται στην C#! Η εκτέλεση της LINQ συνδέεται πάντα με LAMBDA εκφράσεις οι οποίες έχουν την έννοια των Annonymous Μεθόδων.

7.4.11 Αρχιτεκτονική Υλοποίησης - GUI

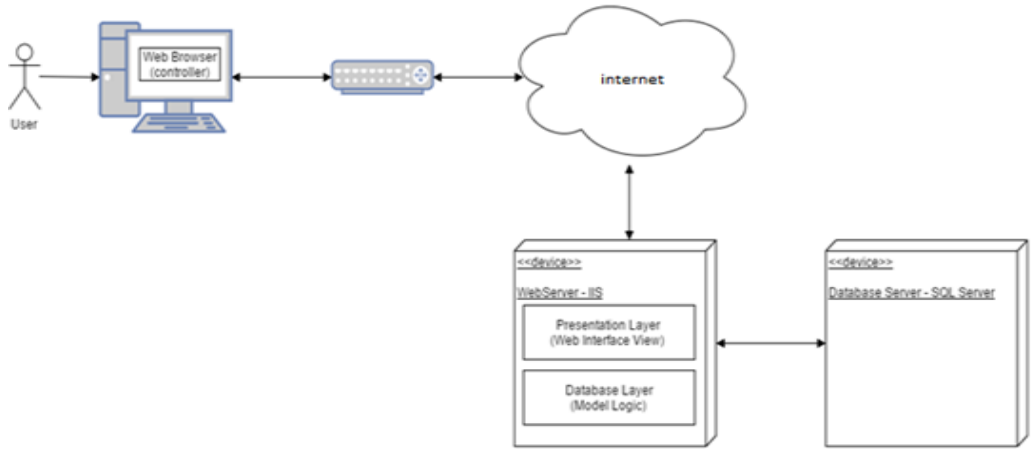
Γενικά, το client-server computing αναφέρεται σε μια βασική αλλαγή στο στυλ των υπολογιστών, την αλλαγή από τα συστήματα που βασίζονται στα μηχανήματα στα συστήματα που βασίζονται στον χρήστη.

Ειδικότερα, ένα σύστημα client-server είναι ένα σύστημα στο οποίο το δίκτυο ενώνει διάφορους υπολογιστικούς πόρους, ώστε οι clients (ή αλλιώς front end) να μπορούν να ζητούν υπηρεσίες από έναν server (ή αλλιώς back end), ο οποίος προσφέρει πληροφορίες ή επιπρόσθετη υπολογιστική ισχύ.

Με άλλα λόγια, στο client-server μοντέλο, ο client θέτει μια αίτηση και ο server επιστρέφει μια ανταπόκριση ή κάνει μια σειρά από ενέργειες. Ο server μπορεί να ενεργοποιείται άμεσα για την αίτηση αυτή ή να προσθέτει την αίτηση σε μια ουρά.

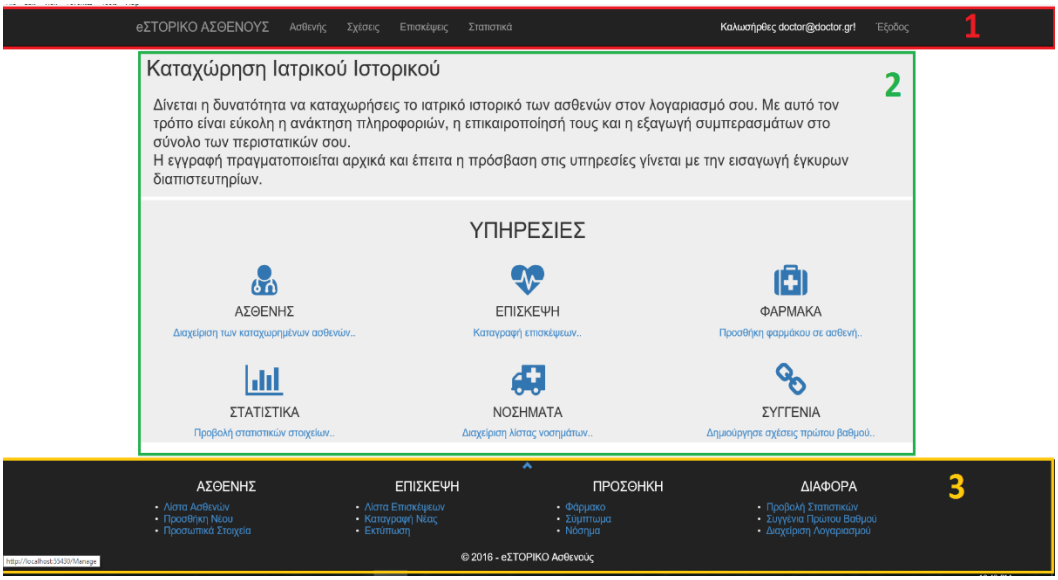


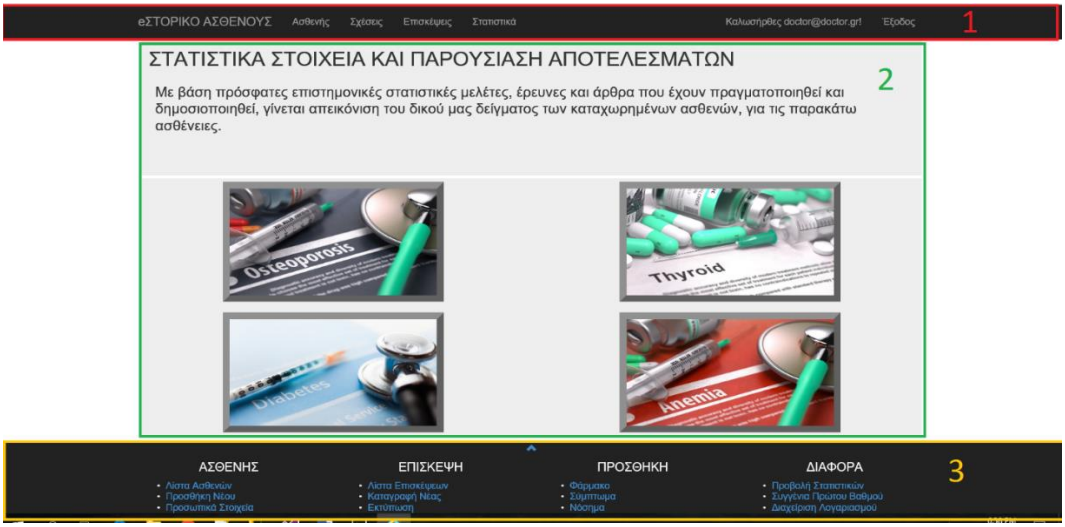
Όπως αναφέρθηκε και προηγουμένως η εφαρμογή συνδέεται με μία βάση δεδομένων και βασίζεται σε ένα γραφικό περιβάλλον στο οποίο ο χρήστης έχει πρόσβαση μέσω διαδικτύου. Έχοντας λάβει υπόψη τις συγκεκριμένες προδιαγραφές του έργου σχεδιάστηκε το ακόλουθο διάγραμμα διανομής (Deployment diagram) το οποίο δηλώνει με ποιο τρόπο είναι εγκατεστημένα τα διάφορα μέρη του συστήματος. Όπως φαίνεται στο ίδιο σύστημα υπάρχει η εφαρμογή (Application) και το σύστημα διαχείρισης βάσης δεδομένων (ΣΔΒΔ - SQL Server). Η εφαρμογή και το ΣΔΒΔ θεωρούνται περιβάλλοντα εκτέλεσης (με βάση το μοντέλο UML).



Το πώς υλοποιήθηκε η εμφάνιση του UI περιγράφεται παρακάτω αναλυτικά και με βάση την συγκεκριμένη φιλοσοφία στηρίχθηκε και η ανάπτυξη όλου του κώδικα και της λογικής.

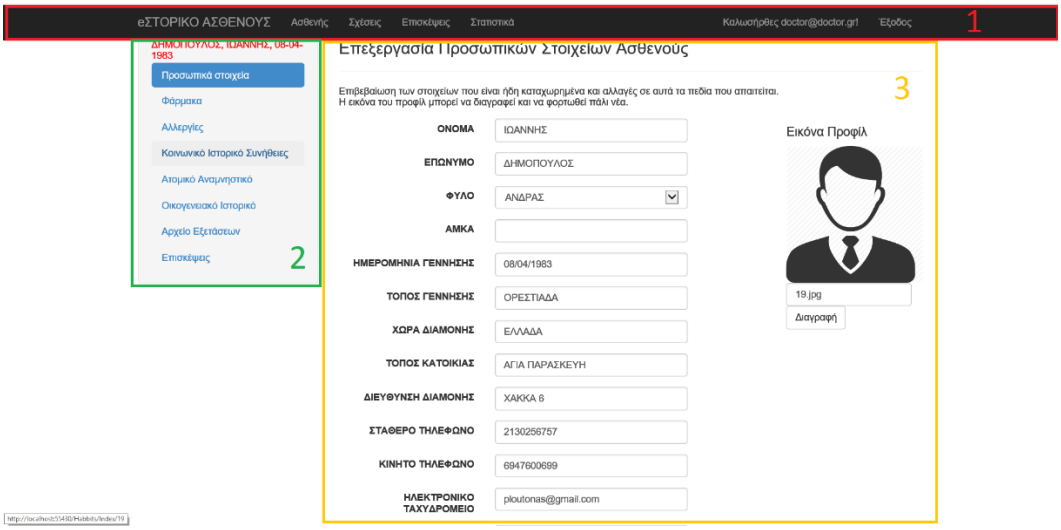
Αρχικά στην πρώτη εικόνα, φαίνεται το αρχικό παράθυρο της εφαρμογής, το οποίο έχει υποστεί κατάτμηση, σε τρία μέρη που αποτελούν και την κυρίως δομή του UI.
Το τμήμα 1 είναι το κυρίως μενού επιλογών. Περιλαμβάνει τις διαθέσιμες βασικές λειτουργίες της εφαρμογής καθώς και τα κουμπιά για την είσοδο ή την έξοδο και πληροφορίες του συνδεδεμένου χρήστη.
Το τμήμα 2 φιλοξενεί όλο τον όγκο της εφαρμογής. Εμφανίζονται πληροφορίες, πλαίσια διαλόγου, καρτέλες προβολής στοιχείων και συμπλήρωσης πεδίων. Είναι δυναμικό και αλλάζει ανάλογα το τι έχει επιλέξει να κάνει ο χρήστης.
Στο τμήμα 3 εμφανίζονται εν κατακλείδι συγκεντρωτικά κάποιες από τις διαθέσιμες επιλογές της εφαρμογής για τον συνδεδεμένο χρήστη.

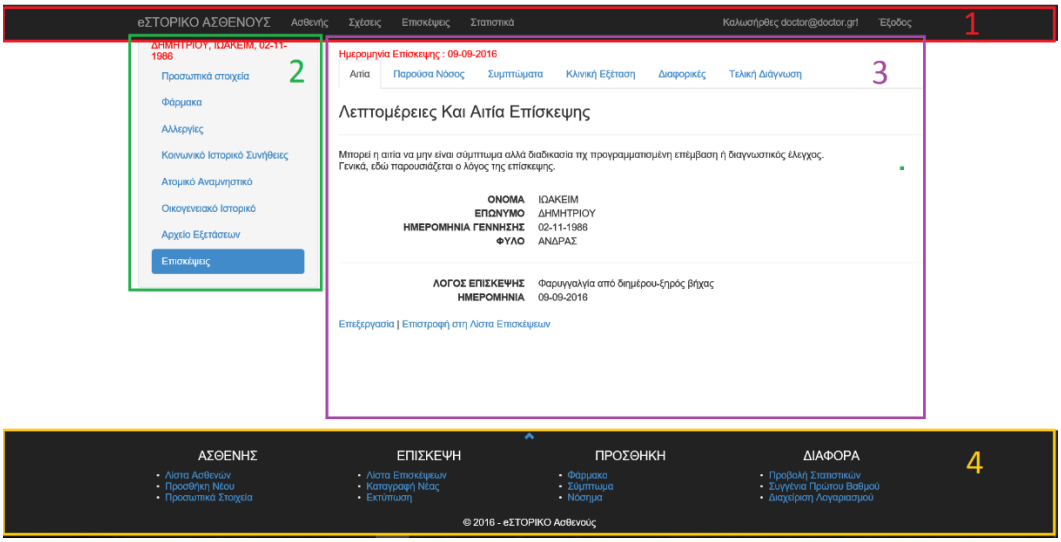




Αφού επιλεχθεί κάποιος ασθενής τα τμήματα 1 και 4 παραμένουν σταθερά και μπορεί ανά πάσα στιγμή ο χρήστης να τα επιλέξει. Στα αριστερά του παραθύρου και σε θέση sidebar εμφανίζεται μία λίστα με επιλογές που περιέχονται οι καρτέλες αναφορικά με τον ασθενή οι οποίες μπορούν να επεξεργαστούν. Έτσι ο χρήστης από το τμήμα 2 επιλέγει ποια στοιχεία θα ήθελε να προσθέσει ή να επεξεργαστεί. Αφού κάνει κλικ, στα δεξιά στο κεντρικό χώρο της οθόνης εμφανίζονται τα πεδία των σχετικών πληροφοριών.

Στην τελευταία εικόνα και αφού έχει δημιουργηθεί επίσκεψη για τον ασθενή, το τμήμα 3 διαφοροποιείται τελείως και εμπλουτίζεται με ένα οριζόντιο υπομενού καρτελών, από το οποίο ο χρήστης μπορεί να προσθέσει και να επεξεργαστεί τις πληροφορίες αναφορικά με την συγκεκριμένη επίσκεψη.





8 ΒΑΣΗ ΔΕΔΟΜΕΝΩΝ

8.1 Εισαγωγή

Η καρδία του συστήματός μας βρίσκεται στην Βάση Δεδομένων της. Από εκεί ξεκίνησαν όλα και αποτελεί τον βασικό μας κορμό, καθώς καθορίζει όλη την λειτουργικότητα και την αποτελεσματικότητά της.

Με τον όρο βάση δεδομένων εννοείται μία συλλογή από συστηματικά μορφοποιημένα σχετιζόμενα δεδομένα στα οποία είναι δυνατή η ανάκτηση δεδομένων μέσω αναζήτησης κατ' απαίτηση.

Ειδικότερα, στην επιστήμη της πληροφορικής και στην καθημερινή χρήση των ηλεκτρονικών υπολογιστών, με τον όρο βάσεις δεδομένων αναφερόμαστε σε οργανωμένες, διακριτές συλλογές σχετιζόμενων δεδομένων ηλεκτρονικά και ψηφιακά αποθηκευμένων, στο λογισμικό που χειρίζεται τέτοιες συλλογές (Σύστημα Διαχείρισης Βάσεων Δεδομένων, ή DBMS) και στο γνωστικό πεδίο που το μελετά. Πέρα από την εγγενή της ικανότητα να αποθηκεύει δεδομένα, η βάση δεδομένων παρέχει μέσω του σχεδιασμού και του τρόπου ιεράρχησης των δεδομένων, τα αποκαλούμενα συστήματα διαχείρισης περιεχομένου, δηλαδή τη δυνατότητα γρήγορης άντλησης και ανανέωσης των δεδομένων.

Με τον όρο σχεσιακή βάση δεδομένων εννοείται μία συλλογή δεδομένων οργανωμένη σε συσχετισμένους πίνακες που παρέχει ταυτόχρονα ένα μηχανισμό για ανάγνωση, εγγραφή, τροποποίηση ή και πιο πολύπλοκες διαδικασίες πάνω στα δεδομένα. Ο σκοπός μιας βάσης δεδομένων είναι η οργανωμένη αποθήκευση πληροφορίας και η δυνατότητα εξαγωγής της πληροφορίας αυτής, ιδίως σε πιο οργανωμένη μορφή, σύμφωνα με ερωτήματα που τίθενται στη σχεσιακή βάση δεδομένων.

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.

Οι πιο κύριες σήμερα στην αγορά Σχεσιακές βάσεις δεδομένων είναι:

- Oracle
- MySQL
- Microsoft SQL Server
- PostgreSQL

Το RDBMS που χρησιμοποιήθηκε για την κατασκευή της δικιάς μας ΒΔ είναι ο Microsoft SQL Server 2014.

8.2 Γενική Περιγραφή ΒΔ

Αρχικά σχεδιάστηκε το σχήμα της βάσης στο χαρτί με βάση την αρχική μας ανάλυση και στην συνέχεια δημιουργήθηκε η βάση δεδομένων μας η οποία θα φιλοξενούσε τα δεδομένα όλου του συστήματος. Τα συγκεκριμένα δεδομένα αφορούν τους χρήστες της εφαρμογής με τα διαπιστευτήρια και τις συνδέσεις τους, τα προσωπικά στοιχεία του ασθενούς με όλες τις πληροφορίες του ιατρικού του φακέλου, τις λεπτομέρειες για κάθε επίσκεψη που καταγράφεται. Για το σκοπό αυτό δημιουργούνται πίνακες, views, triggers και stored procedures.

Αναφορικά με το validation των δεδομένων δόθηκε ιδιαίτερη έμφαση στα πεδία των πινάκων και επιπρόσθετα έγινε χρήση των χαρακτηριστικών Data Annotations (System.ComponentModel.DataAnnotations.Schema) στις ιδιότητες της κλάσης μοντέλων που Entity Framework το οποίο χρησιμοποιήθηκε για την διασύνδεση.

8.3 Ανάλυση Απαιτήσεων ΒΔ

Οι παραδοχές με βάση της οποίες υλοποιήθηκε το σχήμα της Βάσης Δεδομένων μας είναι:

- Για την είσοδο στις λειτουργίες της εφαρμογής θα πρέπει να υπάρχουν έγκυρα διαπιστευτήρια. Ο χειριστής τα εισάγει και αφού διασταυρωθούν με αυτά στην βάση, αν τελικά υπάρχει επιβεβαίωση αποκτάται η πρόσβαση.
- Ο κάθε χρήστης-γιατρός έχει πρόσβαση μόνο στα άτομα τα οποία ο ίδιος έχει καταχωρήσει. Αυτό σημαίνει ότι του δίνεται η πρόσβαση να προβάλει και να τροποποιήσει μετέπειτα μόνο τους δικούς του ασθενείς.
- Κατά την προσθήκη ενός νέου ασθενούς, απαραίτητη είναι η συμπλήρωση ορισμένων ελάχιστων δημογραφικών στοιχείων. Μετά την αρχική καταχώρηση ο γιατρός θα μπορεί να τροποποιήσει τα δεδομένα αυτά σε οποιαδήποτε χρονική στιγμή.
- Κάθε άτομο-ασθενής μπορεί να παίρνει από καμία έως πολλές φαρμακευτικές αγωγές. Ο χειριστής θα μπορεί να εισάγει και να τροποποιήσει τα στοιχεία αυτά.
- Κάθε άτομο-ασθενής μπορεί να έχει από καμία έως πολλές αλλεργίες. Ο χειριστής θα μπορεί να εισάγει και να τροποποιήσει τα στοιχεία αυτά.
- Το κοινωνικό και οικογενειακό ιστορικό είναι μοναδικό για καθένα ασθενή. Ο χειριστής θα μπορεί να εισάγει και να τροποποιήσει τα στοιχεία αυτά.
- Οι σχέσεις συγγένειας πρώτου βαθμού δημιουργούνται μεταξύ δύο ατόμων κάθε φορά. Γίνονται έλεγχοι ώστε να μην επαναληφθεί κάποια σχέση, να μην προστεθεί αντίθετη σχέση με τα ίδια άτομα και για το αν το γένος του ατόμου είναι σύμφωνο με το είδος της σχέσεις (πχ δεν μπορεί να είναι μία γυναίκα πατέρας κάποιου). Συγγενείς πρώτου βαθμού είναι οι γονείς και τα παιδιά. Είναι πολύ σημαντικό εδώ να σημειώνονται και οι ασθενείς που έχει κάποιο μέλος, καθώς από αυτήν την πληροφορία εξάγονται κάποιες εκτιμήσεις για το αν ο ασθενής μας έχει πιθανότητες να εμφανίσει κάποια από αυτές.
- Στο ατομικό αναμνηστικό συμπεριλαμβάνονται τα νοσήματα και οι ασθένειες του ατόμου, καθώς και οι χειρουργικές επεμβάσεις του. Ο κάθε ασθενής μπορεί να έχει από καμία έως αρκετά νοσήματα ή και χειρουργεία.
- Η καταγραφή κάθε επίσκεψης πρέπει οπωσδήποτε να είναι συνδεδεμένη με έναν ασθενή. Πρέπει οπωσδήποτε να συμπληρωθούν η ημερομηνία επίσκεψης και ο λόγος που έφερε τον ασθενή στον γιατρό.
- Για να ολοκληρωθεί η συμπλήρωση όλων των στοιχείων που αφορούν την κάθε επίσκεψη, θα πρέπει ο γιατρός να συμπληρώσει λεπτομέρειες για την αιτία η οποία είναι μία για κάθε επίσκεψη. Στην συνέχεια καταγράφεται η παρούσα νόσος με την αρχική αξιολόγηση που πραγματοποιεί ο γιατρός, η οποία επίσης είναι μία για κάθε επίσκεψη.
- Τα συμπτώματα, οι περιγραφές για την δεδομένη κλινική εξέταση και οι διαγνώσεις που προκύπτουν, μπορεί να είναι από καμία έως πολλές για κάθε ξεχωριστή επίσκεψη του ίδιου ασθενή.

- Με βάση τις συγγένειες που έχουν δημιουργηθεί και τα νοσήματα τα οποία έχουν σημειωθεί για κάθε έναν ασθενή, εξάγονται στοιχεία αναφορικά με τις πιθανότητες κληρονόμησης κάποιων από αυτών.

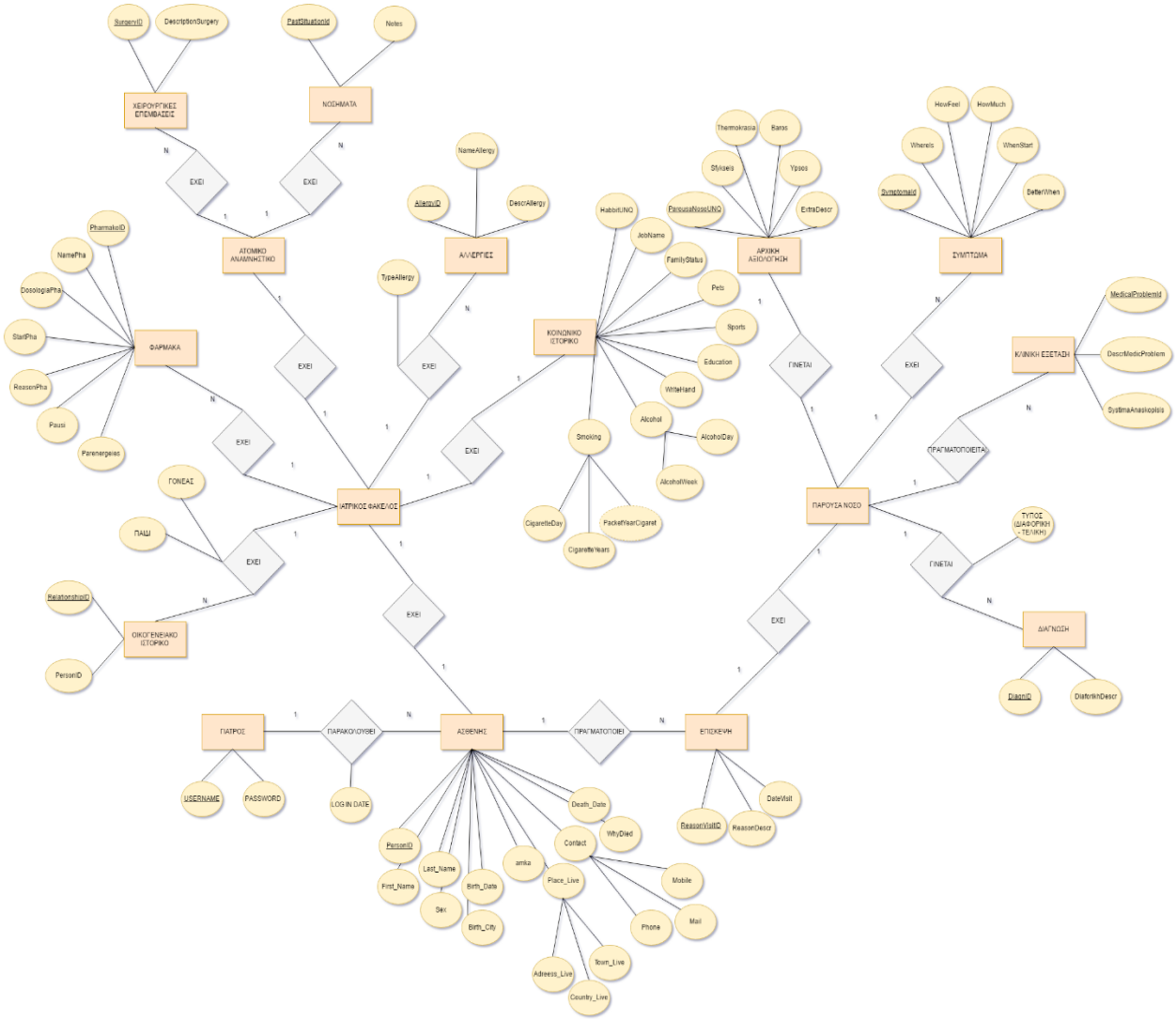
8.4 Εννοιολογικό Μοντέλο Οντοτήτων-Σχέσεων (E-R)

8.4.1 Γενικά

Το μοντέλο οντοτήτων-συσχετίσεων είναι ένα αφαιρετικό ιδεατό μοντέλο δεδομένων, τα οποία έχουν καθορισμένη δομή. Στη μηχανική λογισμικού χρησιμοποιείται για να παρέχει ένα εννοιολογικό σχήμα κατά τη σχεδίαση βάσεων δεδομένων, ως μοντέλο δεδομένων ενός συστήματος και των απαιτήσεών του με top-down προσέγγιση. Ένα διάγραμμα που δημιουργείται με αυτή τη διαδικασία σχεδίασης καλείται διάγραμμα οντοτήτων-συσχετίσεων, ή διάγραμμα Ο/Σ ή ΟΣΔ εν συντομία. Προτάθηκε αρχικά το 1976 από τον Peter Chen, ωστόσο στη συνέχεια επινοήθηκαν πολλές παραλλαγές της διαδικασίας.

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".

8.4.2 Υλοποίηση Διαγράμματος E-R στην ΒΔ



8.5 Σχεσιακό Μοντέλο Δεδομένων

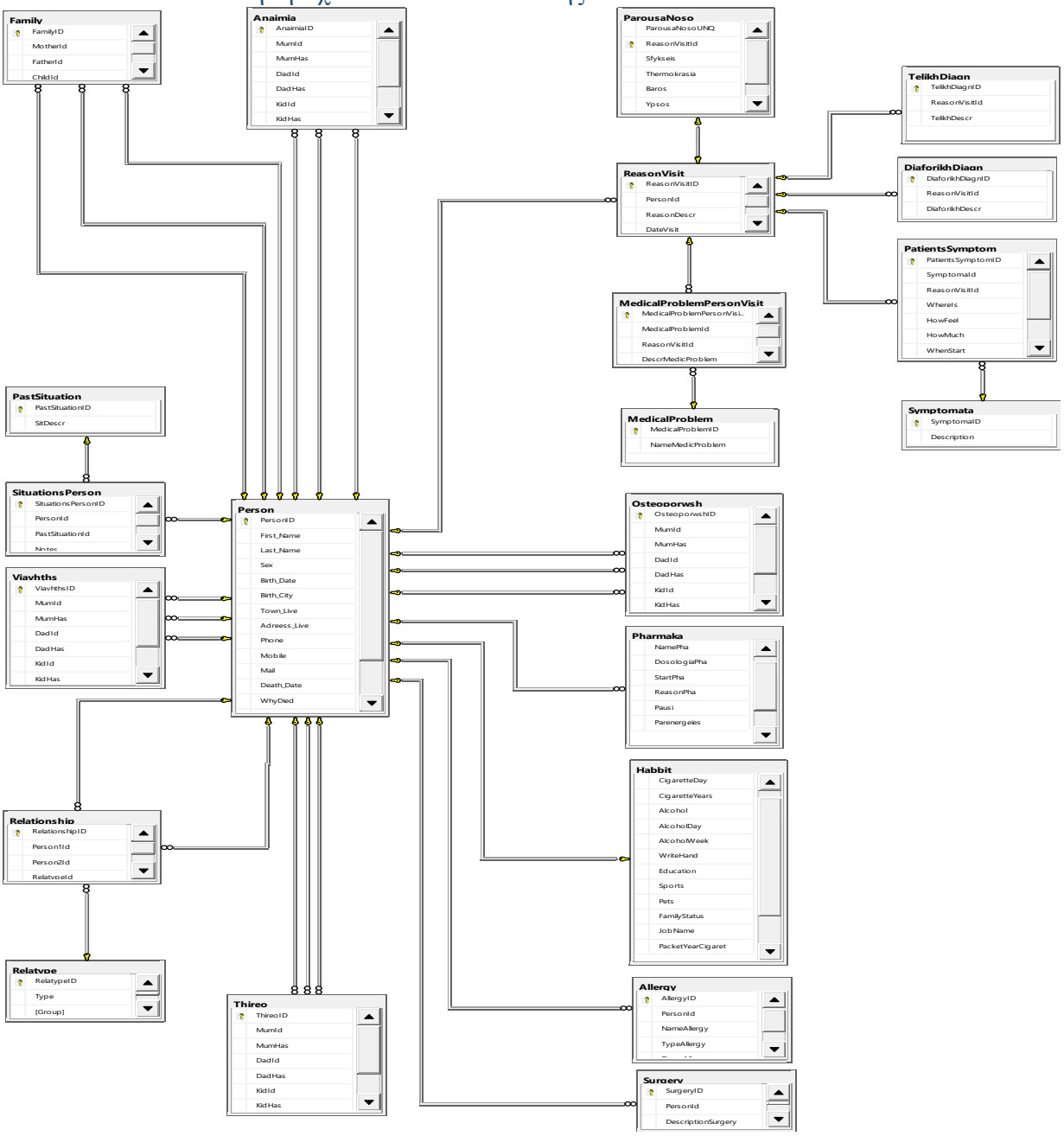
8.5.1 Γενικά

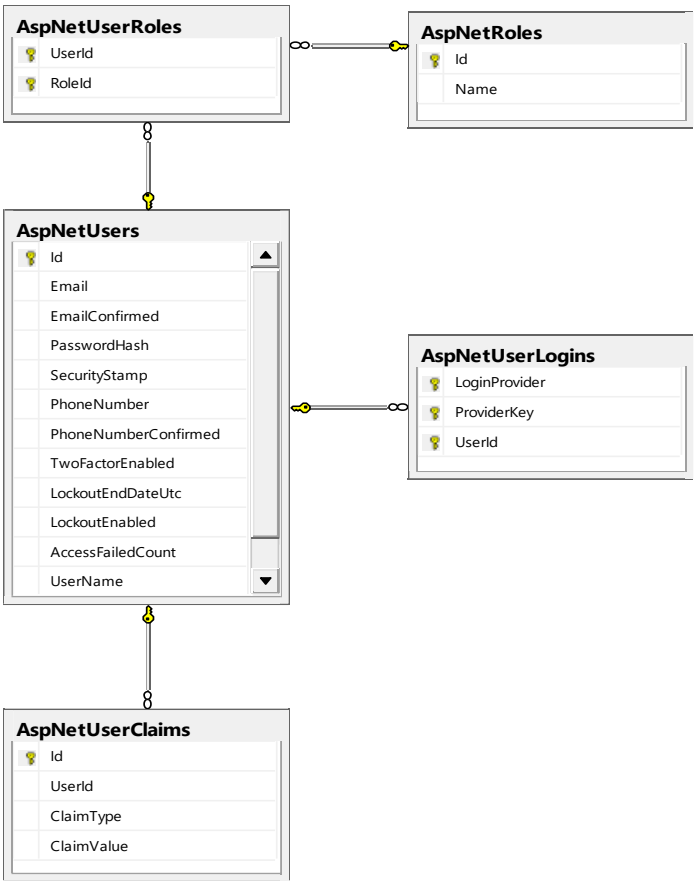
Το σχεσιακό μοντέλο δεδομένων προτάθηκε από Ted Codd της IBM Research το 1970.Ένα σύστημα διαχείρισης βάσης δεδομένων (ΣΔΒΔ) (database management system (DBMS)) αποτελείται από ένα σύνολο δεδομένων και προγράμματα πρόσβασης στα δεδομένα αυτά. Το σύνολο των δεδομένων καλείται βάση δεδομένων (database). Στόχος του ΣΔΒΔ είναι η εύκολη και γρήγορη χρήση και ανάκτηση των δεδομένων. Η διαχείριση των δεδομένων περιλαμβάνει:

- τον ορισμό δομών για τη αποθήκευση των δεδομένων
- τον ορισμό μεθόδων για τη διαχείριση των δεδομένων

Ο ορισμός της δομής της βάσης δεδομένων βασίζεται σε ένα μοντέλο δεδομένων το οποίο ορίζει τον τρόπο που περιγράφονται τα δεδομένα, οι σχέσεις τους, η σημασία τους και οι περιορισμοί πάνω στα δεδομένα αυτά.
Το σχεσιακό μοντέλο (relational model) δεδομένων παριστάνει δεδομένα και τις σχέσεις τους ως ένα σύνολο πινάκων. Κάθε πίνακας (table)αποτελείται από στήλες (columns) με μοναδικά ονόματα. Μια γραμμή (row) του πίνακα παριστάνει μια σχέση (relationship) ανάμεσα σε ένα σύνολο από τιμές.

8.5.2 Υλοποίηση Σχεσιακού Μοντέλου της ΒΔ





8.6 Παρουσίαση Αντικειμένων ΒΔ

8.6.1 Tables

Οι πίνακες, με την ονομασία, τα πεδία, τους τύπους και τα κλειδιά τους, που προέκυψαν από την ανάλυση απαιτήσεων του συστήματος, παρουσιάζονται συνοπτικά παρακάτω:

- Λογαριασμός χειριστή

BEIMAN-SATELITE.P...- dbo.AspNetUsers

Column Name	Data Type	Allow Nulls
Id	nvarchar(128)	☐
Email	nvarchar(256)	☑
EmailConfirmed	bit	☐
PasswordHash	nvarchar(MAX)	☑
SecurityStamp	nvarchar(MAX)	☑
PhoneNumber	nvarchar(MAX)	☑
PhoneNumberConfirmed	bit	☐
TwoFactorEnabled	bit	☐
LockoutEndDateUtc	datetime	☑
LockoutEnabled	bit	☐
AccessFailedCount	int	☐
UserName	nvarchar(256)	☐

Ο πίνακας αυτός χρησιμοποιείται για την αποθήκευση των χειριστών που είναι γιατροί και έχουν δημιουργήσει με την διαδικασία εγγραφής έναν έγκυρο λογαριασμό εισόδου.

- Ασθενής-Επισκέπτης

BEIMAN-SATELITE.P...story - dbo.Person | BEIMAN-SATELITE....o.PatientsSymptom

Column Name	Data Type	Allow Nulls
PersonID	int	☐
First_Name	nvarchar(50)	☐
Last_Name	nvarchar(50)	☐
Sex	nvarchar(50)	☐
Birth_Date	datetime	☑
Birth_City	nvarchar(50)	☑
Town_Live	nvarchar(50)	☑
Adreess_Live	nvarchar(50)	☑
Phone	nvarchar(MAX)	☑
Mobile	nvarchar(MAX)	☑
Mail	nvarchar(MAX)	☑
Death_Date	datetime	☑
WhyDied	nvarchar(MAX)	☑
Country_Live	nvarchar(50)	☑
userCreator	nvarchar(MAX)	☐
amka	varchar(50)	☑

Ο πίνακας αυτός χρησιμοποιείται για την αποθήκευση των δεδομένων των δημογραφικών προσωπικών στοιχείων του ασθενή που επισκέπτεται τον ιατρό.

- Ιατρικός Φάκελος

BEIMAN-SATELITE.Pa...tory - dbo.Allergy

Column Name	Data Type	Allow Nulls
AllergyID	int	☐
PersonId	int	☐
NameAllergy	nvarchar(50)	☑
TypeAllergy	nvarchar(MAX)	☑
DescrAllergy	nvarchar(MAX)	☑

Ο πίνακας αυτός χρησιμοποιείται για την αποθήκευση των δεδομένων των αλλεργιών που έχει ο ασθενής.

BEIMAN-SATELITE.P...story - dbo.Habbit

Column Name	Data Type	Allow Nulls
HabbitUNQ	int	☐
PersonId	int	☐
Smoking	nvarchar(50)	☑
CigaretteDay	nvarchar(50)	☑
CigaretteYears	nvarchar(50)	☑
Alcohol	nvarchar(50)	☑
AlcoholDay	nvarchar(50)	☑
AlcoholWeek	nvarchar(50)	☑
WriteHand	nvarchar(50)	☑
Education	nvarchar(50)	☑
Sports	nvarchar(50)	☑
Pets	nvarchar(50)	☑
FamilyStatus	nvarchar(50)	☑
JobName	nvarchar(MAX)	☑
PacketYearCigaret	nvarchar(50)	☑

Ο πίνακας αυτός χρησιμοποιείται για την αποθήκευση των δεδομένων των λεπτομερειών για το κοινωνικό ιστορικό του ασθενή.

BEIMAN-SATELITE.P...ry - dbo.Pharmaka

Column Name	Data Type	Allow Nulls
PharmakoID	int	☐
PersonId	int	☐
NamePha	nvarchar(50)	☐
DosologiaPha	nvarchar(50)	☑
StartPha	datetime	☑
ReasonPha	nvarchar(MAX)	☑
Pausi	nvarchar(10)	☑
Parenergeies	nvarchar(MAX)	☑

Ο πίνακας αυτός χρησιμοποιείται για την αποθήκευση των δεδομένων των φαρμάκων που αναφέρει ότι παίρνει ο ασθενής και συμπληρώνονται στοιχεία για την δοσολογία, το λόγο λήψης και την διάρκεια.

BEIMAN-SATELITE.P...- dbo.Relationship | BEIMAN-SATELITE.Pa...- dbo.ReasonVisit

Column Name	Data Type	Allow Nulls
RelationshipID	int	☐
Person1Id	int	☐
Person2Id	int	☐
RelatypeId	int	☐

Ο πίνακας αυτός χρησιμοποιείται για την αποθήκευση των δεδομένων των σχέσεων συγγένειας πρώτου βαθμού που έχει ο ασθενής και αποτελούν το οικογενειακό του ιστορικό.

Ο συγκεκριμένος πίνακας περιλαμβάνει και ένα trigger που ενεργοποιείται κατά το Insert ώστε να διασφαλίζονται κάποιες βασικές συνθήκες στις μεταξύ των ατόμων σχέσεις.

TRIGGER [dbo].[Relationship_check_trigger]

```
BEGIN
SET NOCOUNT ON;
DECLARE @RelationshipID INT;
DECLARE @Person1Id INT;
DECLARE @Person2Id INT;
DECLARE @RelatypeId INT;

SELECT @RelationshipID = INSERTED.RelationshipID
FROM INSERTED
SELECT @Person1Id = INSERTED.Person1Id
FROM INSERTED
SELECT @Person2Id = INSERTED.Person2Id
FROM INSERTED
SELECT @RelatypeId = INSERTED.RelatypeId
FROM INSERTED

IF (EXISTS(SELECT      [Person1Id],[Person2Id],[RelatypeId]
FROM [PatientHistory].[dbo].[Relationship]
WHERE [Person1Id] = @Person1Id AND
[Person2Id] = @Person2Id AND [RelatypeId] = @RelatypeId))
BEGIN
RAISERROR('ΥΠΑΡΧΕΙ Η ΣΥΓΚΕΚΡΙΜΕΝΗ ΣΧΕΣΗ',16 ,1)
--ROLLBACK TRANSACTION;
RETURN;
END

IF (EXISTS(SELECT      [Person1Id],[Person2Id],[RelatypeId]
```

```
FROM [PatientHistory].[dbo].[Relationship]
WHERE [Person1Id] = @Person2Id AND
[Person2Id] = @Person1Id AND [RelatypeId] = @RelatypeId))
BEGIN
RAISERROR('ΥΠΑΡΧΕΙ Η ΑΝΤΙΘΕΤΗ ΣΧΕΣΗ',16 ,1)
--ROLLBACK TRANSACTION;
RETURN;
END

IF (@Person2Id=@Person1Id)
BEGIN
RAISERROR('ΔΕΝ ΜΠΟΡΕΙ ΝΑ ΔΗΜΙΟΥΡΓΗΘΕΙ ΣΧΕΣΗ ΜΕ ΤΟ ΙΔΙΟ ΑΤΟΜΟ',16 ,1)
--ROLLBACK TRANSACTION;
RETURN
END

IF (@RelatypeId=1) --ΜΗΤΕΡΑ
IF (EXISTS(SELECT [PersonID] FROM [PatientHistory].[dbo].[Person] WHERE [PersonID]=@Person2Id
AND [Sex]='ΑΝΔΡΑΣ'))
BEGIN
RAISERROR('ΔΕΝ ΜΠΟΡΕΙ ΑΝΔΡΑΣ ΝΑ ΕΙΝΑΙ ΜΗΤΕΡΑ ΣΕ ΑΥΤΗ ΤΗ ΣΧΕΣΗ',16 ,1)
--ROLLBACK TRANSACTION;
RETURN;
END
IF (@RelatypeId=2) --PATERAS
IF (EXISTS(SELECT [PersonID] FROM [PatientHistory].[dbo].[Person] WHERE [PersonID]=@Person2Id
AND [Sex]='ΓΥΝΑΙΚΑ'))
BEGIN
RAISERROR('ΔΕΝ ΜΠΟΡΕΙ ΓΥΝΑΙΚΑ ΝΑ ΕΙΝΑΙ ΠΑΤΕΡΑΣ ΣΕ ΑΥΤΗ ΤΗ ΣΧΕΣΗ',16 ,1)
--ROLLBACK TRANSACTION;
RETURN;
END

INSERT INTO [dbo].[Relationship]
([Person1Id],[Person2Id],[RelatypeId])
VALUES
(@Person1Id,@Person2Id,@RelatypeId)
END
```

BEIMAN-SATELITE.P...ory - dbo.Relatype × BEIMAN-SATELITE.P...- dbo.Relationship

Column Name	Data Type	Allow Nulls
RelatypeID	int	☐
Type	nvarchar(MAX)	☐
[Group]	nvarchar(MAX)	☐

Ο πίνακας αυτός χρησιμοποιείται για την αποθήκευση των δεδομένων των κατηγοριών σχέσεων που μπορούν να πραγματοποιηθούν και σε αυτόν αντιστοιχεί το ξένο κλειδί από τον προηγούμενο πίνακα.

BEIMAN-SATELITE.P...tory - dbo.Surgery × BEIMAN-SATELITE.P...o.SituationsPerson

Column Name	Data Type	Allow Nulls
SurgeryID	int	☐
PersonId	int	☐
DescriptionSurgery	nvarchar(MAX)	☐

Ο πίνακας αυτός χρησιμοποιείται για την αποθήκευση των δεδομένων των χειρουργικών επεμβάσεων που έχει πραγματοποιήσει ο ασθενής.

- Επίσκεψη

BEIMAN-SATELITE.Pa...- dbo.ReasonVisit

Column Name	Data Type	Allow Nulls
ReasonVisitID	int	☐
PersonId	int	☐
ReasonDescr	nvarchar(MAX)	☑
DateVisit	datetime	☐

Ο πίνακας αυτός χρησιμοποιείται για την αποθήκευση των δεδομένων των επισκέψεων που πραγματοποιεί ο ασθενής στο γιατρό.

BEIMAN-SATELITE.P...roblemPersonVisit

Column Name	Data Type	Allow Nulls
MedicalProblemPersonVisitID	int	☐
MedicalProblemId	int	☐
ReasonVisitId	int	☐
DescrMedicProblem	nvarchar(MAX)	☑

Ο πίνακας αυτός χρησιμοποιείται για την αποθήκευση των δεδομένων των σοβαρών νοσημάτων και ασθενειών που είχε ο ασθενής.

BEIMAN-SATELITE....- dbo.ParousaNoso

Column Name	Data Type	Allow Nulls
ParousaNosoUNQ	int	☐
ReasonVisitId	int	☐
Sfykseis	nvarchar(50)	☑
Thermokrasia	decimal(5, 2)	☑
Baros	decimal(5, 2)	☑
Ypsos	decimal(5, 2)	☑
ExtraDescr	nvarchar(MAX)	☑

Ο πίνακας αυτός χρησιμοποιείται για την αποθήκευση των δεδομένων των στοιχείων της παρούσας εικόνας του ασθενή και προκύπτουν από την εξέταση του γιατρού.

BEIMAN-SATELITE....o.PatientsSymptom | BEIMAN-SATELITE....- dbo.ParousaNoso

Column Name	Data Type	Allow Nulls
PatientsSymptomID	int	☐
SymptomaId	int	☐
ReasonVisitId	int	☐
WhereIs	nvarchar(MAX)	☑
HowFeel	nvarchar(MAX)	☑
HowMuch	nvarchar(MAX)	☑
WhenStart	nvarchar(MAX)	☑
BetterWhen	nvarchar(MAX)	☑

Ο πίνακας αυτός χρησιμοποιείται για την αποθήκευση των δεδομένων συμπτωμάτων που παρουσιάζει ο ασθενής κατά την πραγματοποίηση της επίσκεψης και αναφέρει στο γιατρό.

BEIMAN-SATELITE.P...o.SituationsPerson | BEIMAN-SATELITE.P...ory - dbo.Relatype

Column Name	Data Type	Allow Nulls
SituationsPersonID	int	☐
PersonId	int	☐
PastSituationId	int	☐
Notes	nvarchar(MAX)	☑

Ο πίνακας αυτός χρησιμοποιείται για την αποθήκευση των δεδομένων των σοβαρών νοσημάτων και ασθενειών που έχει ο ασθενής.

BEIMAN-SATELITE.P...- dbo.Symptomata | BEIMAN-SATELITE.P...tory - dbo.Surgery

Column Name	Data Type	Allow Nulls
SymptomaID	int	☐
Description	nvarchar(50)	☐

Ο πίνακας αυτός χρησιμοποιείται για την αποθήκευση των δεδομένων των διαφόρων συμπτωμάτων που θέλει ο ιατρός να έχει αποθηκευμένες για να τις αντιστοιχεί στον κάθε ασθενή του.

BEIMAN-SATELITE.P...bo.DiaforikhDiagn

Column Name	Data Type	Allow Nulls
DiaforikhDiagnID	int	☐
ReasonVisitId	int	☐
DiaforikhDescr	nvarchar(MAX)	☑

Ο πίνακας αυτός χρησιμοποιείται για την αποθήκευση των δεδομένων των διαφορικών διαγνώσεων που έχει εξάγει ο ιατρός κατά την εξέταση του ασθενούς στην επίσκεψη.

BEIMAN-SATELITE.Pa...- dbo.TelikhDiagn | BEIMAN-SATELITE.P...- dbo.Symptomata

Column Name	Data Type	Allow Nulls
TelikhDiagnID	int	☐
ReasonVisitId	int	☐
TelikhDescr	nvarchar(MAX)	☑

Ο πίνακας αυτός χρησιμοποιείται για την αποθήκευση των δεδομένων των τελικών διαγνώσεων που έχει καταλήξει ο γιατρός για την συγκεκριμένη επίσκεψη του ασθενή του.

- Πρόβλεψη Κληρονομικότητας

BEIMAN-SATELITE.P...tory - dbo.Anaimia | BEIMAN-SATELITE.Pa...- dbo.TelikhDiagn

Column Name	Data Type	Allow Nulls
AnaimiaID	int	☐
MumId	int	☑
MumHas	nvarchar(MAX)	☑
DadId	int	☑
DadHas	nvarchar(MAX)	☑
KidId	int	☐
KidHas	nvarchar(MAX)	☑
PososTOCHILDREN	nvarchar(MAX)	☑

Ο πίνακας αυτός χρησιμοποιείται για την αποθήκευση των δεδομένων με τα άτομα που σχηματίζουν οικογένεια με βάση τις σχέσεις πρώτου βαθμού συγγένειας και για το αν τελικά τα παιδιά έχουν κακό ιστορικό για Μεσογειακή Αναιμία.

BEIMAN-SATELITE....- dbo.Osteoporwsh | BEIMAN-SATELITE.P...tory - dbo.Anaimia

Column Name	Data Type	Allow Nulls
OsteoporwshID	int	☐
MumId	int	☑
MumHas	nvarchar(MAX)	☑
DadId	int	☑
DadHas	nvarchar(MAX)	☑
KidId	int	☐
KidHas	nvarchar(MAX)	☑
PososTOCHILDREN	nvarchar(MAX)	☑

Ο πίνακας αυτός χρησιμοποιείται για την αποθήκευση των δεδομένων με τα άτομα που σχηματίζουν οικογένεια με βάση τις σχέσεις πρώτου βαθμού συγγένειας και για το αν τελικά τα παιδιά έχουν κακό ιστορικό για Οστεοπόρωση.

BEIMAN-SATELITE.Pa...story - dbo.Thireo | BEIMAN-SATELITE....- dbo.Osteoporwsh

Column Name	Data Type	Allow Nulls
ThireoID	int	☐
MumId	int	☑
MumHas	nvarchar(MAX)	☑
DadId	int	☑
DadHas	nvarchar(MAX)	☑
KidId	int	☐
KidHas	nvarchar(MAX)	☑
PosostoCHILDREN	nvarchar(MAX)	☑

Ο πίνακας αυτός χρησιμοποιείται για την αποθήκευση των δεδομένων με τα άτομα που σχηματίζουν οικογένεια με βάση τις σχέσεις πρώτου βαθμού συγγένειας και για το αν τελικά τα παιδιά έχουν κακό ιστορικό για Hashimoto.

BEIMAN-SATELITE.P...ory - dbo.Viavhths | BEIMAN-SATELITE.Pa...story - dbo.Thireo

Column Name	Data Type	Allow Nulls
ViavhthsID	int	☐
MumId	int	☑
MumHas	nvarchar(MAX)	☑
DadId	int	☑
DadHas	nvarchar(MAX)	☑
KidId	int	☐
KidHas	nvarchar(MAX)	☑
PosostoCHILDREN	nvarchar(MAX)	☑

Ο πίνακας αυτός χρησιμοποιείται για την αποθήκευση των δεδομένων με τα άτομα που σχηματίζουν οικογένεια με βάση τις σχέσεις πρώτου βαθμού συγγένειας και για το αν τελικά τα παιδιά έχουν κακό ιστορικό για Σακχαρώδη Διαβήτη.

8.6.2 Views

Για ευκολία στη σύνταξη ερωτημάτων sql και για να πετύχουμε την απεικόνιση των στοιχείων που επιθυμούμε φτιάξαμε δύο views. Ο σκοπός μας ήταν θα φέρουμε σε πίνακα όλα τα άτομα που έχουν οποιοδήποτε νόσημα και σε μία στήλη να μας εμφανίζει την ηλικία που έχουν σήμερα με βάση την ημερομηνία γέννησης που έχουν δώσει. Επομένως, οι προβολές που προέκυψαν από την ανάλυση απαιτήσεων του συστήματος παρουσιάζονται συνοπτικά παρακάτω:

BEIMAN-SATELITE....PersonNoshmaView

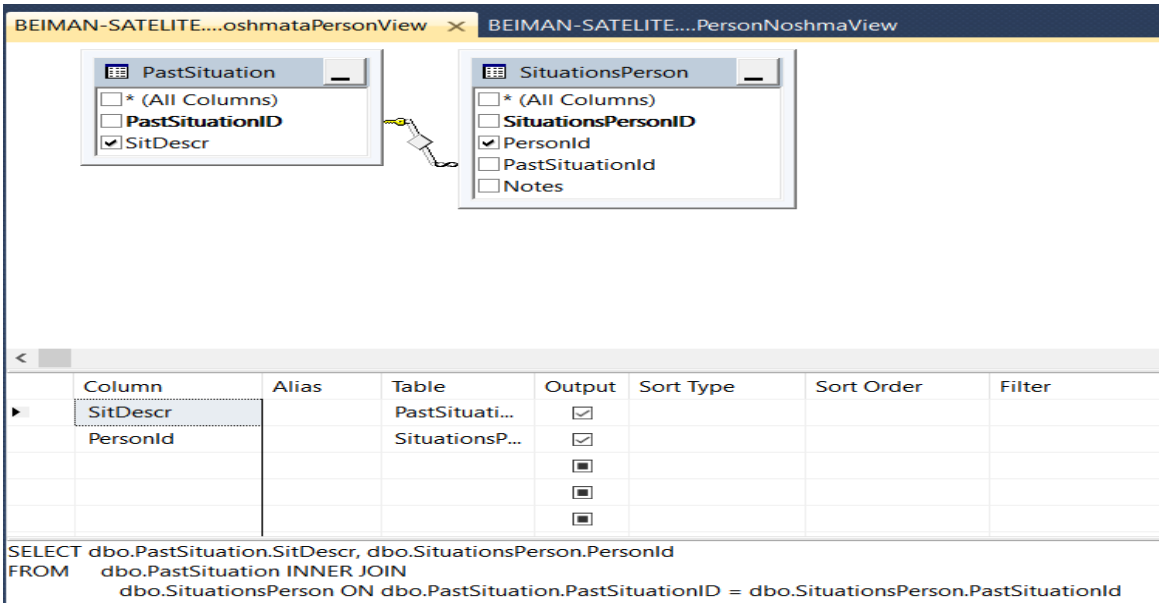
PastSituation: * (All Columns), PastSituationID, SitDescr

SituationsPerson: * (All Columns), SituationsPersonID, PersonId, PastSituationId, Notes

Person: PersonID, First_Name, Last_Name, Sex, Birth_Date

Column	Alias	Table	Output	Sort Type	Sort Order	Filter	Or...	Or...	Or...
PersonID		Person	☑						
Sex		Person	☑						
(CONVERT (int...	AGE		☑						
SitDescr		PastSituati...	☑						

```
SELECT dbo.Person.PersonID, dbo.Person.Sex, (CONVERT(int, CONVERT(char(8), GETDATE(), 112)) - CONVERT(char(8), dbo.Person.Birth_Date, 112)) / 10000 AS AGE, dbo.PastSituation.SitDescr
FROM    dbo.PastSituation INNER JOIN
            dbo.SituationsPerson ON dbo.PastSituation.PastSituationID = dbo.SituationsPerson.PastSituationId INNER JOIN
            dbo.Person ON dbo.SituationsPerson.PersonId = dbo.Person.PersonID
```



8.6.3 Procedures

Για να καταφέρω να εξετάσω τις πιθανότητες που μπορεί να έχει ένας ασθενής να εμφανίσει κάποια ασθένεια για το λόγο ότι την έχουν και οι γονείς του, υλοποίησα τέσσερις διαδικασίες, όσες είναι και τα νοσήματα που μελέτησα και συγκέντρωσα πληροφορίες. Ο σκοπός των διαδικασιών είναι να φτιάχνουν την οικογένεια για κάθε ασθενή δηλαδή ποιοι είναι οι γονείς του και να κρατάνε σαν πληροφορία τα δεδομένα από το ιστορικό του καθένα. Στην συνέχεια ανάλογα την ασθένεια, το ποιοι και πόσοι γονείς την έχουν, διαμορφώνεται και το ποσοστό που αφορά τα κάθε παιδί, προσθέτοντας την κατάλληλη πληροφορία λεκτικού στην αντίστοιχη στήλη. Οι διαδικασίες που προέκυψαν από την ανάλυση απαιτήσεων του συστήματος παρουσιάζονται συνοπτικά παρακάτω που παραθέτετε ο κώδικας sql για την τροποποίησή τους:

- Η διαδικασία που εξετάζει του γονείς, ελέγχει αν έχουν το στίγμα, αν νοσούν από την ασθένεια και αναλόγως τοποθετεί και τα παιδιά στην κατάλληλη κατηγορία πιθανότητας εμφάνισης Μεσογειακής Αναιμίας.

PROCEDURE [dbo].[parentsANAIMIA]

```
BEGIN
SET NOCOUNT ON;

DELETE [PatientHistory].[dbo].[Anaimia]

DECLARE @FatherID int
DECLARE @ChildID int
DECLARE @MotherID int
DECLARE @ChildExists int
DECLARE @AsthenisID int
DECLARE @MamHas varchar(60)
DECLARE @DadHas varchar(60)

BEGIN --CREATE FAMILY
```

```
--SXESEIS PAIDIOY PATERA
DECLARE Find_Father_CUR CURSOR
LOCAL STATIC READ_ONLY FORWARD_ONLY
FOR
SELECT [Person1Id],[Person2Id]  FROM [PatientHistory].[dbo].[Relationship] WHERE [RelatypeId]=2
OPEN Find_Father_CUR
FETCH NEXT FROM Find_Father_CUR INTO @ChildID,@FatherID
WHILE @@FETCH_STATUS = 0
BEGIN
INSERT INTO [PatientHistory].[dbo].[Anaimia] ([MumId],[DadId],[KidId])
VALUES    (null,@FatherID,@ChildID)

FETCH NEXT FROM Find_Father_CUR INTO @ChildID,@FatherID
END
CLOSE Find_Father_CUR
DEALLOCATE Find_Father_CUR

--SXESEIS PAIDIOY MHTERAS
DECLARE Find_Mother_CUR CURSOR
LOCAL STATIC READ_ONLY FORWARD_ONLY
FOR
SELECT [Person1Id],[Person2Id]  FROM [PatientHistory].[dbo].[Relationship] WHERE [RelatypeId]=1
OPEN Find_Mother_CUR
FETCH NEXT FROM Find_Mother_CUR INTO @ChildID,@MotherID
WHILE @@FETCH_STATUS = 0
BEGIN
SET @ChildExists = (SELECT COUNT(*) FROM [PatientHistory].[dbo].[Anaimia] WHERE
[KidId]=@ChildID)

--AN YPARXEI IDI TO PAIDI ME TON PATERA TOU
IF @ChildExists > 0
UPDATE [PatientHistory].[dbo].[Anaimia] SET [MumId] = @MotherID WHERE [KidId]=@ChildID;
ELSE
INSERT INTO [PatientHistory].[dbo].[Anaimia] ([MumId],[DadId],[KidId])
VALUES    (@MotherID,null,@ChildID) ;

FETCH NEXT FROM Find_Mother_CUR INTO @ChildID,@MotherID
END
CLOSE Find_Mother_CUR
DEALLOCATE Find_Mother_CUR
END--CREATE FAMILY

BEGIN --ASTHENEIS ME ANAIMIA STIGMA
--ATOMA ME ANAIMIA
DECLARE Find_ASTHENEIS_CUR CURSOR
LOCAL STATIC READ_ONLY FORWARD_ONLY
FOR
SELECT DISTINCT [PersonId] FROM [PatientHistory].[dbo].[NoshmataPersonView]
WHERE upper([SitDescr]) LIKE '%ΜΕΣΟΓ%ΑΝΑΙΜ%ΣΤ%ΓΜΑ%'
OPEN Find_ASTHENEIS_CUR
FETCH NEXT FROM Find_ASTHENEIS_CUR INTO @AsthenisID
WHILE @@FETCH_STATUS = 0
BEGIN
--ENHMERVSH TOY NOSHMATOS TOU STHN OIKOGENEIA
UPDATE [PatientHistory].[dbo].[Anaimia] SET [MumHas]='ΣΤΙΓΜΑ' WHERE [MumId]=@AsthenisID;
UPDATE [PatientHistory].[dbo].[Anaimia] SET [DadHas]='ΣΤΙΓΜΑ' WHERE [DadId]=@AsthenisID;
```

```
UPDATE [PatientHistory].[dbo].[Anaimia] SET [KidHas]='ΣΤΙΓΜΑ' WHERE [KidId]=@AsthenisID;

FETCH NEXT FROM Find_ASTHENEIS_CUR INTO @AsthenisID
END
CLOSE Find_ASTHENEIS_CUR
DEALLOCATE Find_ASTHENEIS_CUR
END --ASTHENEIS ME ANAIMIA ΣΤΙΓΜΑ

BEGIN --ASTHENEIS ME ANAIMIA ΝΟΣΟΥΝ
--ΑΤΟΜΑ ΜΕ ΑΝΑΙΜΙΑ ΠΟΥ ΝΟΣΟΣΥΝ
DECLARE Find_ASTHENEISnosun_CUR CURSOR
LOCAL STATIC READ_ONLY FORWARD_ONLY
FOR
SELECT DISTINCT [PersonId] FROM [PatientHistory].[dbo].[NoshmataPersonView]
WHERE upper([SitDescr]) LIKE '%ΜΕΣΟΓ%ΑΝΑΙΜ%ΝΟΣΕ%%'
OPEN Find_ASTHENEISnosun_CUR
FETCH NEXT FROM Find_ASTHENEISnosun_CUR INTO @AsthenisID
WHILE @@FETCH_STATUS = 0
BEGIN
--ΕΝΗΜΕΡΩΣΗ ΤΟΥ ΝΟΣΗΜΑΤΟΣ ΤΟΥ ΣΤΗΝ ΟΙΚΟΓΕΝΕΙΑ
UPDATE [PatientHistory].[dbo].[Anaimia] SET [MumHas]='ΝΟΣΕΙ' WHERE [MumId]=@AsthenisID;
UPDATE [PatientHistory].[dbo].[Anaimia] SET [DadHas]='ΝΟΣΕΙ' WHERE [DadId]=@AsthenisID;
UPDATE [PatientHistory].[dbo].[Anaimia] SET [KidHas]='ΝΟΣΕΙ' WHERE [KidId]=@AsthenisID;

FETCH NEXT FROM Find_ASTHENEISnosun_CUR INTO @AsthenisID
END
CLOSE Find_ASTHENEISnosun_CUR
DEALLOCATE Find_ASTHENEISnosun_CUR
END --ASTHENEIS ME ANAIMIA ΝΟΣΟΥΝ

BEGIN --ΠΙΘΑΝΟΤΗΤΕΣ ΠΑΙΔΙΟΥ
DECLARE Find_POSOSTO_CUR CURSOR
LOCAL STATIC READ_ONLY FORWARD_ONLY
FOR
SELECT [MumHas]
,[DadHas]
,[KidId]
FROM [PatientHistory].[dbo].[Anaimia]

OPEN Find_POSOSTO_CUR
FETCH NEXT FROM Find_POSOSTO_CUR INTO @MamHas,@DadHas,@ChildID
WHILE @@FETCH_STATUS = 0
BEGIN
--ΔΙΑΜΟΡΦΩΣΗ ΠΟΣΟΣΤΩΝ
--Δύο υγιείς γονείς -> κανένα παιδί
--Ένας γονέας με το στίγμα -> 50% να έχει στίγμα
--Δύο γονείς με στίγμα -> 25% να νοσήσει και 50% να έχει στίγμα ως υγιής φορέας
--Ένας με στίγμα και ένας να νοσεί -> 50% να νοσήσει με την ασθένεια της Μείζονος
--Θαλασσαιμίας και 50% να έχει το στίγμα ως υγιής φορέας

IF (((@MamHas='ΣΤΙΓΜΑ') AND (@DadHas IS NULL)) OR ((@MamHas IS NULL) AND
(@DadHas='ΣΤΙΓΜΑ')))
UPDATE [PatientHistory].[dbo].[Anaimia] SET [PososTOCHILDREN]='~50% ΝΑ ΕΜΦΑΝΙΣΕΙ' WHERE
[KidId]=@ChildID;
ELSE IF ((@MamHas='ΣΤΙΓΜΑ') AND (@DadHas='ΣΤΙΓΜΑ'))
```

```
UPDATE [PatientHistory].[dbo].[Anaimia] SET [Pososto CHILDREN]='~25% ΝΑ ΝΟΣΗΣΕΙ ΚΑΙ ~50% ΝΑ
ΕΧΕΙ ΣΤΙΓΜΑ' WHERE [KidId]=@ChildID;
ELSE IF (((@MamHas='ΣΤΙΓΜΑ') AND (@DadHas='ΝΟΣΕΙ')) OR ((@MamHas='ΝΟΣΕΙ') AND
(@DadHas='ΣΤΙΓΜΑ')))
UPDATE [PatientHistory].[dbo].[Anaimia] SET [Pososto CHILDREN]='~50% ΝΑ ΝΟΣΗΣΕΙ ΚΑΙ ~50% ΝΑ
ΕΧΕΙ ΣΤΙΓΜΑ' WHERE [KidId]=@ChildID;

FETCH NEXT FROM Find_POSOSTO_CUR INTO @MamHas,@DadHas,@ChildID
END
CLOSE Find_POSOSTO_CUR
DEALLOCATE Find_POSOSTO_CUR
END --PITHANOTHTES PAIDIOU
END
```

- Η διαδικασία που εξετάζει του γονείς, ελέγχει αν έχουν σακχαρώδη διαβήτη και αναλόγως τοποθετεί και τα παιδιά στην κατάλληλη κατηγορία πιθανότητας εμφάνισης Σακχαρώδους Διαβήτη.

PROCEDURE [dbo].[parentsDIAVITIS]

```
BEGIN
SET NOCOUNT ON;

DELETE [PatientHistory].[dbo].[Viavhths]

DECLARE @FatherID int
DECLARE @ChildID int
DECLARE @MotherID int
DECLARE @ChildExists int
DECLARE @AsthenisID int
DECLARE @MamHas varchar(60)
DECLARE @DadHas varchar(60)

BEGIN --CREATE FAMILY
--SXESEIS PAIDIOY PATERA
DECLARE Find_Father_CUR CURSOR
LOCAL STATIC READ_ONLY FORWARD_ONLY
FOR
SELECT [Person1Id],[Person2Id]  FROM [PatientHistory].[dbo].[Relationship] WHERE [RelatypeId]=2
OPEN Find_Father_CUR
FETCH NEXT FROM Find_Father_CUR INTO @ChildID,@FatherID
WHILE @@FETCH_STATUS = 0
BEGIN
INSERT INTO [PatientHistory].[dbo].[Viavhths] ([MumId],[DadId],[KidId])
VALUES    (null,@FatherID,@ChildID)

FETCH NEXT FROM Find_Father_CUR INTO @ChildID,@FatherID
END
CLOSE Find_Father_CUR
DEALLOCATE Find_Father_CUR

--SXESEIS PAIDIOY MHTERAS
DECLARE Find_Mother_CUR CURSOR
LOCAL STATIC READ_ONLY FORWARD_ONLY
```

```
FOR
SELECT [Person1Id],[Person2Id]  FROM [PatientHistory].[dbo].[Relationship] WHERE [RelatypeId]=1
OPEN Find_Mother_CUR
FETCH NEXT FROM Find_Mother_CUR INTO @ChildID,@MotherID
WHILE @@FETCH_STATUS = 0
BEGIN
SET @ChildExists = (SELECT COUNT(*) FROM [PatientHistory].[dbo].[Viavhths] WHERE
[KidId]=@ChildID)

--AN YPARXEI IDI TO PAIDI ME TON PATERA TOU
IF @ChildExists > 0
UPDATE [PatientHistory].[dbo].[Viavhths] SET [MumId] = @MotherID WHERE [KidId]=@ChildID;
ELSE
INSERT INTO [PatientHistory].[dbo].[Viavhths] ([MumId],[DadId],[KidId])
VALUES    (@MotherID,null,@ChildID) ;

FETCH NEXT FROM Find_Mother_CUR INTO @ChildID,@MotherID
END
CLOSE Find_Mother_CUR
DEALLOCATE Find_Mother_CUR
END--CREATE FAMILY

BEGIN --ASTHENEIS ME VIAVHTH
--ATOMA ME VIAVHTH
DECLARE Find_ASTHENEIS_CUR CURSOR
LOCAL STATIC READ_ONLY FORWARD_ONLY
FOR
SELECT DISTINCT [PersonId] FROM [PatientHistory].[dbo].[NoshmataPersonView]
WHERE upper([SitDescr]) LIKE '%ΔΙΑΒ%ΤΗΣ%'
OPEN Find_ASTHENEIS_CUR
FETCH NEXT FROM Find_ASTHENEIS_CUR INTO @AsthenisID
WHILE @@FETCH_STATUS = 0
BEGIN
--ENHMERVSH TOY NOSHMATOS TOU STHN OIKOGENEIA
UPDATE [PatientHistory].[dbo].[Viavhths] SET [MumHas]='Σακχαρώδης Διαβήτης' WHERE
[MumId]=@AsthenisID;
UPDATE [PatientHistory].[dbo].[Viavhths] SET [DadHas]='Σακχαρώδης Διαβήτης' WHERE
[DadId]=@AsthenisID;
UPDATE [PatientHistory].[dbo].[Viavhths] SET [KidHas]='Σακχαρώδης Διαβήτης' WHERE
[KidId]=@AsthenisID;

FETCH NEXT FROM Find_ASTHENEIS_CUR INTO @AsthenisID
END
CLOSE Find_ASTHENEIS_CUR
DEALLOCATE Find_ASTHENEIS_CUR
END --ASTHENEIS ME DIAVHTH

BEGIN --PITHANOTHTES PAIDIOU
DECLARE Find_POSOSTO_CUR CURSOR
LOCAL STATIC READ_ONLY FORWARD_ONLY
FOR
SELECT [MumHas]
,[DadHas]
,[KidId]
FROM [PatientHistory].[dbo].[Viavhths]
```

```
OPEN Find_POSOSTO_CUR
FETCH NEXT FROM Find_POSOSTO_CUR INTO @MamHas,@DadHas,@ChildID
WHILE @@FETCH_STATUS = 0
BEGIN
--DIAMORFWSH POSOSTWN
--Αν έχει διαβήτη ο ένας γονιός σας, έχετε 40% πιθανότητες να τον παρουσιάσετε κι εσείς.
--Αν έχουν και οι δύο, οι πιθανότητες για εσάς πλησιάζουν το 80%.
IF ((@MamHas='Σακχαρώδης Διαβήτης')and(@DadHas='Σακχαρώδης Διαβήτης'))
UPDATE [PatientHistory].[dbo].[Viavhths] SET [PososTOCHILDREN]='~80% ΝΑ ΕΜΦΑΝΙΣΕΙ' WHERE
[KidId]=@ChildID;
ELSE IF ((@MamHas IS NULL) AND (@DadHas IS NULL))
UPDATE [PatientHistory].[dbo].[Viavhths] SET [PososTOCHILDREN]=NULL WHERE [KidId]=@ChildID;
ELSE
UPDATE [PatientHistory].[dbo].[Viavhths] SET [PososTOCHILDREN]='~40% ΝΑ ΕΜΦΑΝΙΣΕΙ' WHERE
[KidId]=@ChildID;

FETCH NEXT FROM Find_POSOSTO_CUR INTO @MamHas,@DadHas,@ChildID
END
CLOSE Find_POSOSTO_CUR
DEALLOCATE Find_POSOSTO_CUR
END --PITHANOTHTES PAIDIOU
END
```

- Η διαδικασία που εξετάζει του γονείς, ελέγχει αν έχουν οστεοπόρωση και αναλόγως τοποθετεί και τα παιδιά στην κατάλληλη κατηγορία πιθανότητας εμφάνισης Οστεοπόρωσης.

PROCEDURE [dbo].[parentsOSTEOPORWSH]

```
BEGIN
SET NOCOUNT ON;

DELETE [PatientHistory].[dbo].[Osteoporwsh]

DECLARE @FatherID int
DECLARE @ChildID int
DECLARE @MotherID int
DECLARE @ChildExists int
DECLARE @AsthenisID int
DECLARE @MamHas varchar(60)
DECLARE @DadHas varchar(60)

BEGIN --CREATE FAMILY
--SXESEIS PAIDIOY PATERA
DECLARE Find_Father_CUR CURSOR
LOCAL STATIC READ_ONLY FORWARD_ONLY
FOR
SELECT [Person1Id],[Person2Id]  FROM [PatientHistory].[dbo].[Relationship] WHERE [RelatypeId]=2
OPEN Find_Father_CUR
FETCH NEXT FROM Find_Father_CUR INTO @ChildID,@FatherID
WHILE @@FETCH_STATUS = 0
BEGIN
INSERT INTO [PatientHistory].[dbo].[Osteoporwsh] ([MumId],[DadId],[KidId])
VALUES    (null,@FatherID,@ChildID)

FETCH NEXT FROM Find_Father_CUR INTO @ChildID,@FatherID
```

```
END
CLOSE Find_Father_CUR
DEALLOCATE Find_Father_CUR

--SXESEIS PAIDIOY MHTERAS
DECLARE Find_Mother_CUR CURSOR
LOCAL STATIC READ_ONLY FORWARD_ONLY
FOR
SELECT [Person1Id],[Person2Id]  FROM [PatientHistory].[dbo].[Relationship] WHERE [RelatypeId]=1
OPEN Find_Mother_CUR
FETCH NEXT FROM Find_Mother_CUR INTO @ChildID,@MotherID
WHILE @@FETCH_STATUS = 0
BEGIN
SET @ChildExists = (SELECT COUNT(*) FROM [PatientHistory].[dbo].[Osteoporwsh] WHERE
[KidId]=@ChildID)

--AN YPARXEI IDI TO PAIDI ME TON PATERA TOU
IF @ChildExists > 0
UPDATE [PatientHistory].[dbo].[Osteoporwsh] SET [MumId] = @MotherID WHERE [KidId]=@ChildID;
ELSE
INSERT INTO [PatientHistory].[dbo].[Osteoporwsh] ([MumId],[DadId],[KidId])
VALUES    (@MotherID,null,@ChildID) ;

FETCH NEXT FROM Find_Mother_CUR INTO @ChildID,@MotherID
END
CLOSE Find_Mother_CUR
DEALLOCATE Find_Mother_CUR
END--CREATE FAMILY

BEGIN --ASTHENEIS ME VIAVHTH
--ATOMA ME VIAVHTH
DECLARE Find_ASTHENEIS_CUR CURSOR
LOCAL STATIC READ_ONLY FORWARD_ONLY
FOR
SELECT DISTINCT [PersonId] FROM [PatientHistory].[dbo].[NoshmataPersonView]
WHERE upper([SitDescr]) LIKE '%ΟΣΤΕΟΠ%ΡΩΣΗ%'
OPEN Find_ASTHENEIS_CUR
FETCH NEXT FROM Find_ASTHENEIS_CUR INTO @AsthenisID
WHILE @@FETCH_STATUS = 0
BEGIN
--ENHMERVSH TOY NOSHMATOS TOU STHN OIKOGENEIA
UPDATE [PatientHistory].[dbo].[Osteoporwsh] SET [MumHas]='Οστεοπόρωση' WHERE
[MumId]=@AsthenisID;
UPDATE [PatientHistory].[dbo].[Osteoporwsh] SET [DadHas]='Οστεοπόρωση' WHERE [DadId]=@AsthenisID;
UPDATE [PatientHistory].[dbo].[Osteoporwsh] SET [KidHas]='Οστεοπόρωση' WHERE [KidId]=@AsthenisID;

FETCH NEXT FROM Find_ASTHENEIS_CUR INTO @AsthenisID
END
CLOSE Find_ASTHENEIS_CUR
DEALLOCATE Find_ASTHENEIS_CUR
END --ASTHENEIS ME DIAVHTH

BEGIN --PITHANOTHTES PAIDIOU
DECLARE Find_POSOSTO_CUR CURSOR
LOCAL STATIC READ_ONLY FORWARD_ONLY
FOR
```

```
SELECT [MumHas]
,[DadHas]
,[KidId]
FROM [PatientHistory].[dbo].[Osteoporwsh]

OPEN Find_POSOSTO_CUR
FETCH NEXT FROM Find_POSOSTO_CUR INTO @MamHas,@DadHas,@ChildID
WHILE @@FETCH_STATUS = 0
BEGIN
--DIAMORFWSH POSOSTWN
--Αν έχει οστεοπόρωση η μητέρα ή ο πατέρας σας, τότε έχετε 70% πιθανότητες να παρουσιάσετε κι εσείς.
--(Η οστεοπόρωση της γιαγιάς ή του παππού, όμως, δεν φαίνεται στατιστικά να σας αφορά ιδιαίτερα.)
IF (@MamHas=@DadHas)
UPDATE [PatientHistory].[dbo].[Osteoporwsh] SET [PososTOCHILDREN]='~70% ΝΑ ΕΜΦΑΝΙΣΕΙ' WHERE
[KidId]=@ChildID;
ELSE IF ((@MamHas IS NULL) AND (@DadHas IS NULL))
UPDATE [PatientHistory].[dbo].[Osteoporwsh] SET [PososTOCHILDREN]=NULL WHERE [KidId]=@ChildID;
ELSE
UPDATE [PatientHistory].[dbo].[Osteoporwsh] SET [PososTOCHILDREN]='~70% ΝΑ ΕΜΦΑΝΙΣΕΙ' WHERE
[KidId]=@ChildID;

FETCH NEXT FROM Find_POSOSTO_CUR INTO @MamHas,@DadHas,@ChildID
END
CLOSE Find_POSOSTO_CUR
DEALLOCATE Find_POSOSTO_CUR
END --PITHANOTHTES PAIDIOU
END
```

- Η διαδικασία που εξετάζει του γονείς, ελέγχει αν έχουν θυρεοειδίτιδα Χασιμότο και αναλόγως τοποθετεί και τα παιδιά στην κατάλληλη κατηγορία πιθανότητας εμφάνισης Θυρεοειδίτιδα Χασιμότο.

PROCEDURE [dbo].[parentsTHIREO]

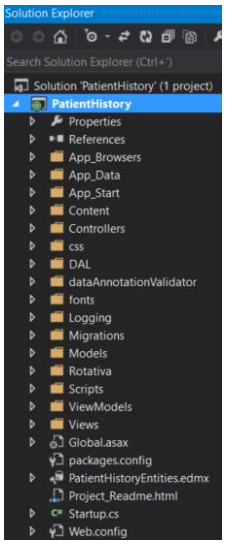
```
BEGIN
SET NOCOUNT ON;

DELETE [PatientHistory].[dbo].[Thireo]

DECLARE @FatherID int
DECLARE @ChildID int
DECLARE @MotherID int
DECLARE @ChildExists int
DECLARE @AsthenisID int
DECLARE @MamHas varchar(60)
DECLARE @DadHas varchar(60)

BEGIN --CREATE FAMILY
--SXESEIS PAIDIOY PATERA
DECLARE Find_Father_CUR CURSOR
LOCAL STATIC READ_ONLY FORWARD_ONLY
FOR
SELECT [Person1Id],[Person2Id]  FROM [PatientHistory].[dbo].[Relationship] WHERE [RelatypeId]=2
OPEN Find_Father_CUR
```

```
FETCH NEXT FROM Find_Father_CUR INTO @ChildID,@FatherID
WHILE @@FETCH_STATUS = 0
BEGIN
INSERT INTO [PatientHistory].[dbo].[Thireo] ([MumId],[DadId],[KidId])
VALUES    (null,@FatherID,@ChildID)

FETCH NEXT FROM Find_Father_CUR INTO @ChildID,@FatherID
END
CLOSE Find_Father_CUR
DEALLOCATE Find_Father_CUR

--SXESEIS PAIDIOY MHTERAS
DECLARE Find_Mother_CUR CURSOR
LOCAL STATIC READ_ONLY FORWARD_ONLY
FOR
SELECT [Person1Id],[Person2Id]  FROM [PatientHistory].[dbo].[Relationship] WHERE [RelatypeId]=1
OPEN Find_Mother_CUR
FETCH NEXT FROM Find_Mother_CUR INTO @ChildID,@MotherID
WHILE @@FETCH_STATUS = 0
BEGIN
SET @ChildExists = (SELECT COUNT(*) FROM [PatientHistory].[dbo].[Thireo] WHERE [KidId]=@ChildID)

--AN YPARXEI IDI TO PAIDI ME TON PATERA TOU
IF @ChildExists > 0
UPDATE [PatientHistory].[dbo].[Thireo] SET [MumId] = @MotherID WHERE [KidId]=@ChildID;
ELSE
INSERT INTO [PatientHistory].[dbo].[Thireo] ([MumId],[DadId],[KidId])
VALUES    (@MotherID,null,@ChildID) ;

FETCH NEXT FROM Find_Mother_CUR INTO @ChildID,@MotherID
END
CLOSE Find_Mother_CUR
DEALLOCATE Find_Mother_CUR
END--CREATE FAMILY

BEGIN --ASTHENEIS ME διαταραχή του θυρεοειδούς
--ATOMA ME διαταραχή του θυρεοειδούς
DECLARE Find_ASTHENEIS_CUR CURSOR
LOCAL STATIC READ_ONLY FORWARD_ONLY
FOR
SELECT DISTINCT [PersonId] FROM [PatientHistory].[dbo].[NoshmataPersonView]
WHERE upper([SitDescr]) LIKE '%ΘΥΡΕΟΕΙΔ%ΤΙΔΑ%HASHIMOTO%'
OPEN Find_ASTHENEIS_CUR
FETCH NEXT FROM Find_ASTHENEIS_CUR INTO @AsthenisID
WHILE @@FETCH_STATUS = 0
BEGIN
--ENHMERVSH TOY NOSHMATOS TOU STHN OIKOGENEIA
UPDATE [PatientHistory].[dbo].[Thireo] SET [MumHas]='HASHIMOTO' WHERE [MumId]=@AsthenisID;
UPDATE [PatientHistory].[dbo].[Thireo] SET [DadHas]='HASHIMOTO' WHERE [DadId]=@AsthenisID;
UPDATE [PatientHistory].[dbo].[Thireo] SET [KidHas]='HASHIMOTO' WHERE [KidId]=@AsthenisID;

FETCH NEXT FROM Find_ASTHENEIS_CUR INTO @AsthenisID
END
CLOSE Find_ASTHENEIS_CUR
DEALLOCATE Find_ASTHENEIS_CUR
END --ASTHENEIS ME διαταραχή του θυρεοειδούς
```

```
BEGIN --PITHANOTHTES PAIDIOU
DECLARE Find_POSOSTO_CUR CURSOR
LOCAL STATIC READ_ONLY FORWARD_ONLY
FOR
SELECT [MumHas]
,[DadHas]
,[KidId]
FROM [PatientHistory].[dbo].[Thireo]

OPEN Find_POSOSTO_CUR
FETCH NEXT FROM Find_POSOSTO_CUR INTO @MamHas,@DadHas,@ChildID
WHILE @@FETCH_STATUS = 0
BEGIN
--DIAMORFWSH POSOSTWN
--Πρόσφατες μελέτες δείχνουν ότι αν κάποιος πάσχει από κάποια διαταραχή του θυρεοειδούς,
--κάθε ένα από τα παιδιά του έχει 50% πιθανότητες να κληρονομήσει το γονίδιο
--και να παρουσιάσει κάποια στιγμή στη ζωή του πρόβλημα θυρεοειδούς.
IF ((@MamHas='HASHIMOTO')  OR  (@DadHas='HASHIMOTO'))
UPDATE [PatientHistory].[dbo].[Thireo] SET [PososTOCHILDREN]='~25% ΝΑ ΝΟΣΗΣΕΙ ΚΑΙ ~50% ΝΑ
ΕΜΦΑΝΙΣΕΙ ΔΙΑΤΑΡΑΧΗ ΤΟΥ ΘΥΡΕΟΕΙΔΟΥΣ' WHERE [KidId]=@ChildID;
ELSE
UPDATE [PatientHistory].[dbo].[Thireo] SET [PososTOCHILDREN]=NULL WHERE [KidId]=@ChildID;

FETCH NEXT FROM Find_POSOSTO_CUR INTO @MamHas,@DadHas,@ChildID
END
CLOSE Find_POSOSTO_CUR
DEALLOCATE Find_POSOSTO_CUR
END --PITHANOTHTES PAIDIOU
END
```

9 ΠΑΡΟΥΣΙΑΣΗ ΕΦΑΡΜΟΓΗΣ

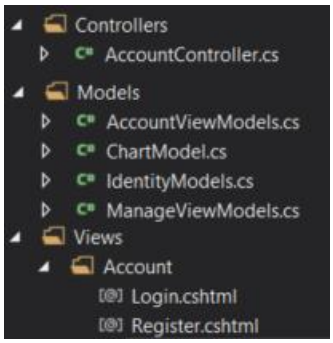
Η δομή της εφαρμογής βασίστηκε στο πρότυπο MVC όπως προαναφέραμε. Στα Models ενσωματώνεται η λογική, τα δεδομένα και οι κανόνες της εφαρμογής, στα Views είναι οι διεπαφές με τον χρήστη και στους Controllers χειρίζονται οι αλληλεπιδράσεις και ενημερώνεται το μοντέλο ή η προβολή.



Στην παραπάνω εικόνα παρουσιάζεται η δομή των φακέλων του project που δημιουργήσαμε στο VS για την εφαρμογή μας.

9.1 Σύνδεση Χειριστή

Την αρχική οθόνη μπορεί να την επισκεφθεί οποιοσδήποτε, αλλά για να αποκτηθεί πρόσβαση στις υπηρεσίες της εφαρμογής θα πρέπει να πραγματοποιηθεί επιτυχημένη σύνδεση με έγκυρα διαπιστευτήρια. Για τις πληροφορίες που χρειάζονται αποθήκευση και μετέπειτα ανάκτηση δημιουργήθηκε ο πίνακας [AspNetUsers]. Σε αυτόν κατά την πρώτη εγγραφή γίνεται εισαγωγεί των δεδομένων με τα στοιχεία των χρηστών και τον κωδικό πρόσβασης περασμένο από hash διαδικασία. Ενώ κατά την διαδικασία σύνδεσης γίνεται σύγκριση των διαπιστευτηρίων και αναλόγως αποκτάται ή όχι πρόσβαση.



Αρχεία που εμπλέκονται στη λειτουργία της εγγραφής και της σύνδεσης.

Για τις παραπάνω διαδικασίες η κύρια κλάση που χρησιμοποιείται είναι η ApplicationUser και κληρονομεί όπως βλέπουμε από την IdentityUser του namespace Microsoft.AspNet.Identity, το

οποίο προστέθηκε και χρησιμοποιήθηκε στο project. Ο ορισμός του αντικειμένου γίνεται στο αρχείο IdentityModels με κατάληξη cs.

```
public class ApplicationUser : IdentityUser
{
    public async Task<ClaimsIdentity> GenerateUserIdentityAsync(UserManager<ApplicationUser> manager)
    {
        var userIdentity = await manager.CreateIdentityAsync(this,
DefaultAuthenticationTypes.ApplicationCookie);
        return userIdentity;
    }
}
```

Επίσης στο ίδιο αρχείο ορίζεται και η κλάση ApplicationDbContext για να γίνεται η διαχείριση των δεδομένων μας ως αντικείμενα. Ως Default έχουμε ονομάσει την σύνδεση με την ΒΔ που γίνεται η αντίστοιχη δήλωση στο web.config αρχείο. Έτσι με την κλάση αυτή μετατρέπονται τα ερωτήματά μας που είναι σε Linq σε Sql ερωτήματα ώστε να αποσταλλούν στη ΒΔ.

```
public class ApplicationDbContext : IdentityDbContext<ApplicationUser>
{
    public ApplicationDbContext()
        : base("Default", throwIfV1Schema: false)
    {
    }

    public static ApplicationDbContext Create()
    {
        return new ApplicationDbContext();
    }
}
```

Οι οντότητες οι οποίες θα παίξουν σπουδαίο ρόλο σε αυτήν της φάση είναι το αντικείμενο που δημιουργώ κατά την εγγραφή και αυτό κατά την σύνδεση. Παρακάτω φαίνονται οι ιδιότητές τους.

```
namespace PatientHistory.Models (AccountViewModels.cs)
  public class LoginViewModel
  {
    [Required]
    [Display(Name = "Email")]
    [EmailAddress]
    public string Email { get; set; }

    [Required]
    [DataType(DataType.Password)]
    [Display(Name = "Password")]
    public string Password { get; set; }

    [Display(Name = "Να με θυμάσαι")]
    public bool RememberMe { get; set; }
  }

  public class RegisterViewModel
  {
    [Required]
```

```
    [EmailAddress]
    [Display(Name = "Email")]
    public string Email { get; set; }

    [Required]
    [StringLength(100, ErrorMessage = "The {0} must be at least {2} characters long.", MinimumLength = 6)]
    [DataType(DataType.Password)]
    [Display(Name = "Password")]
    public string Password { get; set; }

    [DataType(DataType.Password)]
    [Display(Name = "Confirm password")]
    [Compare("Password", ErrorMessage = "The password and confirmation password do not match.")]
    public string ConfirmPassword { get; set; }
  }
```

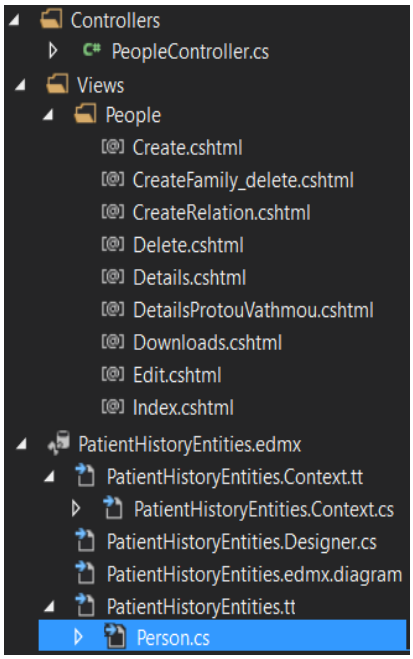
Μέσα στον ορισμό της κλάσης τύπου controller για τους λογαριασμούς, οι διαδικασίες που έχουμε καθορίζει είναι αυτή της εγγραφής και της σύνδεσης. Όπως φαίνεται παρακάτω τροφοδοτούνται με τα αντίστοιχα αντικείμενα που έχω αρχικά ορίσει και στη συνέχεια δημιουργήσει για να ολοκληρωθούν οι αλληλεπιδράσεις που επιθυμώ.

```
public class AccountController : Controller

    public async Task<ActionResult> Register(RegisterViewModel model)
    {
      if (ModelState.IsValid)
      {
        var user = new ApplicationUser { UserName = model.Email, Email = model.Email };
        var result = await UserManager.CreateAsync(user, model.Password);
        if (result.Succeeded)
        {
          await SignInManager.SignInAsync(user, isPersistent:false, rememberBrowser:false);

          return RedirectToAction("Index", "Home");
        }
        AddErrors(result);
      }

      // If we got this far, something failed, redisplay form
      return View(model);
    }

       public async Task<ActionResult> Login(LoginViewModel model, string returnUrl)
    {
      if (!ModelState.IsValid)
      {
        return View(model);
      }
      var result = await SignInManager.PasswordSignInAsync(model.Email, model.Password,
model.RememberMe, shouldLockout: false);
      switch (result)
      {
        case SignInStatus.Success:
          return RedirectToLocal(returnUrl);
        case SignInStatus.LockedOut:
```

```
            return View("Lockout");
        case SignInStatus.RequiresVerification:
            return RedirectToAction("SendCode", new { ReturnUrl = returnUrl, RememberMe =
model.RememberMe });
        case SignInStatus.Failure:
        default:
            ModelState.AddModelError("", "Αποτυχημένη Προσπάθεια Σύνδεσης.");
            return View(model);
    }
}
```

9.2 Λίστα Ασθενών και Καταχώρηση Νέου

Για να ξεκινήσει ο χειριστής να αξιοποιήσει τις λειτουργίες της εφαρμογής θα πρέπει να γίνει η προσθήκη του ασθενή. Για το σκοπό αυτό ορίστηκε η μία οντότητα ως person και το σύνολο ως people.



Οι ιδιότητες της οντότητας που κρατάει τα προσωπικά στοιχεία του ασθενή που πραγματοποιεί την επίσκεψη καθώς και οι σχέσεις που έχει με τις υπόλοιπες οντότητες, εμφανίζονται στο παρακάτω κώδικα. Χαρακτηριστικά μπορεί να αναφερθεί ότι υποχρεωτικά για την δημιουργία ενός αντικειμένου τύπου person είναι το όνοματεπώνυμο, το φύλο, η ημερομηνία γέννησης και επιπλέον μπορεί να ανέβει και φωτογραφία προφίλ.

```
public partial class Person
{
    [System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage",
"CA2214:DoNotCallOverridableMethodsInConstructors")]
    public Person()
    {
        this.Allergies = new HashSet<Allergy>();
        this.DiseaseAdults = new HashSet<DiseaseAdult>();
        this.DiseaseChilds = new HashSet<DiseaseChild>();
        this.Relationships = new HashSet<Relationship>();
        this.Relationships1 = new HashSet<Relationship>();
```

```
        this.Pharmakas = new HashSet<Pharmaka>();
        this.ReasonVisits = new HashSet<ReasonVisit>();
        this.SituationsPersons = new HashSet<SituationsPerson>();
        this.Surgeries = new HashSet<Surgery>();
        this.Anaimias = new HashSet<Anaimia>();
        this.Anaimias1 = new HashSet<Anaimia>();
        this.Anaimias2 = new HashSet<Anaimia>();
        this.Families = new HashSet<Family>();
        this.Families1 = new HashSet<Family>();
        this.Families2 = new HashSet<Family>();
        this.Osteoporwshes = new HashSet<Osteoporwsh>();
        this.Osteoporwshes1 = new HashSet<Osteoporwsh>();
        this.Osteoporwshes2 = new HashSet<Osteoporwsh>();
        this.Viavhths = new HashSet<Viavhth>();
        this.Viavhths1 = new HashSet<Viavhth>();
        this.Viavhths2 = new HashSet<Viavhth>();
        this.Thireos = new HashSet<Thireo>();
        this.Thireos1 = new HashSet<Thireo>();
        this.Thireos2 = new HashSet<Thireo>();
    }

    public int PersonID { get; set; }

    [Required(ErrorMessage = "Το πεδίο Όνομα είναι υποχρεωτικό")]
    [StringLength(50, ErrorMessage = "Το ΟΝΟΜΑ δεν μπορεί να είναι μεγαλύτερο από 50 χαρακτήρες.")]
    [Display(Name = "ΟΝΟΜΑ")]
    public string First_Name { get; set; }

    [Required(ErrorMessage = "Το πεδίο ΕΠΩΝΥΜΟ είναι υποχρεωτικό")]
    [StringLength(50, ErrorMessage = "Το Επώνυμο δεν μπορεί να είναι μεγαλύτερο από 50 χαρακτήρες.")]
    [Display(Name = "ΕΠΩΝΥΜΟ")]
    public string Last_Name { get; set; }

    [Required(ErrorMessage = "Το πεδίο ΦΥΛΟ είναι υποχρεωτικό")]
    [Display(Name = "ΦΥΛΟ")]
    public string Sex { get; set; }

    [Required(ErrorMessage = "Το πεδίο ΗΜΕΡΟΜΗΝΙΑ ΓΕΝΝΗΣΗΣ είναι υποχρεωτικό")]
    [Display(Name = "ΗΜΕΡΟΜΗΝΙΑ ΓΕΝΝΗΣΗΣ")]
    //[CheckDateRange]
    [DisplayFormat(DataFormatString = "{0:dd/MM/yyyy}", ApplyFormatInEditMode = true)]
    public DateTime Birth_Date { get; set; }

    [StringLength(50)]
    [Display(Name = "ΤΟΠΟΣ ΓΕΝΝΗΣΗΣ")]
    public string Birth_City { get; set; }

    [Display(Name = "ΧΩΡΑ ΔΙΑΜΟΝΗΣ")]
    public string Country_Live { get; set; }

    [StringLength(50)]
    [Display(Name = "ΤΟΠΟΣ ΚΑΤΟΙΚΙΑΣ")]
    public string Town_Live { get; set; }

    [StringLength(50)]
    [Display(Name = "ΔΙΕΥΘΥΝΣΗ ΔΙΑΜΟΝΗΣ")]
```

```
    public string Adreess_Live { get; set; }

    [DataType(DataType.PhoneNumber)]
    [RegularExpression(@"^\(?([0-9]{3})\)?[-. ]?([0-9]{3})[-. ]?([0-9]{4})$", ErrorMessage = "ΜΗ Έγκυρος
Αριθμός Τηλεφώνου")]
    [Display(Name = "ΣΤΑΘΕΡΟ ΤΗΛΕΦΩΝΟ")]
    public string Phone { get; set; }

    [DataType(DataType.PhoneNumber)]
    [RegularExpression(@"^\(?([0-9]{3})\)?[-. ]?([0-9]{3})[-. ]?([0-9]{4})$", ErrorMessage = "ΜΗ Έγκυρος
Αριθμός Κινητού")]
    [Display(Name = "ΚΙΝΗΤΟ ΤΗΛΕΦΩΝΟ")]
    public string Mobile { get; set; }

    [EmailAddress(ErrorMessage = "ΜΗ Έγκυρος Λογαριασμός Ηλεκτρονικού Ταχυδρομείου")]
    [Display(Name = "ΗΛΕΚΤΡΟΝΙΚΟ ΤΑΧΥΔΡΟΜΕΙΟ")]
    public string Mail { get; set; }

    [Display(Name = "ΗΜΕΡΟΜΗΝΙΑ ΘΑΝΑΤΟΥ")]
    //[CheckDateRange]
    [DisplayFormat(DataFormatString = "{0:dd/MM/yyyy}", ApplyFormatInEditMode = true)]
    public Nullable<System.DateTime> Death_Date { get; set; }

    [Display(Name = "ΑΙΤΙΑ ΘΑΝΑΤΟΥ")]
    [DataType(DataType.MultilineText)]
    [StringLength(100, ErrorMessage = "Η ΑΙΤΙΟΛΟΓΙΑ ΘΑΝΑΤΟΥ δεν μπορεί να είναι μεγαλύτερο από 100
χαρακτήρες.")]
    public string WhyDied { get; set; }

    [Required]
    public string userCreator { get; set; }

    [Display(Name = "ΟΝΟΜΑΤΕΠΩΝΥΜΟ-ΗΜΕΡΟΜΗΝΙΑ ΓΕΝΝΗΣΗ")]
    public string Full_Name_BirthDay
    {
      get
      {
        return Last_Name + ", " + First_Name + ", " + Birth_Date.ToString("dd/MM/yyyy");
      }
    }

    public string AvatarImage
    {
      get { return PersonID.ToString().Replace(" ", string.Empty) + ".jpg"; }
    }

    [Display(Name = "ΑΜΚΑ")]
    [DataType(DataType.PhoneNumber)]
    [RegularExpression(@"^\(?([0-9]{4})\)?[-. ]?([0-9]{4})[-. ]?([0-9]{3})$", ErrorMessage = "ΜΗ Έγκυρος
ΑΜΚΑ")]
    public string amka { get; set; }

    [System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage",
"CA2227:CollectionPropertiesShouldBeReadOnly")]
    public virtual ICollection<Allergy> Allergies { get; set; }
```

```
    [System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage",
"CA2227:CollectionPropertiesShouldBeReadOnly")]
    public virtual ICollection<DiseaseAdult> DiseaseAdults { get; set; }
    [System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage",
"CA2227:CollectionPropertiesShouldBeReadOnly")]
    public virtual ICollection<DiseaseChild> DiseaseChilds { get; set; }
    public virtual Habbit Habbit { get; set; }
    [System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage",
"CA2227:CollectionPropertiesShouldBeReadOnly")]
    public virtual ICollection<Relationship> Relationships { get; set; }
    [System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage",
"CA2227:CollectionPropertiesShouldBeReadOnly")]
    public virtual ICollection<Relationship> Relationships1 { get; set; }
    [System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage",
"CA2227:CollectionPropertiesShouldBeReadOnly")]
    public virtual ICollection<Pharmaka> Pharmakas { get; set; }
    [System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage",
"CA2227:CollectionPropertiesShouldBeReadOnly")]
    public virtual ICollection<ReasonVisit> ReasonVisits { get; set; }
    [System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage",
"CA2227:CollectionPropertiesShouldBeReadOnly")]
    public virtual ICollection<SituationsPerson> SituationsPersons { get; set; }
    [System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage",
"CA2227:CollectionPropertiesShouldBeReadOnly")]
    public virtual ICollection<Surgery> Surgeries { get; set; }
    [System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage",
"CA2227:CollectionPropertiesShouldBeReadOnly")]
    public virtual ICollection<Anaimia> Anaimias { get; set; }
    [System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage",
"CA2227:CollectionPropertiesShouldBeReadOnly")]
    public virtual ICollection<Anaimia> Anaimias1 { get; set; }
    [System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage",
"CA2227:CollectionPropertiesShouldBeReadOnly")]
    public virtual ICollection<Anaimia> Anaimias2 { get; set; }
    [System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage",
"CA2227:CollectionPropertiesShouldBeReadOnly")]
    public virtual ICollection<Family> Families { get; set; }
    [System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage",
"CA2227:CollectionPropertiesShouldBeReadOnly")]
    public virtual ICollection<Family> Families1 { get; set; }
    [System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage",
"CA2227:CollectionPropertiesShouldBeReadOnly")]
    public virtual ICollection<Family> Families2 { get; set; }
    [System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage",
"CA2227:CollectionPropertiesShouldBeReadOnly")]
    public virtual ICollection<Osteoporwsh> Osteoporwshes { get; set; }
    [System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage",
"CA2227:CollectionPropertiesShouldBeReadOnly")]
    public virtual ICollection<Osteoporwsh> Osteoporwshes1 { get; set; }
    [System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage",
"CA2227:CollectionPropertiesShouldBeReadOnly")]
    public virtual ICollection<Osteoporwsh> Osteoporwshes2 { get; set; }
    [System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage",
"CA2227:CollectionPropertiesShouldBeReadOnly")]
    public virtual ICollection<Viavhth> Viavhths { get; set; }
```

```
    [System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage",
"CA2227:CollectionPropertiesShouldBeReadOnly")]
    public virtual ICollection<Viavhth> Viavhths1 { get; set; }
    [System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage",
"CA2227:CollectionPropertiesShouldBeReadOnly")]
    public virtual ICollection<Viavhth> Viavhths2 { get; set; }
    [System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage",
"CA2227:CollectionPropertiesShouldBeReadOnly")]
    public virtual ICollection<Thireo> Thireos { get; set; }
    [System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage",
"CA2227:CollectionPropertiesShouldBeReadOnly")]
    public virtual ICollection<Thireo> Thireos1 { get; set; }
    [System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage",
"CA2227:CollectionPropertiesShouldBeReadOnly")]
    public virtual ICollection<Thireo> Thireos2 { get; set; }
  }
```

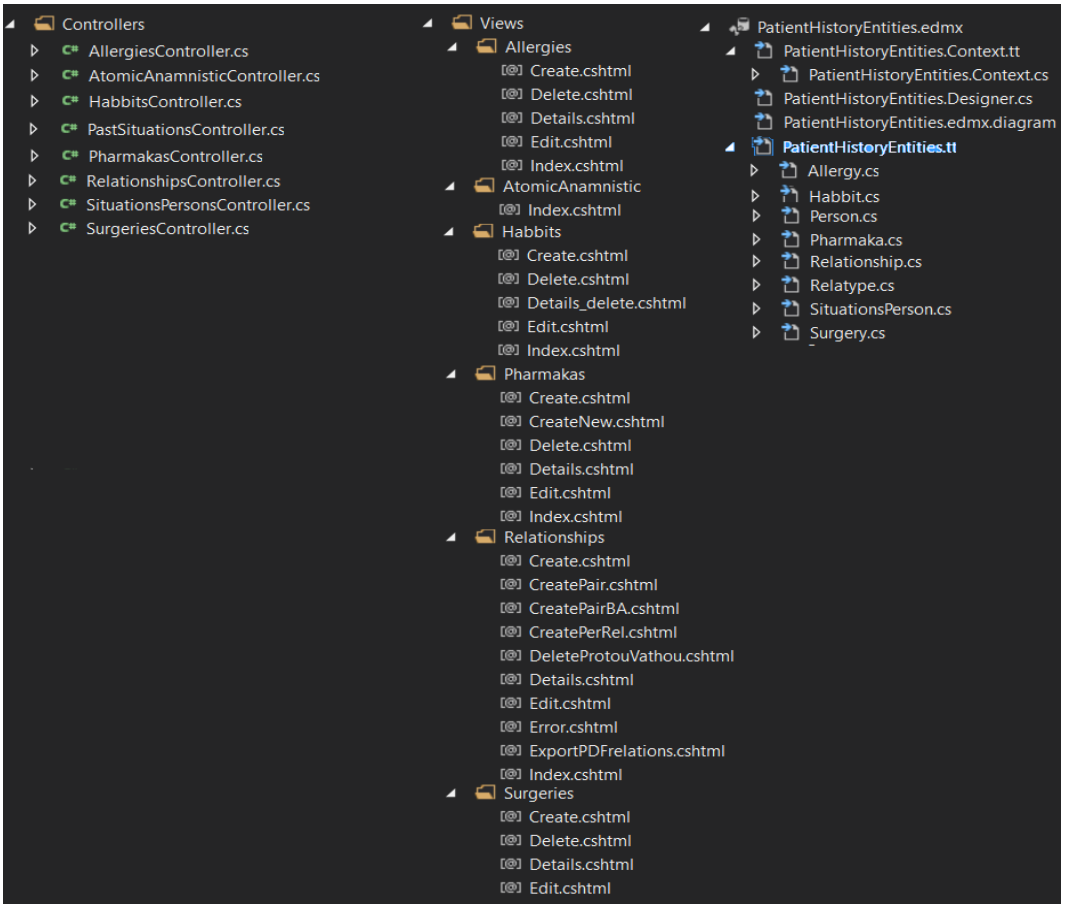
Σε αυτήν την κλάση δημιουργήθηκαν οι διαδικασίες οι οποίες έχουν ως βάση αναφοράς τον ασθενή. Εδώ καθορίστηκε η λειτουργικότητα της κύριας σελίδας που προβάλει το σύνολο των ασθενών, με την ταξινόμηση, την αναζήτηση και την σελιδοποίηση των εγγραφών. Επίσης γίνεται ο έλεγχος για να στοιχεία που πρόκειται να εισαχθούν στην ΒΔ κατά την προσθήκη νέου ατόμου. Επιπρόσθετα περιλαμβάνονται και οι στανταρτ λειτουργίες CRUD (Create, Read, Update, Delete).

```
  public class PeopleController : Controller
  {
    private PatientHistoryEntities db = new PatientHistoryEntities();

    public ActionResult Index(string sortOrder, string currentFilter, string searchString, int? page, string
emptyPerson)
    {
      Session["SelectedPersonId"] = null;
      ViewBag.CurrentSort = sortOrder;
      ViewBag.LastNameSortParm = String.IsNullOrEmpty(sortOrder) ? "lastname_desc" : "";
      ViewBag.SexSortParm = sortOrder == "Sex" ? "sex_desc" : "Sex";
      ViewBag.emptyPerson = String.IsNullOrEmpty(emptyPerson) ? "false" : "Επέλεξε Ασθενή";

      if (searchString != null)
      {          page = 1;          }
      else
      {          searchString = currentFilter;          }

      ViewBag.CurrentFilter = searchString;
      var userName = User.Identity.Name;
      var persons = from s in db.People.Where(c => c.userCreator == userName)
                select s;
      //case-sensitive comparison
      if (!String.IsNullOrEmpty(searchString))
      {
        persons = persons.Where(s => s.Last_Name.ToUpper().Contains(searchString.ToUpper())
        || s.First_Name.ToUpper().Contains(searchString.ToUpper())).Where(c => c.userCreator == userName);
      }

      switch (sortOrder)
      {
```

```
            case "lastname_desc":
                persons = persons.OrderByDescending(s => s.Last_Name);
                break;
            case "Sex":
                persons = persons.OrderBy(s => s.Sex);
                break;
            case "sex_desc":
                persons = persons.OrderByDescending(s => s.Sex);
                break;
            default:
                persons = persons.OrderBy(s => s.Last_Name);
                break;
        }

        int pageSize = 10;
        int pageNumber = (page ?? 1);

        return View(persons.ToPagedList(pageNumber, pageSize));
    }


    public ActionResult UploadPhoto(string avatarImage,
    HttpPostedFileBase photo)
    {
        string path = @"C:\Users\ioannis\Documents\Visual Studio
2013\Projects\PatientHistory\PatientHistory\Content\avatar\"
        + avatarImage;

        if (photo != null)
            photo.SaveAs(path);

        return RedirectToAction("Edit", new RouteValueDictionary(new { controller = "People", action = "Edit",
Id = Session["SelectedPersonId"] }));
    }

    [HttpPost]
    [ValidateAntiForgeryToken]
    public ActionResult DeletePhoto(string photoFileName)
    {
        var photoName = "";
        photoName = photoFileName;
        string fullPath = Request.MapPath("~/Content/avatar/"
        + photoName);

        if (System.IO.File.Exists(fullPath))
        {
            System.IO.File.Delete(fullPath);
        }
        ViewBag.PersonId = Session["SelectedPersonId"];

        return RedirectToAction("Edit", new RouteValueDictionary(new { controller = "People", action = "Edit",
Id = Session["SelectedPersonId"] }));
    }}}
```

9.3 Καρτέλες Ασθενούς

Ο ιατρικός φάκελος του ασθενούς που πραγματοποιεί την επίσκεψη περιέχει πολλές πληροφορίες. Για την ομαδοποίηση όλων αυτών των στοιχείων δημιουργήθηκαν ξεχωριστές καρτέλες στις οποίες γίνεται μοναδικός χειρισμός για κάθε μία αντίστοιχη οντότητα.



Ο ιατρικός φάκελος για κάθε ασθενή, περιλαμβάνει τις αλλεργίες του, τις φαρμακευτικές του αγωγές, τις λεπτομέριες της κοινωνικής του ζωής, το ατομικό του αναμνηστικό με τα παλιότερα νοσήματα και χειρουργικές επεμβάσεις, το οικογενειακό του ιστορικό και τέλος το αρχείο των εξετάσεων σε ηλεκτρονική μορφή που μπορεί να υπάρχουν. Για την διαχείριση όλων αυτών των πληροφοριών έχουν δημουργηθεί διαδικασίες στους controllers της κάθε οντότητας με τις οποίες έχουν καθοριστεί οι βασικές λειτουργίες CRUD, έλεγχοι ορθότητας, συγκέντρωση στοιχείων για προβολή τους.

Αναλυτικά για την διαχείριση των εξετάσεων σε ηλεκτρονική μορφή και του οικογενιεακού ιστορικού που αναφέρεται στις σχέσεις συγγένειας πρώτου βαθμού, δημιουργήθηκαν επιπλέον μέθοδοι μέσα στον controller του Person, οι οποίες παρατίθενται παρακάτω.

```
public class PeopleController : Controller
```

```
  {
    private PatientHistoryEntities db = new PatientHistoryEntities();

    public ActionResult DetailsProtouVathmou(int? id)
    {
      if (id == null)
      {
        return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
      }
      Person person = db.People.Find(id);
      if (person == null)
      {
        return HttpNotFound();
      }
      Session["SelectedPersonId"] = id;

      //αναζητηση για πιθανοτητες εμφάνισης ασθενιών λόγω ιστορικού
      db.parentsANAIMIA();
      ViewBag.noticeAnaimia = db.Anaimias.Where(c => c.KidId ==
id).Select(x=>x.PososTOCHILDREN).SingleOrDefault();
      db.parentsDIAVITIS();
      ViewBag.noticeDiavitis = db.Viavhths.Where(c => c.KidId == id).Select(x =>
x.PososTOCHILDREN).SingleOrDefault();
      db.parentsOSTEOPORWSH();
      ViewBag.noticeOsteo = db.Osteoporwshes.Where(c => c.KidId == id).Select(x =>
x.PososTOCHILDREN).SingleOrDefault();
      db.parentsTHIREO();
      ViewBag.noticeThireo = db.Thireos.Where(c => c.KidId == id).Select(x =>
x.PososTOCHILDREN).SingleOrDefault();
      return View(person);
    }

    //create protu vathmu
    //CreateRelation
    public ActionResult CreateRelation(int? idPatient, int? idRelatype)
    {
      if ((idPatient == null) || (idRelatype == null))
      {
        return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
      }

      //ΒΡΙΣΚΩ ΤΟ ΠΡΩΤΟ ΜΕΛΟΣ
      var personPatient = db.People.Where(i => i.PersonID == idPatient).ToList();
      //ΒΡΙΣΚΩ ΤΗΝ ΜΕΤΑΞΥ ΤΟΥΣ ΣΧΕΣΗ
      var relatypeFamily = db.Relatypes.Where(i => i.RelatypeID == idRelatype).ToList();

      //PERSONOUT = PERSON_OB-PERSON_1B-PERSON_2B-PERSON_3B
      //ΠΡΟΒΟΛΗ ΣΤΗ ΛΙΣΤΑ ΜΕ ΤΙΣ ΕΠΙΛΟΓΕΣ ΓΙΑ ΤΟ ΔΕΥΤΕΡΟ ΜΕΛΟΣ ΜΟΝΟ
      //ALL_PERSONS-PERSONOUT

      //ΒΡΙΣΚΩ ΤΟ ΔΕΥΤΕΡΟ ΜΕΛΟΣ
      var person_0B = db.People.Where(i => i.PersonID == idPatient).Select(i => i.PersonID).ToList();
      //ΒΡΙΣΚΩ ΟΛΕΣ ΤΙΣ ΓΥΝΑΙΚΕΣ - ΑΝΔΡΕΣ
      var person_3B = new List<int>();
      var person_4A = new List<int>();
      if (idRelatype == 1)//ΜΗΤΕΡΑ
```

```
        {   //ΟΙ ΑΝΔΡΕΣ ΔΕΝ ΝΜΠΟΡΕΙ ΝΑ ΕΙΝΑΙ ΥΠΟΨΗΦΙΟΙ ΓΙΑ ΤΗ ΣΧΕΣΗ ΜΗΤΕΡΑΣ-ΑΔΕΛΦΗΣ
            person_3B = db.People.Where(i => i.Sex == "ΑΝΔΡΑΣ").Select(i => i.PersonID).ToList();
            //ΤΑ ΑΤΟΜΑ ΠΟΥ ΕΧΟΥΝ ΗΔΗ ΜΗΤΕΡΑ ΔΕΝ ΜΠΟΡΟΥΝ ΝΑ ΕΧΟΥΝ ΚΑΙ ΔΕΥΤΕΡΗ ΜΗΤΕΡΑ
            person_4A = db.Relationships.Where(t => t.RelatypeId == 1).Select(t => t.Person1Id).ToList();
        }
        else if (idRelatype == 2)//ΠΑΤΕΡΑΣ-ΑΔΕΛΦΟΣ
        {   //ΟΙ ΓΥΝΑΙΚΕΣ ΔΕΝ ΜΠΟΡΕΙ ΝΑ ΕΙΝΑΙ ΥΠΟΨΗΦΙΕΣ
            person_3B = db.People.Where(i => i.Sex == "ΓΥΝΑΙΚΑ").Select(i => i.PersonID).ToList();
            //ΤΑ ΑΤΟΜΑ ΠΟΥ ΕΧΟΥΝ ΗΔΗ ΠΑΤΕΡΑ ΔΕΝ ΜΠΟΡΟΥΝ ΝΑ ΕΧΟΥΝ ΚΑΙ ΔΕΥΤΕΡΟ ΠΑΤΕΡΑ
            person_4A = db.Relationships.Where(t => t.RelatypeId == 2).Select(t => t.Person1Id).ToList();
        }

        //ΒΡΙΣΚΩ ΤΑ ΑΤΟΜΑ ΜΕ ΤΑ ΟΠΟΙΑ ΤΟ ΠΡΩΤΟ ΜΕΛΟΣ ΕΧΕΙ ΗΔΗ ΜΙΑ ΣΧΕΣΗ
        //ΑΝ ΕΧΕΙ ΗΔΗ ΜΙΑ ΑΠΟ ΤΙς ΔΙΑΘΕΣΙΜΕΣ ΣΧΕΣΕΙΣ ΔΕΝ ΜΠΟΡΕΙ ΜΕ ΤΟ ΙΔΙΟ ΑΤΟΜΟ ΝΑ ΕΧΕΙ
ΚΑΙ ΚΑΠΟΙΑ ΑΛΛΗ
        var person_1B = db.Relationships.Include(c => c.Person).Where(c => c.Person1Id == idPatient).Select(c =>
c.Person2Id).ToList();
        var person_2B = db.Relationships.Include(c => c.Person1).Where(c => c.Person2Id == idPatient).Select(c
=> c.Person1Id).ToList();
        //ΕΝΩΣΗ ΟΛΩΝ ΤΩΝ ΕΠΙΜΕΡΟΥΣ ΛΙΣΤΩΝ ΠΟΥ ΕΧΩ ΣΥΓΚΕΝΡΩΣΕΙ
        var personsOUT = person_0B.Union(person_1B).Union(person_2B).Union(person_3B).ToList();

        var userName = User.Identity.Name;
        //EXCEPT ΤΗΣ ΕΝΩΣΗΣ
        var person_B = db.People.Where(s => !personsOUT.Contains(s.PersonID)).Where(c => c.userCreator ==
userName).ToList();

        if ((personPatient == null) || (relatypeFamily == null) || (person_B == null))
        {
            return HttpNotFound();
        }

        //Edit Controller
        //Person newFamily = db.People.Include(i => i.SituationsPersons).Where(i => i.PersonID == id).Single();

        //var newFamily = new Person();
        //newFamily.SituationsPersons = new List<SituationsPerson>();
        PopulateAssignedPastSituationData();

        ViewBag.Person2Id = new SelectList(person_B, "PersonID", "Full_Name_BirthDay");

        ViewBag.userName = userName;

        Session["SelectedPersonId"] = idPatient;
        Session["SelectedRelatype"] = idRelatype;
        return View();
    }


    public ActionResult CreateRelation([Bind(Include =
"PersonID,First_Name,Last_Name,Sex,Birth_Date,Birth_City,Town_Live,Adreess_Live,Phone,Mobile,Mail,Death
_Date,WhyDied,Country_Live,userCreator")] Person person,
        string[] selectedSituations)
    {
        if (ModelState.IsValid)
        {
```

```
            db.People.Add(person);
            db.SaveChanges();


            Relationship newFamily = new Relationship();

            if (Session["SelectedRelatype"].ToString() == "1")//mum
            {
                newFamily = new Relationship { Person1Id = (int)Session["SelectedPersonId"], Person2Id =
person.PersonID, RelatypeId = 1 };
            }
            if (Session["SelectedRelatype"].ToString() == "2")//dad
            {
                newFamily = new Relationship { Person1Id = (int)Session["SelectedPersonId"], Person2Id =
person.PersonID, RelatypeId = 2 };
            }
            if (Session["SelectedRelatype"].ToString() == "3")//kid
            {
                Person thePatient = db.People.Find((int)Session["SelectedPersonId"]);
                if (thePatient.Sex == "ΑΝΔΡΑΣ")
                {
                    newFamily = new Relationship { Person1Id = person.PersonID, Person2Id =
(int)Session["SelectedPersonId"], RelatypeId = 2 };
                }
                if (thePatient.Sex == "ΓΥΝΑΙΚΑ")
                {
                    newFamily = new Relationship { Person1Id = person.PersonID, Person2Id =
(int)Session["SelectedPersonId"], RelatypeId = 1 };
                }
            }

            //ΥΠΑΡΧΕΙ Η ΣΥΓΚΕΚΡΙΜΕΝΗ ΣΧΕΣΗ
            var relationshipExist = db.Relationships.Where(c => c.Person1Id == newFamily.Person1Id).Where(c =>
c.Person2Id == newFamily.Person2Id).Where(c => c.RelatypeId == newFamily.RelatypeId).FirstOrDefault();
            if (relationshipExist != null)
            {
                @ViewBag.InsertNewRelationship = "ΥΠΑΡΧΕΙ Η ΣΥΓΚΕΚΡΙΜΕΝΗ ΣΧΕΣΗ";
            }
            //ΥΠΑΡΧΕΙ Η ΑΝΤΙΘΕΤΗ ΣΧΕΣΗ
            var relationshipOpp = db.Relationships.Where(c => c.Person1Id == newFamily.Person2Id).Where(c =>
c.Person1Id == newFamily.Person2Id).Where(c => c.RelatypeId == newFamily.RelatypeId).FirstOrDefault();
            if (relationshipOpp != null)
            {
                @ViewBag.InsertNewRelationship = "ΥΠΑΡΧΕΙ Η ΑΝΤΙΘΕΤΗ ΣΧΕΣΗ";
            }
            //ΔΕΝ ΜΠΟΡΕΙ ΝΑ ΔΗΜΙΟΥΡΓΗΘΕΙ ΣΧΕΣΗ ΜΕ ΤΟ ΙΔΙΟ ΑΤΟΜΟ
            if (newFamily.Person1Id == newFamily.Person2Id)
            {
                @ViewBag.InsertNewRelationship = "ΔΕΝ ΜΠΟΡΕΙ ΝΑ ΔΗΜΙΟΥΡΓΗΘΕΙ ΣΧΕΣΗ ΜΕ ΤΟ ΙΔΙΟ
ΑΤΟΜΟ";
            }
            //ΔΕΝ ΜΠΟΡΕΙ ΑΝΔΡΑΣ ΝΑ ΕΙΝΑΙ ΜΗΤΕΡΑ ΣΕ ΑΥΤΗ ΤΗ ΣΧΕΣΗ
            //mhtera
            var person2WOMAN = db.People.Where(c => c.PersonID == newFamily.Person2Id).Where(c => c.Sex
== "ΑΝΔΡΑΣ").FirstOrDefault();
            if ((newFamily.RelatypeId == 1) && (person2WOMAN != null))
```

```
            {
                @ViewBag.InsertNewRelationship = "ΔΕΝ ΜΠΟΡΕΙ ΑΝΔΡΑΣ ΝΑ ΕΙΝΑΙ ΜΗΤΕΡΑ ΣΕ ΑΥΤΗ ΤΗ
ΣΧΕΣΗ";
            }
            //ΔΕΝ ΜΠΟΡΕΙ ΓΥΝΑΙΚΑ ΝΑ ΕΙΝΑΙ ΠΑΤΕΡΑΣ ΣΕ ΑΥΤΗ ΤΗ ΣΧΕΣΗ
            //pateras
            var person2MAN = db.People.Where(c => c.PersonID == newFamily.Person2Id).Where(c => c.Sex ==
"ΓΥΝΑΙΚΑ").FirstOrDefault();
            if ((newFamily.RelatypeId == 2) && (person2MAN != null))
            {
                @ViewBag.InsertNewRelationship = "ΔΕΝ ΜΠΟΡΕΙ ΓΥΝΑΙΚΑ ΝΑ ΕΙΝΑΙ ΠΑΤΕΡΑΣ ΣΕ ΑΥΤΗ
ΤΗ ΣΧΕΣΗ";
            }

            if (@ViewBag.InsertNewRelationship == null)
            {
                db.Relationships.Add(newFamily);
                db.SaveChanges();

                if (selectedSituations != null)
                {
                    SituationsPerson situations_perso = new SituationsPerson();

                    foreach (var situation in selectedSituations)
                    {
                        situations_perso.PastSituationId = int.Parse(situation);
                        situations_perso.PersonId = person.PersonID;
                        situations_perso.Notes = "-";
                        db.SituationsPersons.Add(situations_perso);
                        db.SaveChanges();
                    }
                }

                return RedirectToAction("DetailsProtouVathmou", new RouteValueDictionary(new { controller =
"People", action = "DetailsProtouVathmou", Id = Session["SelectedPersonId"] }));
            }
        }

        return View(person);
    }


    [HttpPost]
    public ActionResult Upload(HttpPostedFileBase file)
    {
        try
        {
            if (file.ContentLength > 0)
            {
                var fileName = Path.GetFileName(file.FileName);

                var directoryPerson = @"C:/Users/ioannis/Documents/Visual Studio
2013/Projects/PatientHistory/files/" + Session["SelectedPersonId"] + "/";
                if (!Directory.Exists(directoryPerson))
                    Directory.CreateDirectory(directoryPerson);
```

```
            var path = Path.Combine(directoryPerson, fileName);
            file.SaveAs(path);
        }
        ViewBag.PersonId = Session["SelectedPersonId"];
        return RedirectToAction("Downloads");
    }
    catch
    {
        ViewBag.PersonId = Session["SelectedPersonId"];
        return RedirectToAction("Downloads");
    }
}

public ActionResult Downloads()
{
    var directoryPerson = @"C:/Users/ioannis/Documents/Visual Studio 2013/Projects/PatientHistory/files/" +
Session["SelectedPersonId"] + "/";
    if (!Directory.Exists(directoryPerson))
        Directory.CreateDirectory(directoryPerson);

    var dir = new System.IO.DirectoryInfo(directoryPerson);
    System.IO.FileInfo[] fileNames = dir.GetFiles("*.*");
    List<string> items = new List<string>();

    foreach (var file in fileNames)
    {
        items.Add(file.Name);
    }
    ViewBag.PersonId = Session["SelectedPersonId"];
    return View(items);
}

public FileResult Download(string ImageName)
{
  var directoryPerson = @"C:/Users/ioannis/Documents/Visual Studio 2013/Projects/PatientHistory/files/" +
Session["SelectedPersonId"] + "/";
  return File(directoryPerson + ImageName, System.Net.Mime.MediaTypeNames.Application.Octet);
}
  }
```

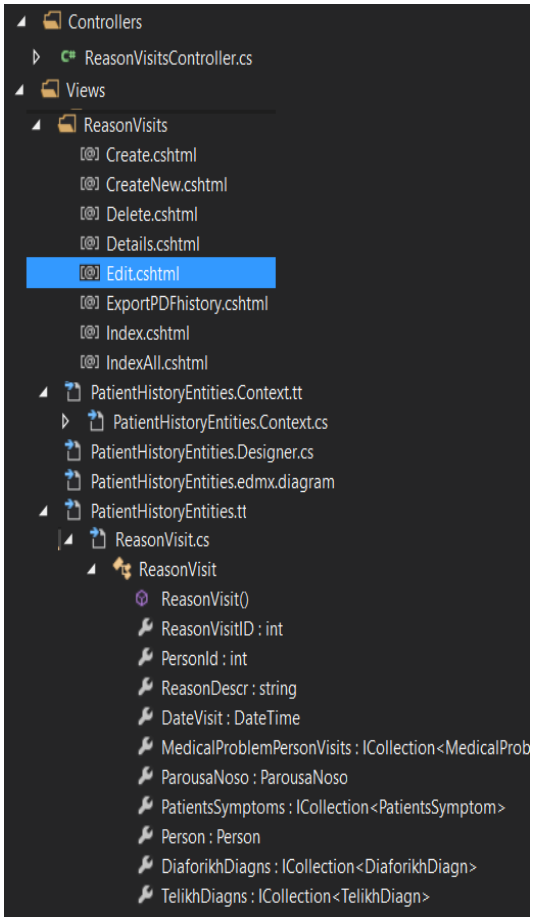
Ενδεικτικά ακολουθούν κάποιες μέθοδοι οι οποίοι είναι στους controller που έχουμε δημιουργήσει για τις υπόλοιπες οντότητες και πλαισιώνουν τον ιατρικό φάκελο.

```
public class AllergiesController : Controller
 {
    private PatientHistoryEntities db = new PatientHistoryEntities();

    //Delete
    public ActionResult Delete(int? id)
    {
        if (id == null)
        {
            return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
        }
        Allergy allergy = db.Allergies.Find(id);
```

```
        if (allergy == null)
        {
            return HttpNotFound();
        }
        return View(allergy);
                    }



    public class PharmakasController : Controller
    {
        private PatientHistoryEntities db = new PatientHistoryEntities();

        //Create
        public ActionResult Create(string emptyPerson)
        {
            var userName = User.Identity.Name;
            var persons = db.People.Where(c=>c.userCreator==userName).ToList();
            ViewBag.PersonId = new SelectList(persons, "PersonID", "Full_Name_BirthDay");

            ViewBag.emptyPerson = String.IsNullOrEmpty(emptyPerson) ? "false" : "Επέλεξε Ασθενή";
            return View();
        }

        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult Create([Bind(Include =
"PharmakoID,PersonId,NamePha,DosologiaPha,StartPha,ReasonPha,Pausi,Parenergeies")] Pharmaka pharmaka)
        {
            if (ModelState.IsValid)
            {
                db.Pharmakas.Add(pharmaka);
                db.SaveChanges();
                return RedirectToAction("Index", new { id = pharmaka.PersonId });
            }

            ViewBag.PersonId = pharmaka.PersonId;
            return View(pharmaka);
        }
                    }
```

9.4 Λίστα Επισκέψεων και Προσθήκη Νέας

Μετά την πρώτη καταγραφή των γενικότερων στοιχείων του ασθενούς και της ενημέρωσης των κατηγοριών του ιατρικού του φακέλου, ο γιατρός επικεντρώνεται στην συγκέντρωση πληροφοριών για το λόγο που πραγματοποιήθηκε η επίσκεψη. Για να γίνει αυτό, προσθέτει μία νέα εγγραφή επίσκεψης για τον ασθενή που έχει απέναντί του.



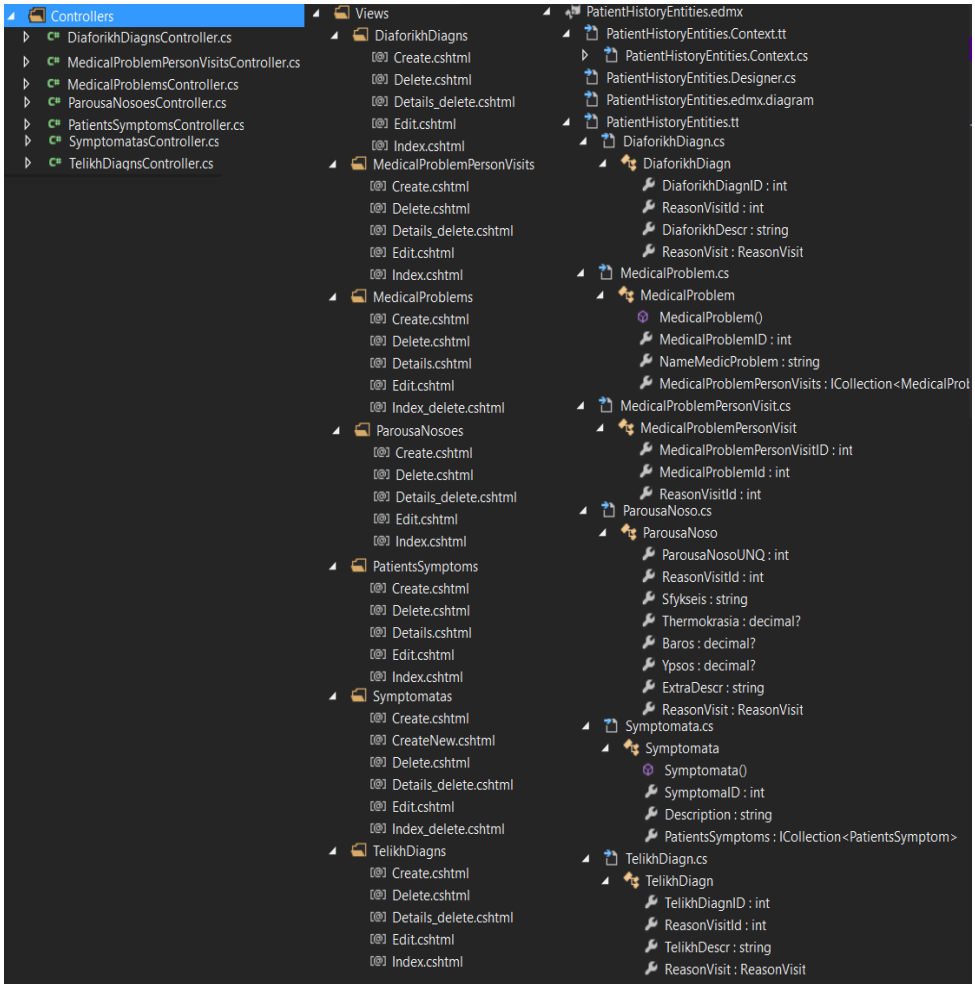
Η κλάση ReasonVisitsController.cs περιλαμβάνει τον απαιτούμενο κώδικα για την εκτέλεση των διαδικασιών για την οντότητα της επίσκεψης, όπως φαίνεται παρακάτω:

```
public class ReasonVisitsController : Controller
{
    private PatientHistoryEntities db = new PatientHistoryEntities();

    //GET
    public ActionResult IndexAll(string emptyPerson)
    {
        var reasonVisits = db.ReasonVisits.OrderByDescending(c=>c.DateVisit);
        ViewBag.emptyPerson = String.IsNullOrEmpty(emptyPerson) ? "false" : "Επέλεξε Ασθενή";
        return View(reasonVisits.ToList());
    }

    // GET eHISTORY For PDF
    [AllowAnonymous]
    public ActionResult ExportPDFhistory(int? id)
    {
        if (id == null)
        {
            return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
        }

        PrintVisit printVisit = new PrintVisit();
```

```
            printVisit.ReasonVisitPDF = db.ReasonVisits.Where(c=>c.ReasonVisitID==id);
            var personID = db.ReasonVisits.Where(c=>c.ReasonVisitID==id).Select(c=>c.PersonId).Single();
            printVisit.PersonPDF = db.People.Where(c => c.PersonID == personID).FirstOrDefault();

            ViewBag.AllergyPDF = db.Allergies.Where(c => c.PersonId == personID);
            ViewBag.PharmakaPDF = db.Pharmakas.Where(c => c.PersonId == personID);
            ViewBag.SurgeryPDF = db.Surgeries.Where(c => c.PersonId == personID);
            ViewBag.SituationPDF = db.SituationsPersons.Where(c => c.PersonId == personID);

            ViewBag.TelikhDiagnPDF = db.TelikhDiagns.Where(c=>c.ReasonVisitId==id);
            ViewBag.SymptomataPDF = db.PatientsSymptoms.Where(c=>c.ReasonVisitId==id);
            ViewBag.ParousaNosoPDF = db.ParousaNosoes.Where(c=>c.ReasonVisitId==id);
            ViewBag.DiaforikhDiagnPDF = db.DiaforikhDiagns.Where(c=>c.ReasonVisitId==id);
            ViewBag.ProblemsPDF = db.MedicalProblemPersonVisits.Where(c => c.ReasonVisitId == id);

            return View("ExportPDFhistory", "_LayoutPDF", printVisit);
        }

        [AllowAnonymous]
        public ActionResult UrlAsPDF(string Controller_Action_View)
        {
            return new Rotativa.UrlAsPdf(Controller_Action_View) { FileName =
Controller_Action_View.Replace("/", "_") + ".pdf" };   //"/ReasonVisits/ExportPDFhistory"
        }

        //Create
        public ActionResult CreateNew(string emptyPerson)
        {
            var userName = User.Identity.Name;
            var persons = db.People.Where(c => c.userCreator == userName).ToList();
            ViewBag.PersonId = new SelectList(persons, "PersonID", "Full_Name_BirthDay");

            ViewBag.emptyPerson = String.IsNullOrEmpty(emptyPerson) ? "false" : "Επέλεξε Ασθενή";
            return View();
        }

        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult CreateNew([Bind(Include = "ReasonVisitID,PersonId,ReasonDescr,DateVisit")]
ReasonVisit reasonVisit)
        {
            if (ModelState.IsValid)
            {
                db.ReasonVisits.Add(reasonVisit);
                db.SaveChanges();
                ViewBag.PersonId = reasonVisit.PersonId;
                ViewBag.ReasonVisitId = reasonVisit.ReasonVisitID;
                return RedirectToAction("Details", new RouteValueDictionary(new { controller = "ReasonVisits", action
= "Details", Id = reasonVisit.ReasonVisitID }));
            }

            ViewBag.PersonId = reasonVisit.PersonId;
            ViewBag.ReasonVisitId = reasonVisit.ReasonVisitID;
            return View(reasonVisit);
        }
       }
```

9.5 Καρτέλες Επίσκεψης

Οι πληροφορίες που χαρακτηρίζουν μία επίσκεψη μπορούν να ομαδοποιηθούν. Αυτό θα μας βοηθήσει να δημιουργήσουμε οντότητες οι οποίες αντιστοιχούν σε πίνακες στην ΒΔ και για την επεξεργασία τους οργανώθηκαν καρτέλες οι οποίες προβάλουν τα πεδία τους και μας επιτρέπουν με τις κατάλληλες λειτουργίες στους controllers να δημιουργήσουμε νέες, να τροποποιήσουμε τις υπάρχουσες και να διαγράψουμε.



Η κλάση ParousaNosoesController.cs περιλαμβάνει τον απαιτούμενο κώδικα για την εκτέλεση των διαδικασιών για την οντότητα που δημιουργήσαμε για τα χαρακτηριστικά της παρούσας νόσου της κάθε επίσκεψης, όπως φαίνεται παρακάτω:

```
public class ParousaNosoesController : Controller
{
    private PatientHistoryEntities db = new PatientHistoryEntities();
    // GET: ParousaNosoes/Details/5
    public ActionResult Details(int? id)
    {
        if (id == null)
```

```
        {
            return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
        }
        ParousaNoso parousaNoso = db.ParousaNosoes.Find(id);
        if (parousaNoso == null)
        {
            return HttpNotFound();
        }
        ViewBag.ReasonVisitId = parousaNoso.ReasonVisitId;
        ViewBag.PersonId = parousaNoso.ReasonVisit.PersonId;
        return View(parousaNoso);
    }
```

Η κλάση DiaforikhDiagnsController.cs περιλαμβάνει τον απαιτούμενο κώδικα για την εκτέλεση των διαδικασιών για την οντότητα που δημιουργήσαμε για τα χαρακτηριστικά της διαφορικής διάγνωσης της κάθε επίσκεψης, όπως φαίνεται παρακάτω:

```
public class DiaforikhDiagnsController : Controller
{
    private PatientHistoryEntities db = new PatientHistoryEntities();

    // GET: DiaforikhDiagns/Delete/5
    public ActionResult Delete(int? id)
    {
        if (id == null)
        {
            return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
        }
        DiaforikhDiagn diaforikhDiagn = db.DiaforikhDiagns.Find(id);
        if (diaforikhDiagn == null)
        {
            return HttpNotFound();
        }
        ViewBag.ReasonVisitId = diaforikhDiagn.ReasonVisitId;
        ViewBag.PersonId = diaforikhDiagn.ReasonVisit.PersonId;
        return View(diaforikhDiagn);
    }

    // POST: DiaforikhDiagns/Delete/5
    [HttpPost, ActionName("Delete")]
    [ValidateAntiForgeryToken]
    public ActionResult DeleteConfirmed(int id)
    {
        DiaforikhDiagn diaforikhDiagn = db.DiaforikhDiagns.Find(id);
        db.DiaforikhDiagns.Remove(diaforikhDiagn);
        db.SaveChanges();
        ViewBag.ReasonVisitId = diaforikhDiagn.ReasonVisitId;
        return RedirectToAction("Index", new RouteValueDictionary(new { controller = "DiaforikhDiagns", action
= "Index", Id = diaforikhDiagn.ReasonVisitId }));
    }

    protected override void Dispose(bool disposing)
    {
        if (disposing)
        {
            db.Dispose();
```

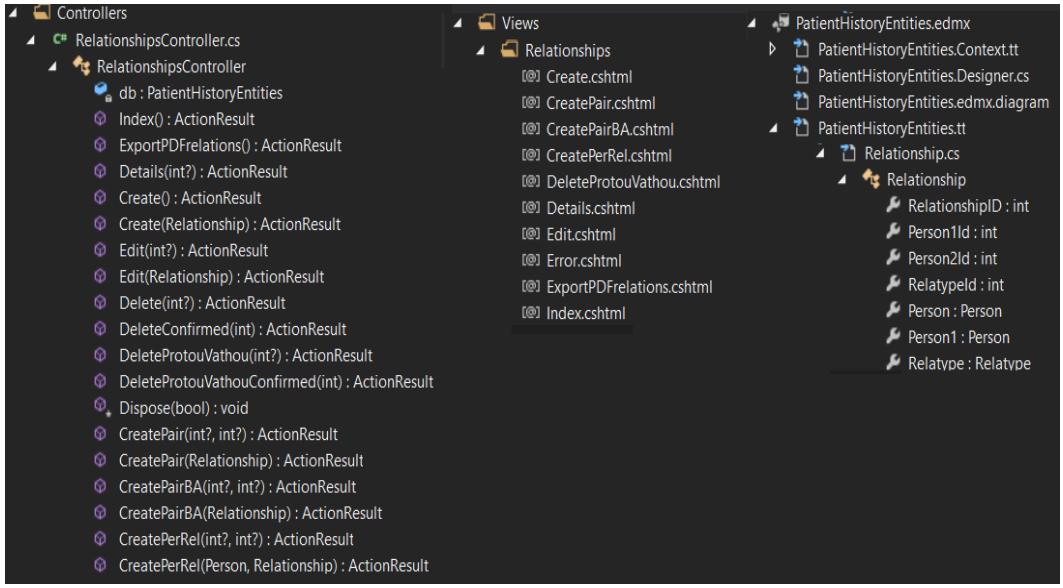
```
        }
        base.Dispose(disposing);
    }
}
```

Η κλάση PatientsSymptomsController.cs περιλαμβάνει τον απαιτούμενο κώδικα για την εκτέλεση των διαδικασιών για την οντότητα που δημιουργήσαμε για τα χαρακτηριστικά των συμπτωμάτων της κάθε επίσκεψης, όπως φαίνεται παρακάτω:

```
public class PatientsSymptomsController : Controller
{
    private PatientHistoryEntities db = new PatientHistoryEntities();

    public ActionResult Create(int? id)
    {
        ViewBag.ReasonVisitId = id;
        var patientsSymptomsPersonId = db.ReasonVisits.Where(c => c.ReasonVisitID == id).Select(t =>
t.PersonId).SingleOrDefault();
        ViewBag.PersonId = patientsSymptomsPersonId;
        ViewBag.SymptomaId = new SelectList(db.Symptomatas, "SymptomaID", "Description");
        Session["SelectedPersonId"] = patientsSymptomsPersonId;
        Session["SelectedReasonVisitId"]= id;
        return View();
    }
```

9.6 Σχέσεις Συγγένειας

Πιο συγκεκριμένα για το οικογενειακό ιστορικό, ο χειριστής γιατρός έχει την δυνατότητα να δημιουργήσει σχέσεις συγγένειας πρώτου βαθμού μεταξύ των καταχωρημένων ασθενών του ή και νέων ατόμων. Το τύπος αυτών των σχέσεων μπορεί να είναι οι γονείς και τα παιδιά κάπου. Για την διαχείριση της οντότητας που δημιουργήθηκε για αυτό το σκοπό δημιουργήθηκαν μέθοδοι στην κλάση του controller RelationshipsController και παρακάτω ενδεικτικά παρουσιάζεται ο κώδικας για την προβολή των λεπτομερειών μιας σχλεσης που περνάει ως όρισμα στην διαδικασία.



```
public class RelationshipsController : Controller
{
    private PatientHistoryEntities db = new PatientHistoryEntities();

    // GET: Relationship/Details/5
    public ActionResult Details(int? id)
    {
        if (id == null)
        {
            return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
        }
        Relationship relationship = db.Relationships.Find(id);
        if (relationship == null)
        {
            return HttpNotFound();
        }
        //αναζητηση για πιθανοτητες εμφάνισης ασθενιών λόγω ιστορικού
        db.parentsANAIMIA();
        ViewBag.noticeAnaimia = db.Anaimias.Where(c => c.KidId == relationship.Person1Id).Select(x =>
x.PososTOCHILDREN).SingleOrDefault();
        db.parentsDIAVITIS();
        ViewBag.noticeDiavitis = db.Viavhths.Where(c => c.KidId == relationship.Person1Id).Select(x =>
x.PososTOCHILDREN).SingleOrDefault();
        db.parentsOSTEOPORWSH();
        ViewBag.noticeOsteo = db.Osteoporwshes.Where(c => c.KidId == relationship.Person1Id).Select(x =>
x.PososTOCHILDREN).SingleOrDefault();
        db.parentsTHIREO();
        ViewBag.noticeThireo = db.Thireos.Where(c => c.KidId == relationship.Person1Id).Select(x =>
x.PososTOCHILDREN).SingleOrDefault();
        return View(relationship);
    }
```

9.7 Στατιστικά για τα Καταχωρημένα Άτομα

Οι ασθένειες οι οποίες μας απασχόλησαν είναι η Οστεοπόρωση, ο Σακχαρώδης Διαβήτης, η Θυρεοειδίτιδα Χασιμότο και η Μεσογειακή Αναιμία.

Αρχικά για κάθε ασθένεια συγκεντρώθηκαν πληροφορίες για πιθανότητες εμφάνισης σε παιδιά σθενών και καταγράφηκαν οι συνθήκες ικανοποίησης εφαρμογής της κληρονομικότητας. Στην συνέχεια δημιουργήθηκε Procedure για κάθε ασθένεια στον Sql Server, όπως προαναφέρθηκε και στην ανάλυση της ΒΔ. Οι τέσσερις που χρειάστηκαν είναι [parentsANAIMIA], [parentsDIAVITIS], [parentsOSTEOPORWSH], [parentsTHIREO] .

Σχολιάζοντας συγκεντρωτικά, με την διαδικασία εξετάζονται τα άτομα που έχουν συγγένειες πρώτου βαθμού και αν οι γονείς τους έχουν νοσήσει από τις προαναφερθήσες ασθένειες γίνεται κατάταξη του παιδιού στην αντίστοιχη κατηγορία με το ποσοστό εμφάνισης στο μέλλον της ασθένειας των γονέων. Τα αποτελέσματα συγκντρώνονται σε σχετικό πίνακα, από τον οποίο γίνεται και η τροφοδοσία των διαγραμμάτων απεικόνισης στατστικών.

Για τον σχεδιασμό διαγραμμάτων χρησιμοποιήθηκε η κλάση Chart από τους Helpers του System.Web.

Στους controller για κάθε ασθένεια υπάρχουν μέθοδοι οι οποίοι αφού συγκεντρώσουν τα συνολικά στοιχεία και τα επεξεργαστούν δημιουργούν ένα αντικείμενο τύπου chart το οποίο και το τροφοδοτούν με αυτά τα δεδομένα ώστε να προβληθεί από το view.

```
ArrayList xValue = new ArrayList();
ArrayList yValue = new ArrayList();
var data =
from personwtihdiavhth in db.AgePersonNoshmaViews.Where(c => c.SitDescr == "Σακχαρώδης Διαβήτης")
group personwtihdiavhth by personwtihdiavhth.Sex into Pososto CHILDRENGroup
select new { ViavhthsDescr = Pososto CHILDRENGroup.Key, ViavhthsSynolo =
Pososto CHILDRENGroup.Count() };

data.ToList().ForEach(dt => xValue.Add(dt.ViavhthsDescr));
data.ToList().ForEach(dt => yValue.Add(dt.ViavhthsSynolo));

Chart chart = new Chart(width: 600, height: 400, theme: ChartTheme.Yellow)
.AddTitle("Πλήθος Ανά Φύλο Ασθενών Με Σακχαρώδης Διαβήτης")
.AddSeries("ViavhthsDescr", chartType: "Pie", xValue: xValue, yValues: yValue);
chart.Write("bmp");
```

10 ΣΥΜΠΕΡΑΣΜΑΤΑ

10.1 Σύνοψη και Παρουσίαση Συμπερασμάτων

Ο Ηλεκτρονικός Ιατρικός Φάκελος Ασθενούς αποτελεί αντικείμενο ερευνών τόσο στην ιατρική όσο και στην πληροφορική. Ο σκοπός της παρούσας διπλωματικής εργασίας είναι να παρουσιάσει αρχικά με συνοπτικό τρόπο το θεωρητικό υπόβαθρο αναφορικά με την ιατρική αλλά και την τεχνολογική πλευρά του θέματος για την αποθήκευση και επεξεργασία ιατρικών δεδομένων και στη συνέχεια να γίνει υλοποίηση ενός συστήματος, το οποίο να δείχνει τη λειτουργία του Ηλεκτρονικού Ιατρικού Φακέλου Ασθενούς.

Η υλοποίηση αφορούσε ένα Πληροφοριακό Σύστημα Ηλεκτρονικού Ιατρικού Φάκελου Ασθενούς για ένα ιδιωτικό ιατρείο οποιασδήποτε ειδικότητας. Βασικός στόχος ήταν να εντοπιστούν τα κρίσιμα ζητήματα σε σχέση με το σχεδιασμό και την υλοποίηση ενός τέτοιου συστήματος, ώστε να διασφαλίζεται η λειτουργικότητα του. Είναι σαφές ότι το ζήτημα δεν είναι δυνατόν να εξαντληθεί μέσω της εργασίας μας.

Δόθηκε ιδιαίτερα περισσότερο βάση, στον τρόπο εκμετάλλευσης και υλοποίησης νέων τεχνολογιών και frameworks, αναδεικνύοντας τα οφέλη τους, τόσο από την μεριά του δημιουργού, όσο και από την οπτική γωνία του τελικού χρήστη.

Η διαγνωστική διερεύνηση των παθολογικών καταστάσεων του ασθενούς που επισκέφτεται τον γιατρό, περιλαμβάνει τη λήψη του ιστορικού, την αντικειμενική εξέταση του ασθενούς και τις εργαστηριακές εξετάσεις. Είναι γεγονός ότι το ιστορικό έχει βασική θέση, καθώς πιστεύεται ότι κατευθύνει στην ορθή διάγνωση στο 60 - 80% των περιπτώσεων.

Με τον Ηλεκτρονικό Ιατρικό Φάκελο Ασθενούς ο γιατρός μπορεί εύκολα να αποκτήσει πρόσβαση ανά πάσα στιγμή στα ιατρικά δεδομένα, να τα επεξεργαστεί για εξαγωγή σωστών συμπερασμάτων σε ότι αφορά την εικόνα των παρελθόντων ιατρικών εξετάσεων και των μελλοντικών ενεργειών του ασθενούς.

10.2 Μελλοντικές Επεκτάσεις

Ο ευρύτερος τομέας της Ιατρικής που ασχολείται η υπάρχουσα εφαρμογή, παρέχει πολλές δυνατότητες για επέκταση των λειτουργιών της. Κατά συνέπεια, υπάρχουν αρκετά ζητήματα που θα μπορούσαν να μελετηθούν ως μελλοντικές επεκτάσεις της. Ενδεικτικά αναφέρουμε τα παρακάτω:

- Η χρήση του ΑΜΚΑ ως πρωτεύων κλειδιού θα αντιμετώπιζε το πρόβλημα για την μοναδικότητα του ασθενούς, ώστε να αποφεύγονται οι διπλογραφίες και επίσης σε περίπτωση καθολικής εφαρμογής του συστήματος από όλες τις υπηρεσίες του ιατρικού συστήματος, θα βοηθούσε στην ορθή ταυτοποίηση του ασθενούς κάθε φορά.

Για ευκολία και για εξυπηρέτηση όλων, το ιδανικό θα ήταν η ύπαρξη μιας κάρτας για κάθε ασθενή, η οποία θα είναι μοναδική. Συγκεκριμένα, ο κάθε ασθενής θα διαθέτει μία κάρτα και έχοντας αυτή την κάρτα ο ίδιος ή ο γιατρός του θα μπορεί να έχει πρόσβαση στο ιατρικό ιστορικό του. Η πορεία του ασθενούς καταγράφεται και παρακολουθείται πλήρως, από την είσοδό του στο νοσοκομείο μέχρι και μετά την έξοδο του από αυτό. Ο ασθενής αντιμετωπίζεται με ενιαίο τρόπο σε όλη την έκταση του νοσοκομείου και ανεξάρτητα από τις επαφές του με τις διάφορες υπηρεσίες του νοσοκομείου. Ο ασθενής θα μπορεί να προσέρχεται στα φαρμακεία και να εκτελεί τη συνταγή του γιατρού μόνο με την προσωπική του κάρτα ασθενούς. Το φαρμακείο εκτελεί τη συνταγή και αυτόματα πιστώνεται το ποσό σε τραπεζικό του λογαριασμό, ενώ ο ασθενής πληρώνει τη συμμετοχή του.

- Η εφαρμογή ενιαίων κωδικοποιήσεων (π.χ. κωδικοποίηση διαγνώσεων, κωδικοποίηση φαρμάκων κλπ.) στα δεδομένα που καταγράφονται ώστε να υπάρχει σαφής καθορισμός των στοιχείων και των διαγνώσεων. Με αυτήν την τυποποίηση θα εξυπηρετείτε η ανταλλαγή δεδομένων μεταξύ όλων των εμπλεκομένων.
- Μία από τις πιο βασικές προσθήκες λειτουργίας στην εφαρμογή θα είναι να επιτρέπει στον ασθενή, εφόσον το επιθυμεί, να προσθέτει εγγραφές στο Ιστορικό του, να καταγράψει τα συμπτώματα που έχει ή την αντίδραση του οργανισμού του στη θεραπεία που ακολουθεί, ή οτιδήποτε άλλο πιστεύει ότι πρέπει να γνωρίζει ο θεράπων γιατρός ή η ιατρική ομάδα που τον παρακολουθεί. Με την δυνατότητα αυτή, θα μπορεί ο ασθενής να επικοινωνεί «εγγράφως» με τον γιατρό του, και ο γιατρός να τροποποιεί άμεσα, αν χρειασθεί, την θεραπευτική αγωγή, εκδίδοντας νέα συνταγή ή παραπεμπτικά για διαγνωστικές εξετάσεις. Η ανωτέρω δυνατότητα είναι σημαντική διότι ο ασθενής δεν «συμμετέχει», και στην πράξη δεν θα υπάρχει πληροφόρηση στο φάκελο για τον «άνθρωπο», ποιο είναι σύμφωνα με τον ίδιο το πρόβλημα υγείας που αντιμετωπίζει, ποια είναι η γνώμη του για την αποτελεσματικότητα της θεραπείας, τις τυχόν παρενέργειες, κ.ά. Στη φόρμα της εφαρμογής μπορούν να προβλεφθούν fields στα οποία ο ασθενής θα μπορεί, εφόσον έχει τις γνώσεις και δεξιότητες να συμπληρώνει τις πληροφορίες που επιθυμεί να γνωρίζει η ιατρική του ομάδα ή σε αντίθετη περίπτωση θα πρέπει να τα συμπληρώνει ο γιατρός αφού φροντίσει με τις κατάλληλες ερωτήσεις να πάρει τις σχετικές πληροφορίες δηλ. θα πρέπει πάντοτε στο ιατρικό ιστορικό να συμμετέχει ο ασθενής και να είναι καταγεγραμμένες οι απόψεις του και το πρόβλημα που εκείνος δηλώνει ότι αντιμετωπίζει.
- Αναφορικά με την συνταγογράφηση φαρμάκων, σε ένα κεντρικό σύστημα ηλεκτρονικής καταγραφής, θα πετύχαινε να υπάρχει πλήρης αδιαφάνεια στο τι συνταγογραφείται από το ιατρικό προσωπικό. Η απουσία ελέγχου οδηγεί σε υπερσυνταγογράφηση φαρμάκων που συχνά βλάπτει τη δημόσια υγεία και δημιουργεί εξαιρετικά υψηλά έξοδα υγείας για τα ασφαλιστικά ταμεία.
- Θα μπορούσε να εγκατασταθεί ένας mail server ώστε να χρησιμοποιείται για την ταυτοποίηση του λογαριασμού mail που δηλώνει κατά την αρχική του εγγραφή ο χρήστης με την αποστολή ενός μηνύματος επιβεβαίωσης και επιπλέον θα ήταν χρήσιμο και κατά την διαδικασία ανάκτησης των διαπιστευτηρίων του.

Λαμβάνοντας υπόψη όλα τα παραπάνω η διπλωματική εργασία στόχευε να δείξει ένα μικρό κομμάτι της τεχνολογίας που μπορούμε να χρησιμοποιήσουμε για την παρακολούθηση της υγείας του ατόμου. Τέλος, βαθύτερος στόχος αυτής της διπλωματικής είναι να αποτελέσει τον κινητήριο μοχλό έμπνευσης για την υλοποίηση ενός τέτοιου συστήματος και με τις απαραίτητες προσθήκες επιπλέον λειτουργιών, να εκσυγχρονιστεί η Ιατρική Φροντίδα του Ατόμου.

11 ΕΓΚΑΤΑΣΤΑΣΗ ΣΥΣΤΗΜΑΤΟΣ

Το πληροφοριακό σύστημα που αναπτύχθηκε μπορεί να εγκατασταθεί και να λειτουργήσει σε ένα υπολογιστή με τις παρακάτω τουλάχιστον απαιτήσεις:

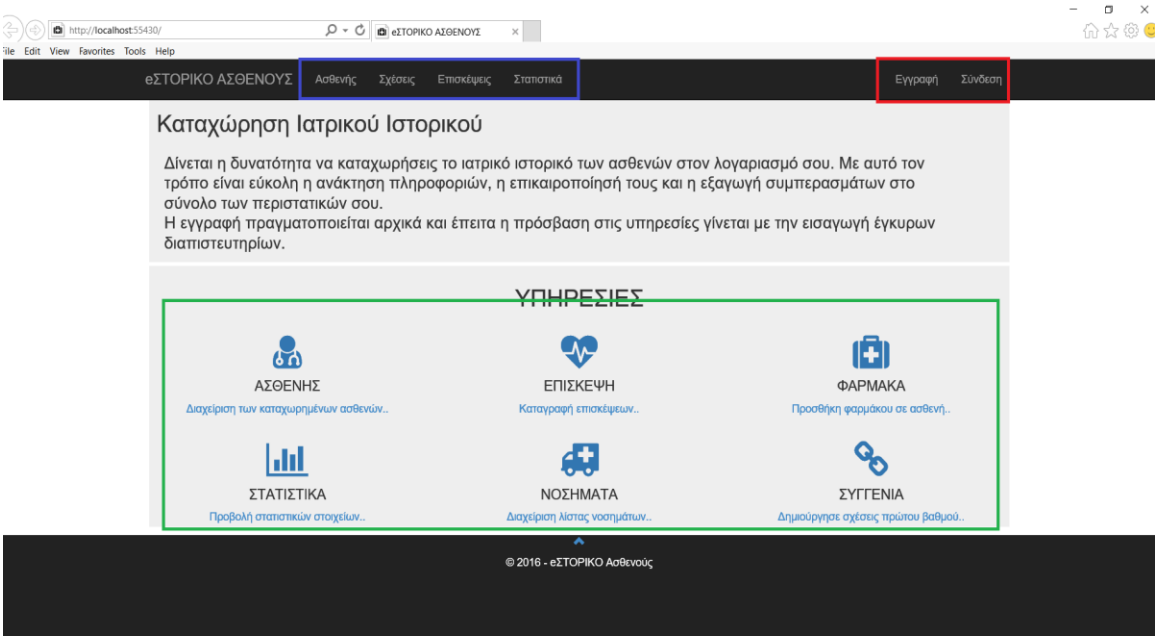
- Για λειτουργικό σύστημα θα χρειαστεί Microsoft Windows 7 ή μεταγενέστερες εκδόσεις.
- Ως περιβάλλον ανάπτυξης λογισμικού εγκατάστασης το Microsoft Visual Studio Έκδοση Professional 2013 με εγκατεστημένο το .Net Framework 4.5 ή μεταγενέστερες εκδόσεις.
- Για την βάση δεδομένων δημιουργία ένας Microsoft SQL Server 2014.
- Πρέπει να τονιστεί ότι για την δημοσίευση της εφαρμογής χρησιμοποιήθηκε ο IIS Express του Visual Studio και το URL που πρέπει να χτυπήσει ο χειριστής είναι το http://localhost:55430/ (η πόρτα 55430 είναι η προεπιλεγμένη από το VS).
- Αφού πρώτα δημιουργήσουμε την βάση δεδομένων μας στον SQL Server τρέχοντας το αρχείο patienthistory.sql που περιέχει όλον τον sql κώδικα, στη συνέχεια απαιτείται το άνοιγμα από το φάκελο του project μας του αρχείου patienthistory.sln. Στο αρχείο web.congif ορίζουμε το connection string ώστε να επικοινωνεί η εφαρμογή με την βάση δεδομένων μας και πλέον είμαστε έτοιμοι να τρέξουμε το project μας, localhost με την βοήθεια του IIS Express.

Η web εφαρμογή μας είναι τώρα διαθέσιμη και οποιοσδήποτε μπορεί να χτυπήσει τον web server μας μέσω ενός browser, μπορεί να δει την αρχική μας σελίδα και να κάνει χρήση των λειτουργιών της.

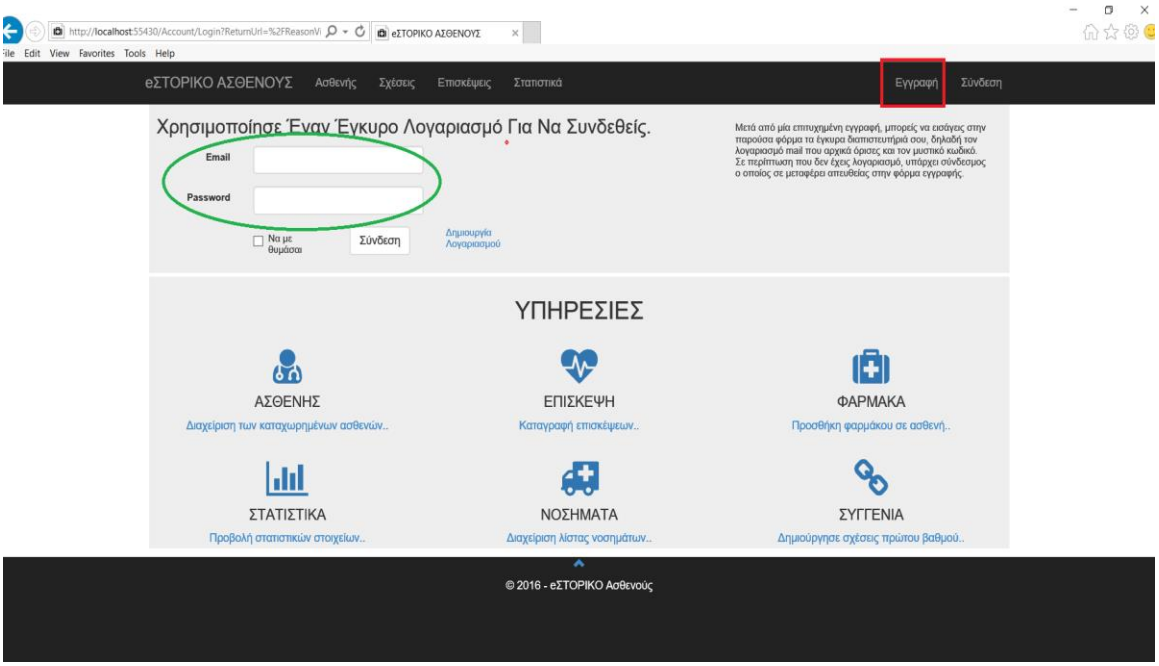
12 ΕΓΧΕΙΡΙΔΙΟ ΧΡΗΣΗΣ ΕΦΑΡΜΟΓΗΣ

12.1 Σύνδεση Χειριστή

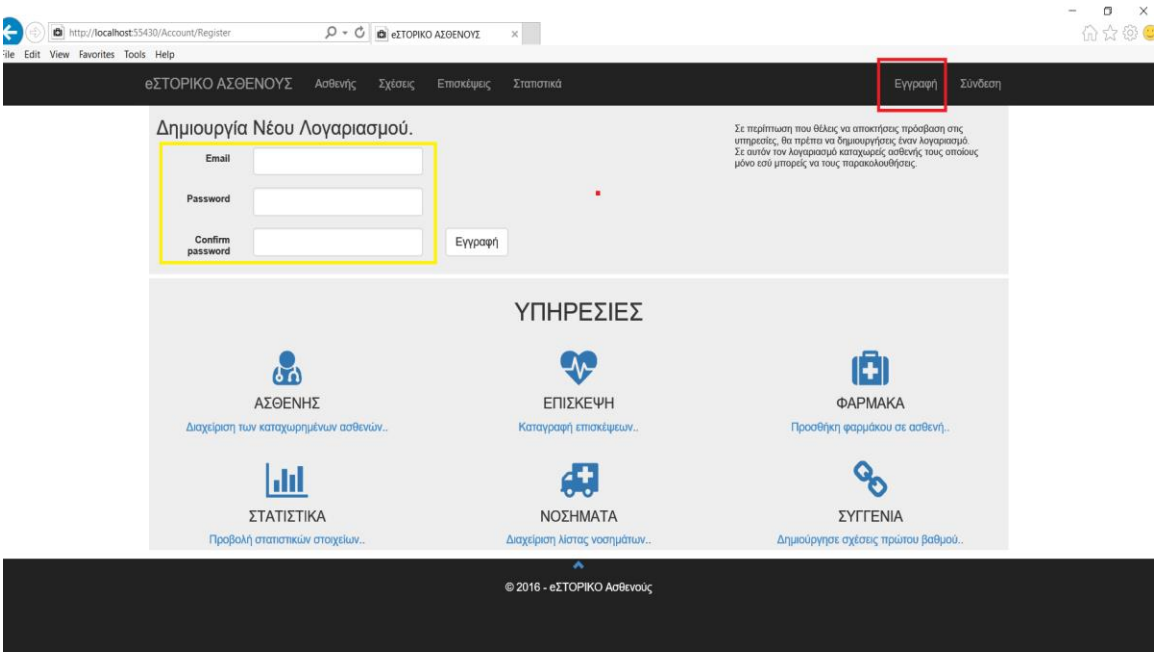
Παρακάτω φαίνεται η αρχική οθόνη που συναντά ο χειριστής αφού πρώτα πληκτρολογήσει στο φυλλομετρητή την διεύθυνση της εφαρμογής μας. Στην πάνω οριζόντια μπάρα εμφανίζονται οι επιλογές του μενού, οι οποίες είναι επιλέξιμες μόνο μετά από επιτυχημένη είσοδο των διαπιστευτηρίων.



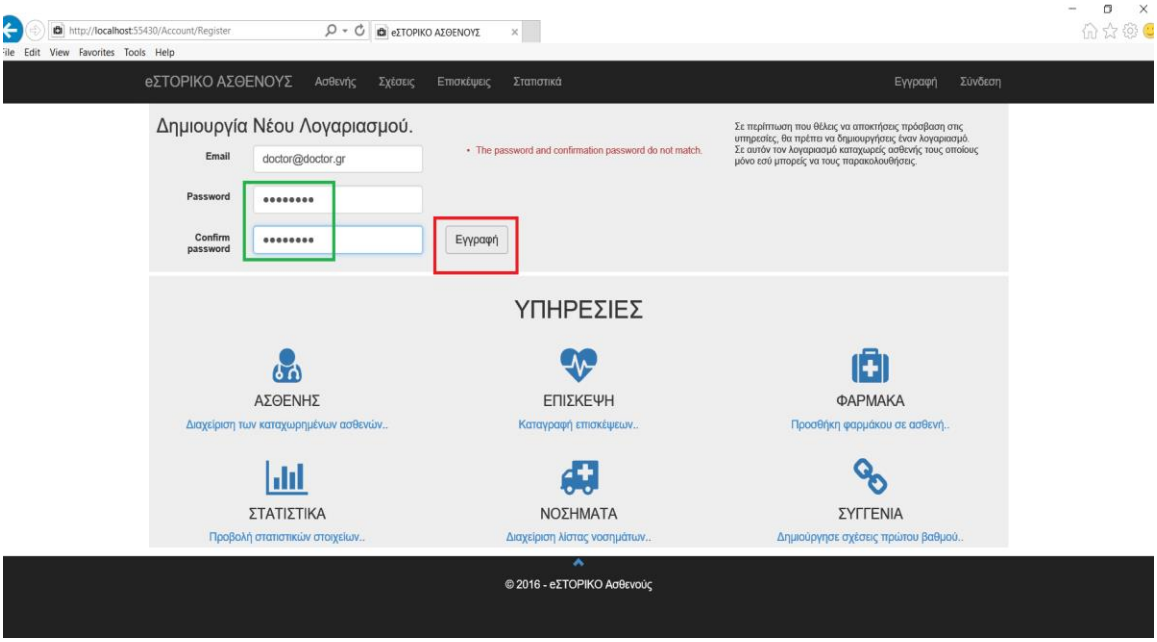
Για να αποκτηθεί πρόσβαση στις δυνατότητες της εφαρμογής θα πρέπει ο χειριστής που στην περίπτωσή μας είναι γιατρός να δημιουργήσει αρχικά έναν έγκυρο λογαριασμό. Για το λόγο αυτό επιλέγει από την πάνω οριζόντια μπάρα, το κουμπί «Εγγραφή».



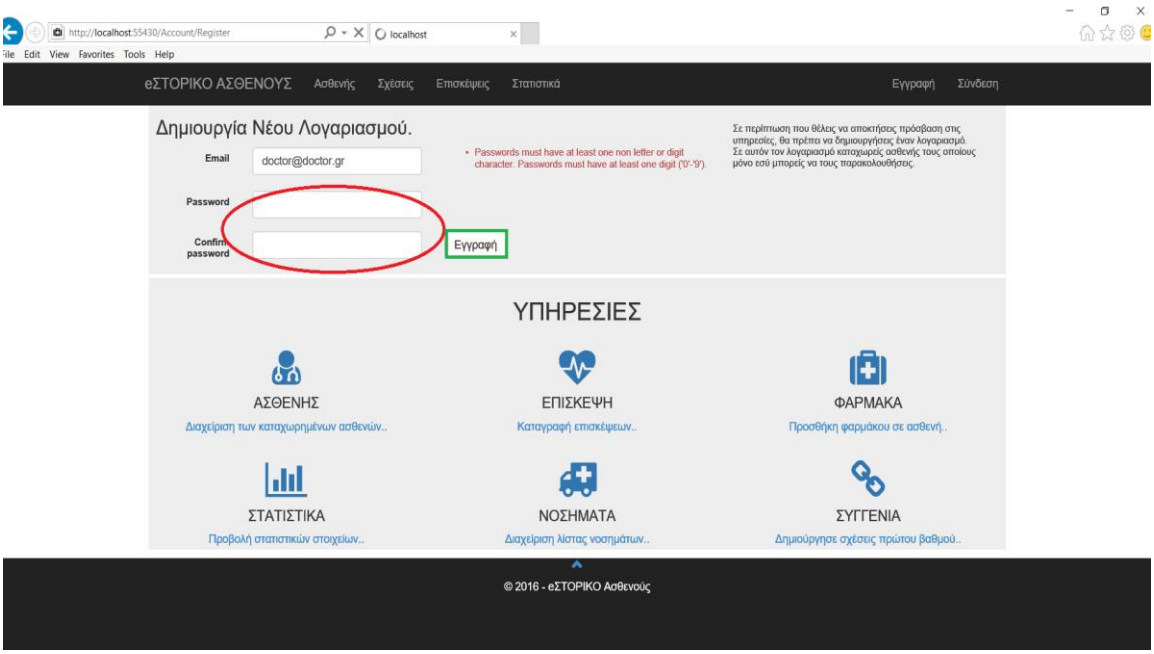
Στην συνέχεια που εμφανίζεται μία φόρμα για να συμπληρώσει τα απαραίτητα στοιχεία που απαιτούνται για την δημιουργία ενός νέου λογαριασμού εισόδου στην εφαρμογή. Ο λογαριασμός mail που θα δηλωθεί, θα χρησιμοποιηθεί ως όνομα χρήστη στην εφαρμογή και θα πρέπει να είναι έγκυρο, δηλαδή να ικανοποιεί τις συνθήκες λογικής σύνταξης ενός λογαριασμού email.



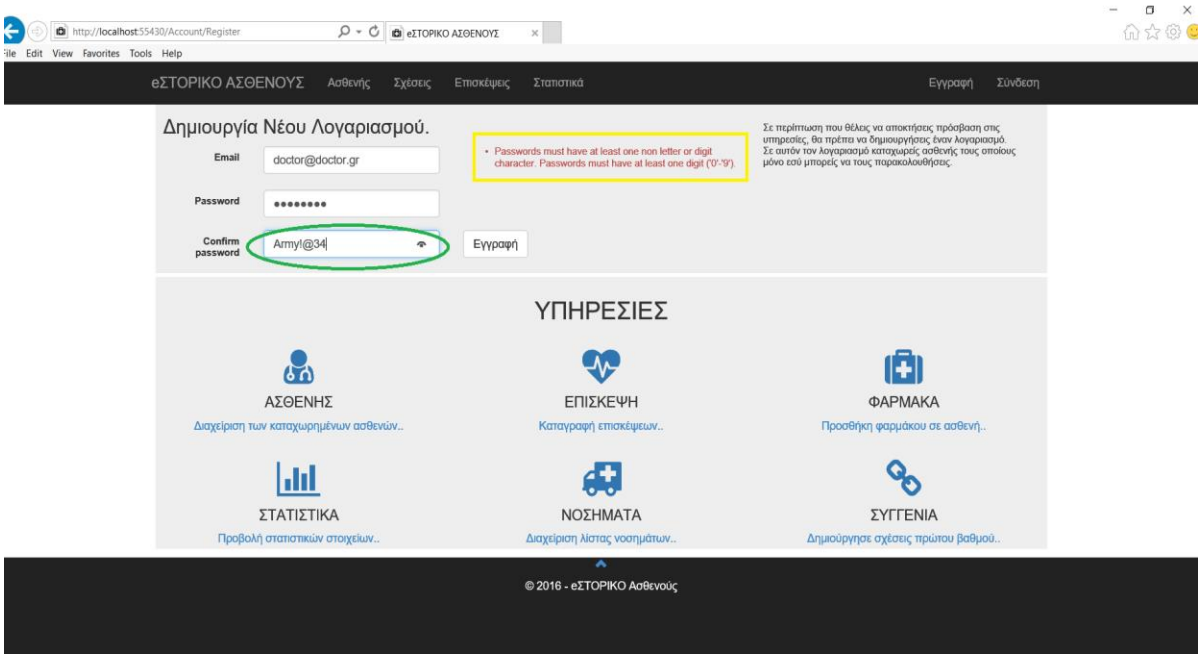
Πατώντας το κουμπί για την εγγραφή του νέου χρήστη, πραγματοποιείται έλεγχος μεταξύ των δύο τελευταίων πεδίων, ώστε να επιβεβαιωθεί ο δηλωμένος κωδικός. Στην περίπτωση που υπάρχει ασυμφωνία εμφανίζεται μήνυμα λάθους.



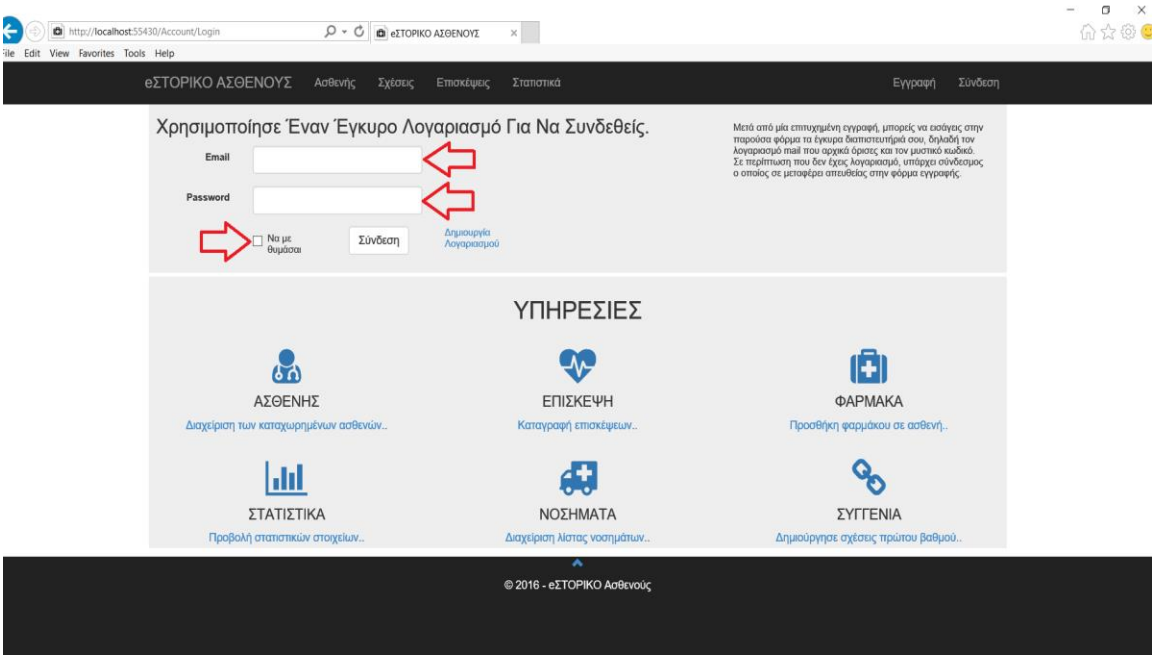
Επιπλέον πολύ σημαντικό είναι και κατά πόσο η ισχύς του κωδικού είναι δυνατή. Θα πρέπει να υπάρχει ελάχιστο μήκος 8 και ο συνδυασμός να περιλαμβάνει μικρούς, κεφαλαίους χαρακτήρες, αριθμούς και σύμβολα για να θεωρεί έγκυρος.



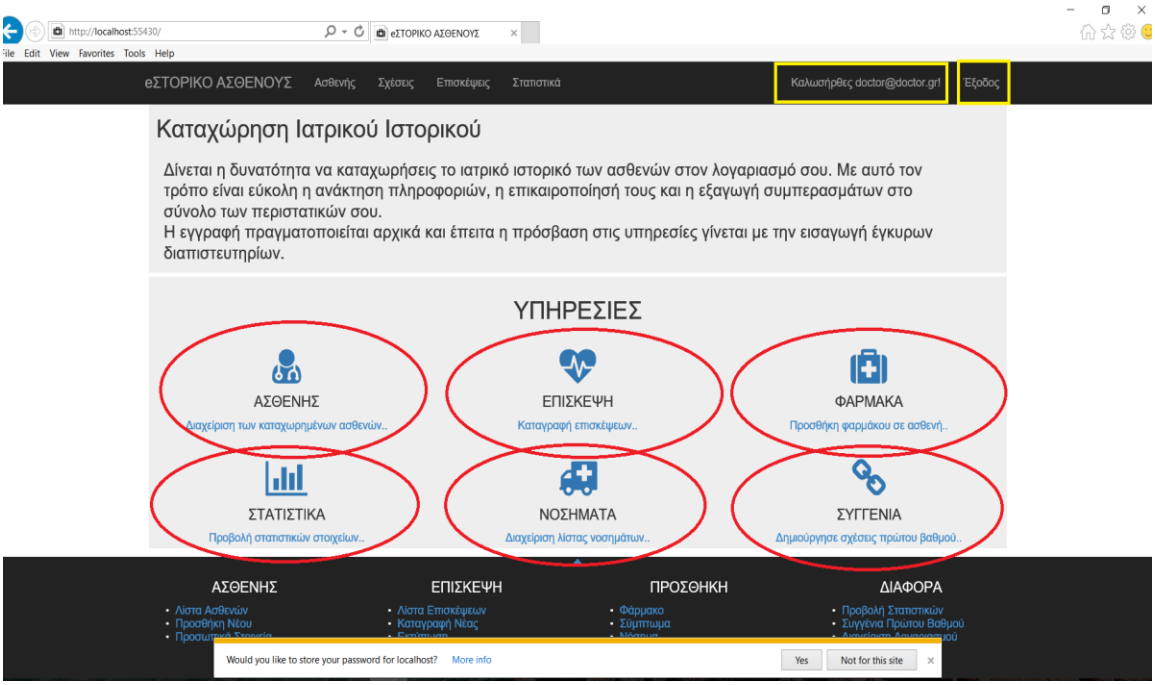
Παρακάτω εμφανίζονται τα διαπιστευτήρια που δημιουργήθηκαν για την απεικονίσει της παρουσίασης. Όπως βλέπουμε το email είναι doctor@doctor.gr και κωδικός ο Army!@34. Με βάση αυτόν τον λογαριασμό, θα καταχωρήσουμε ασθενείς που μας έχουν επισκεφτεί.



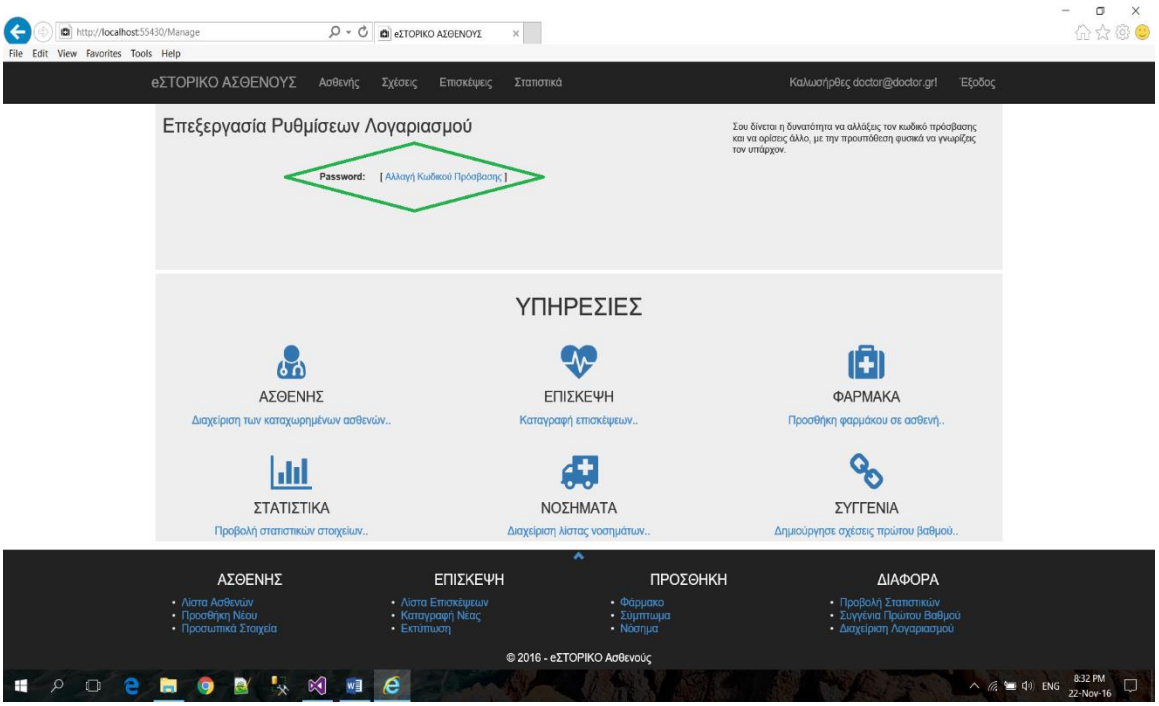
Επιλέγοντας πάνω δεξιά «ΣΥΝΔΕΣΗ» ή κάνοντας κλικ σε κάποια από τις επιλογές που εμφανίζονται μας ζητείται να εισάγουμε τα διαπιστευτήριά μας. Επιπλέον μας δίνεται η δυνατότητα να τικάρουμε την επιλογή «να με θυμάσαι», ώστε να διατηρηθούν τα στοιχεία εισόδου μας.



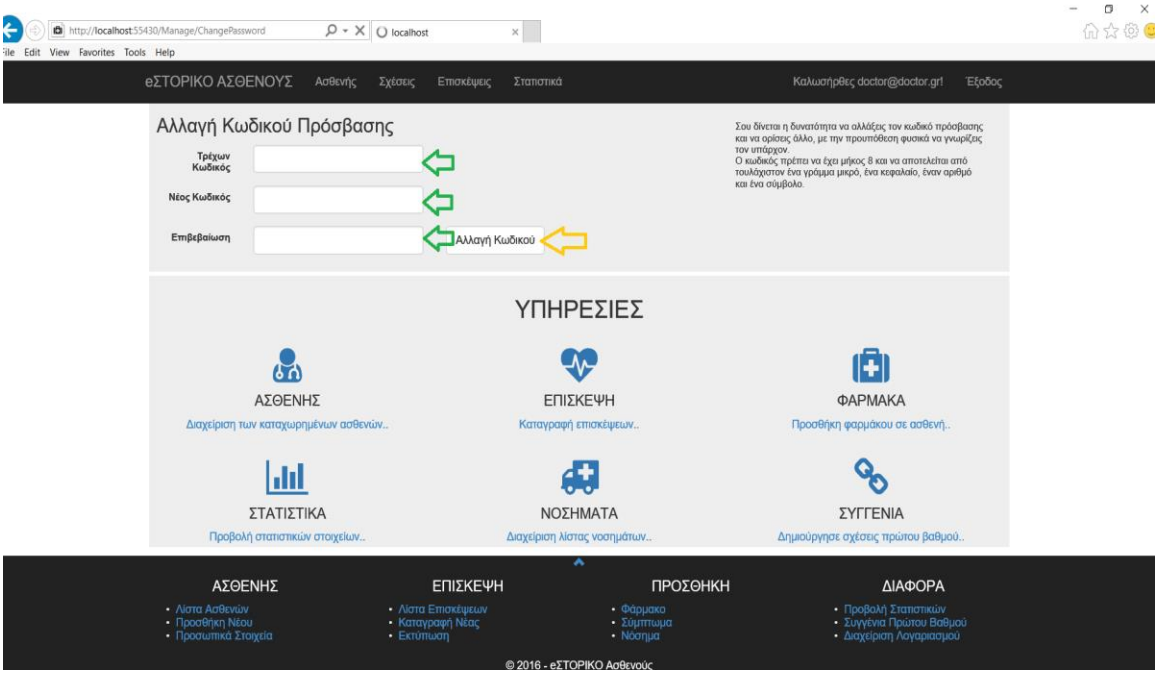
Όπως παρατηρούμε, αφού εισέλθουμε στην εφαρμογή εμφανίζεται μήνυμα καλωσορίσματος με το όνομα χρήστη που έχουμε που είναι ο λογαριασμός Email όπως προαναφέραμε και πάνω δεξιά έχουμε την δυνατότητα να κάνουμε αποσύνδεση πατώντας στο «ΕΞΟΔΟΣ».



Πατώντας πάνω στο μήνυμα καλωσορίσματος, εμφανίζεται η επιλογή ώστε να αλλάξουμε τον κωδικό πρόσβασης αν το επιθυμούμε.

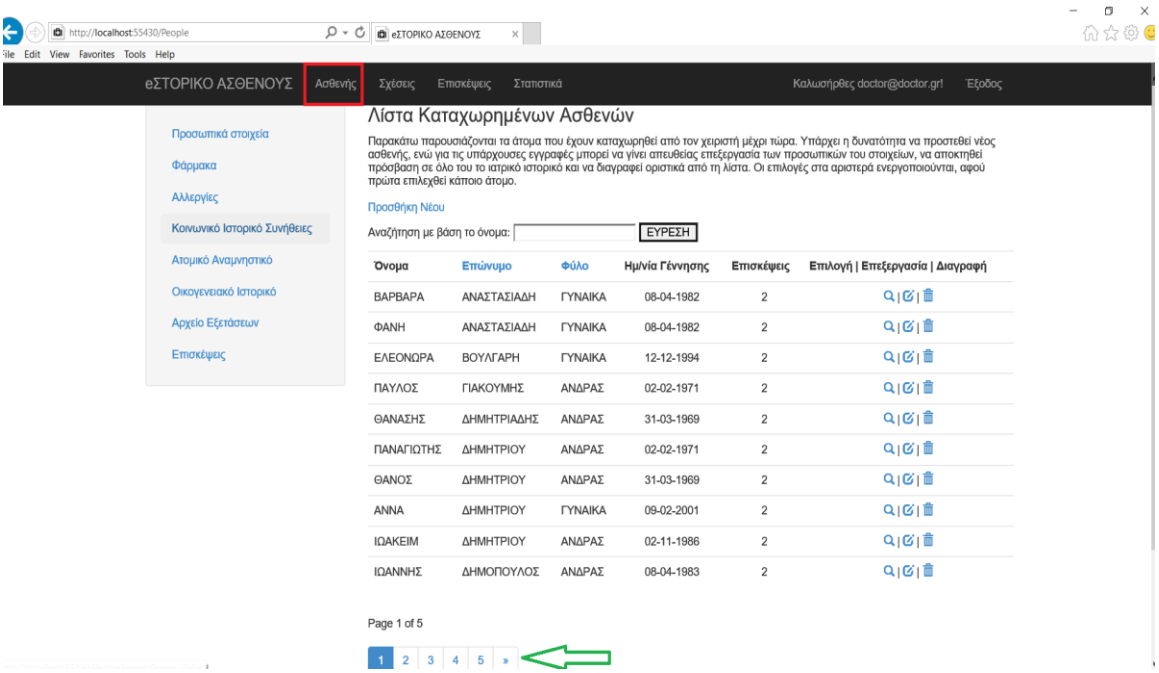


Για λόγους ασφαλείας μας ζητείται να ξαναγράψουμε και τον τρέχον κωδικό και στην συνέχεια τον νέο που επιθυμούμε πληκτρολογώντας τον δύο φορές για επιβεβαίωση.

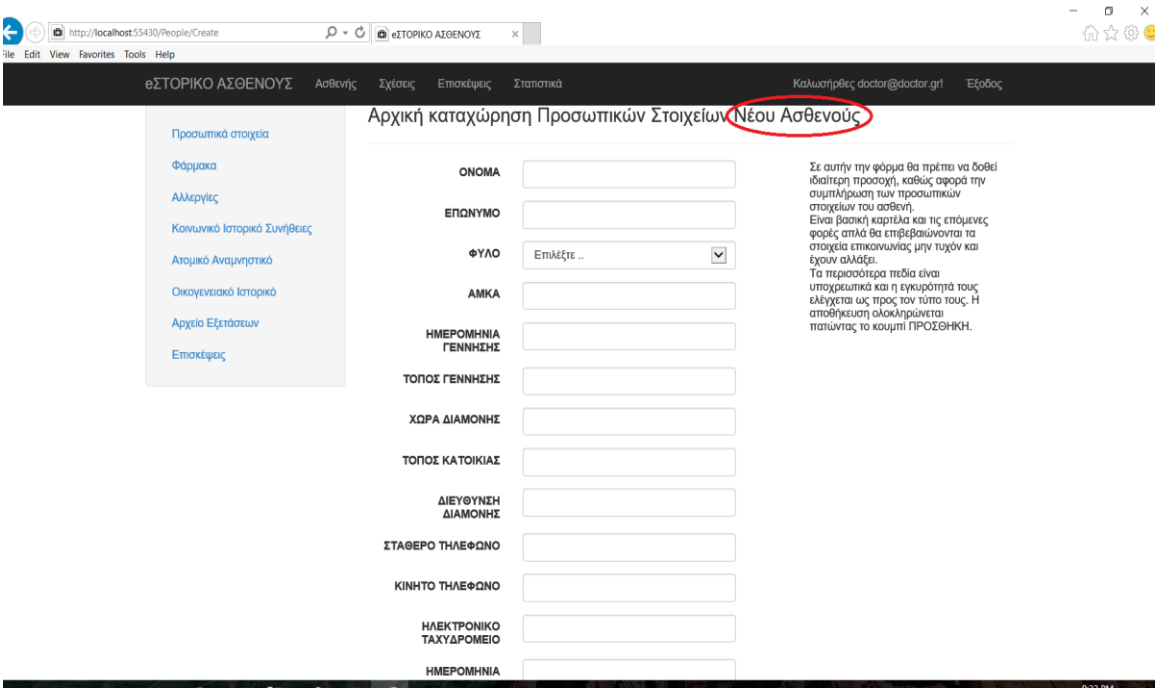


12.2 Λίστα Ασθενών και Καταχώρηση Νέου

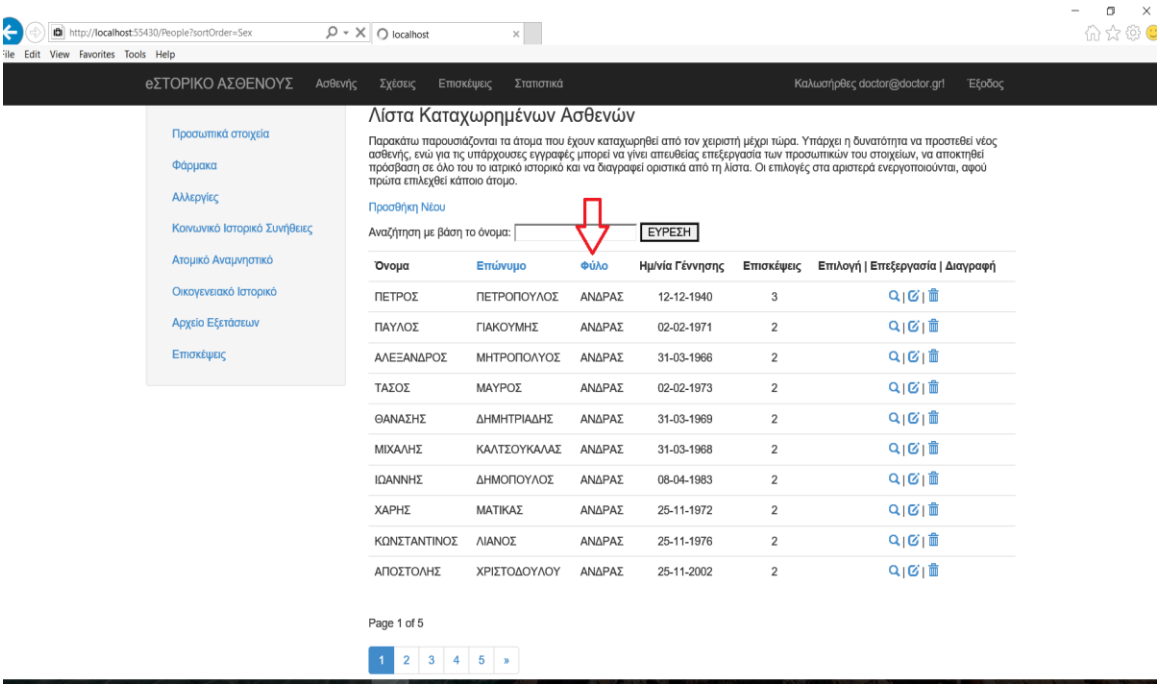
Όταν επιλέξει ο γιατρός που έχει ήδη κάνει επιτυχημένη είσοδο, το «ΑΣΘΕΝΗΣ» εμφανίζονται όλα τα άτομα που έχει καταχωρήσει. Αναλυτικά σε μία λίστα προβάλλεται το όνομα, το επώνυμο, το φύλο, η ημερομηνία γέννησης, ο αριθμός των επισκέψεων που έχει πραγματοποιήσει και ένα μενού με διαθέσιμες λειτουργίες για κάθε εγγραφή ατόμου. Για ευκολία στην προβολή έχουν δημιουργηθεί σελίδες των 10 ατόμων.



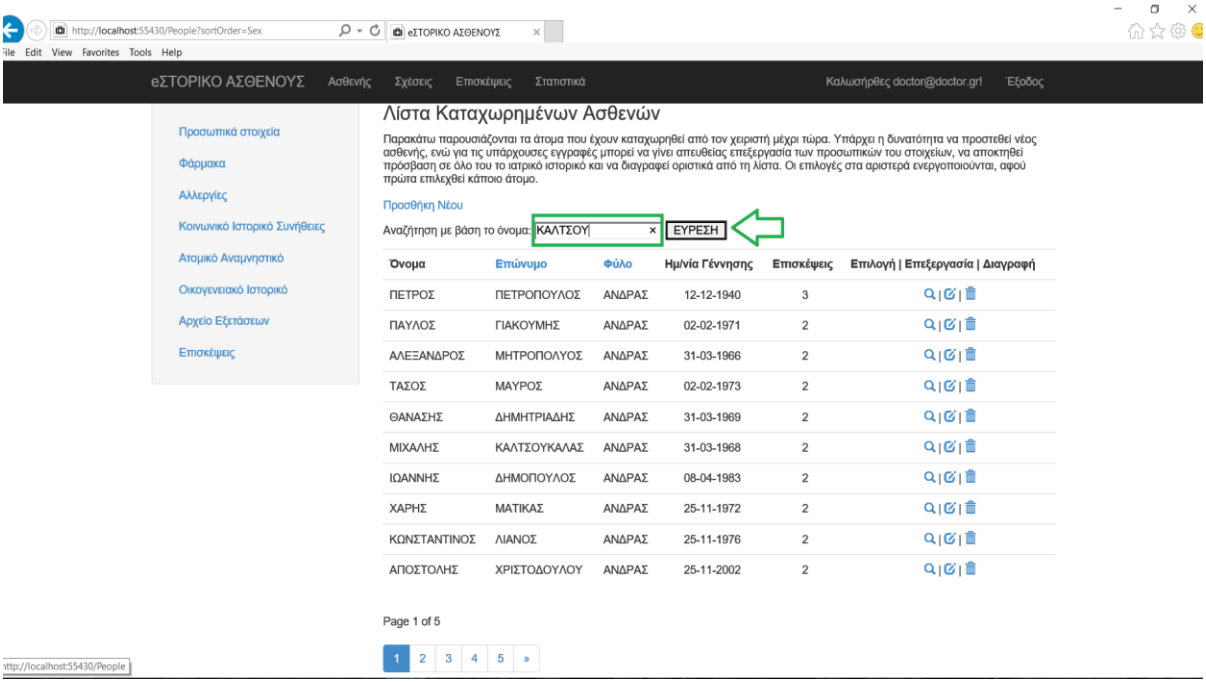
Ο γιατρός μπορεί να προσθέσει νέο άτομο συμπληρώνοντας την φόρμα με τα προσωπικά δημογραφικά στοιχεία του ασθενούς.

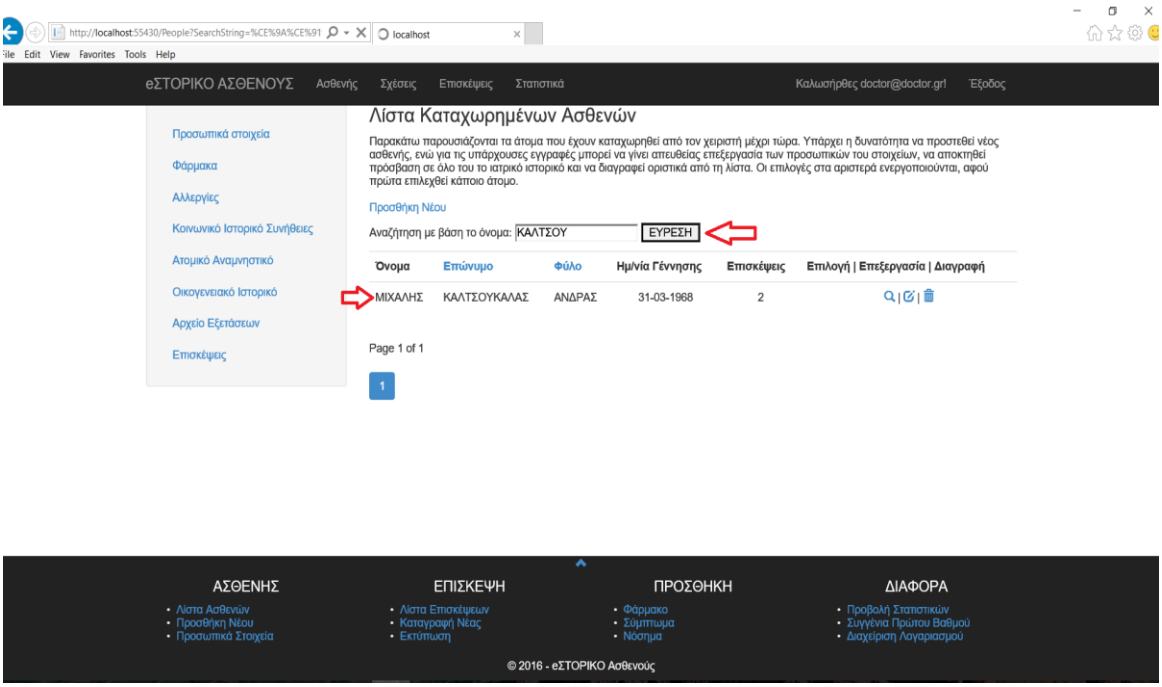


Επιπλέον προτέθηκαν και δύο φίλτρα ταξινόμησης των δεδομένων που προβάλλονται. Το πρώτο είναι αλφαβητικά με βάση το Επώνυμο και το δεύτερο κατά φύλο.

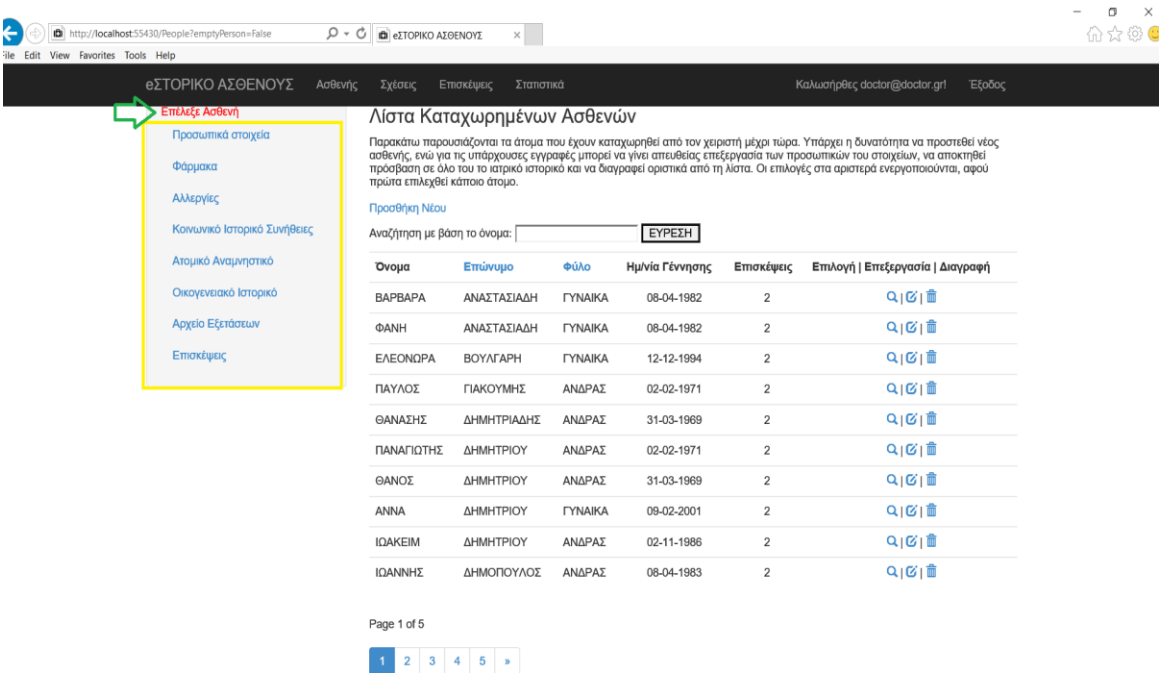


Λόγω του πλήθους των ατόμων και για ταχύτητα στην αναζήτηση απευθείας του ατόμου που μας ενδιαφέρει, υπάρχει και ένα πεδίο αναζήτησης που με κάποιο μέρος του ονόματος ή του επωνύμου μας εμφανίζει στη λίστα μόνο τα σχετικά άτομα.

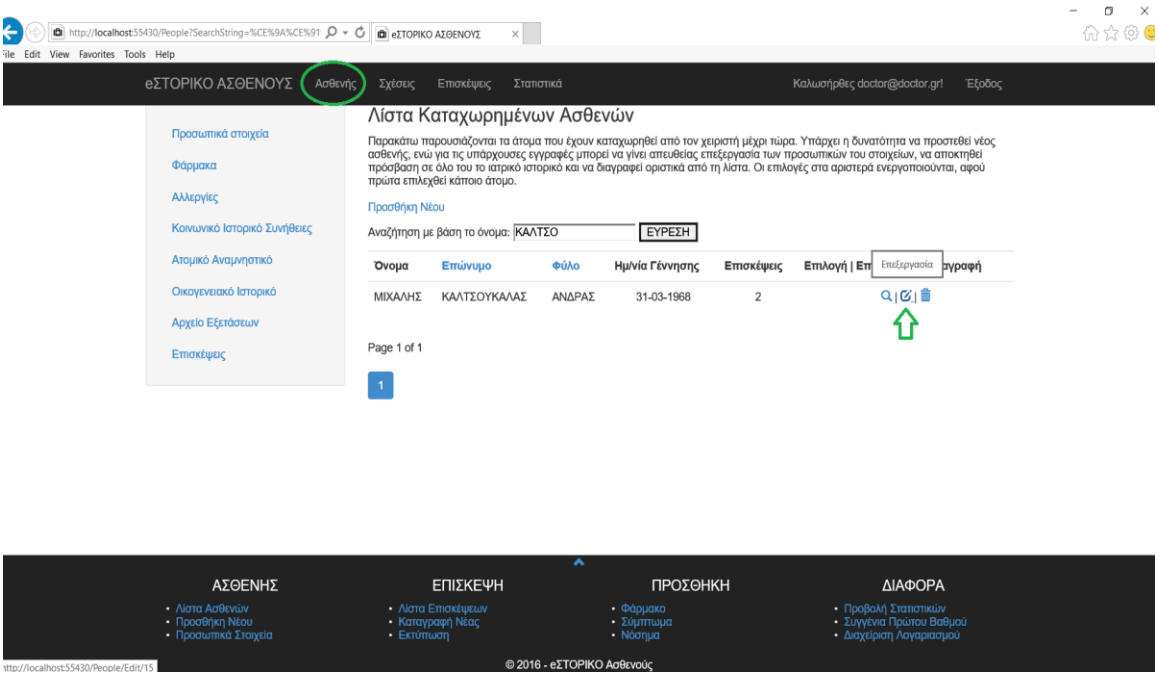




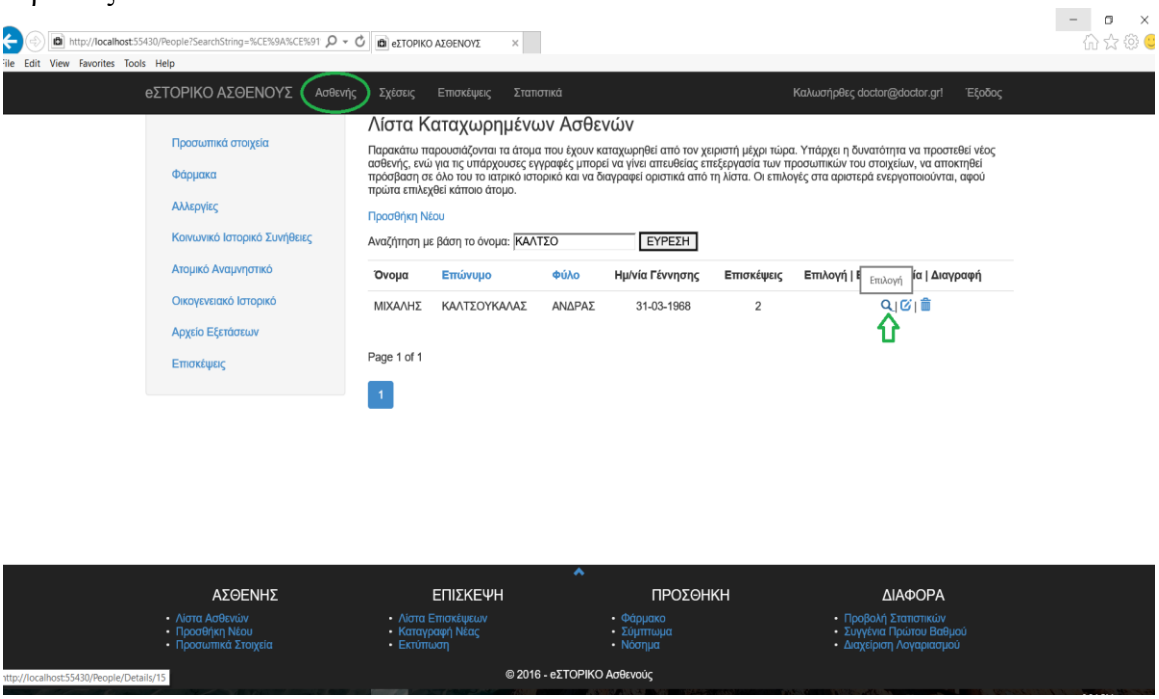
Στα δεξιά όπως παρατηρούμε έχει εμφανιστεί ένα κάθετο υπομενού με κάποιες επιλογές οι οποίες ενεργοποιούνται αφού πρώτα γίνει επιλογή κάποιου ασθενή από την λίστα των καταχωρημένων ατόμων.



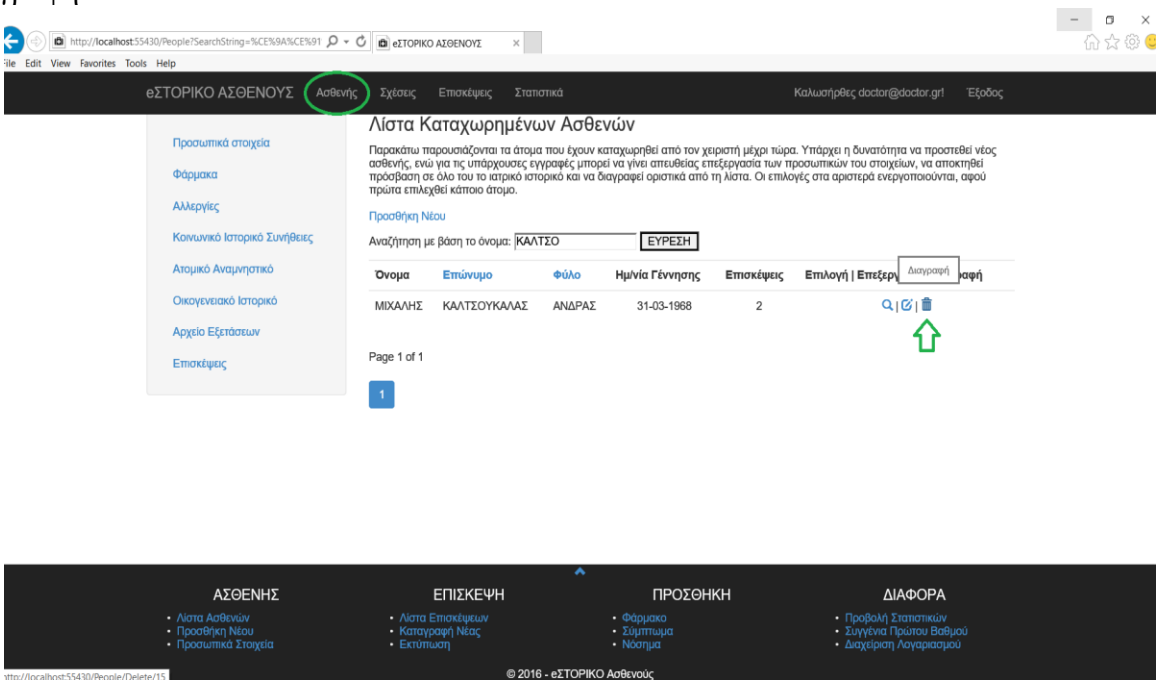
Για κάθε άτομο που επιθυμούμε υπάρχουν διαθέσιμες τρεις λειτουργίες. Αρχικά η «ΕΠΕΞΕΡΓΑΣΙΑ» με την οποία μεταφερόμαστε στην καρτέλα των δημογραφικών στοιχείων του ασθενούς και μπορούμε να τροποποιήσουμε τα πεδία αποθηκεύοντας της νέα κατάσταση.



Με την λειτουργία «ΕΠΙΛΟΓΗ» ενεργοποιούνται όλες οι καρτέλες του ατόμου οι οποίες αφορούν τις επιλογές του κάθετου υπομενού. Πλέον ο γιατρός για το τρέχον άτομο που τα στοιχεία του εμφανίζονται με κόκκινα γράμματα πάνω αριστερά μπορεί να δει και να επεξεργαστεί όλες τις καρτέλες του.

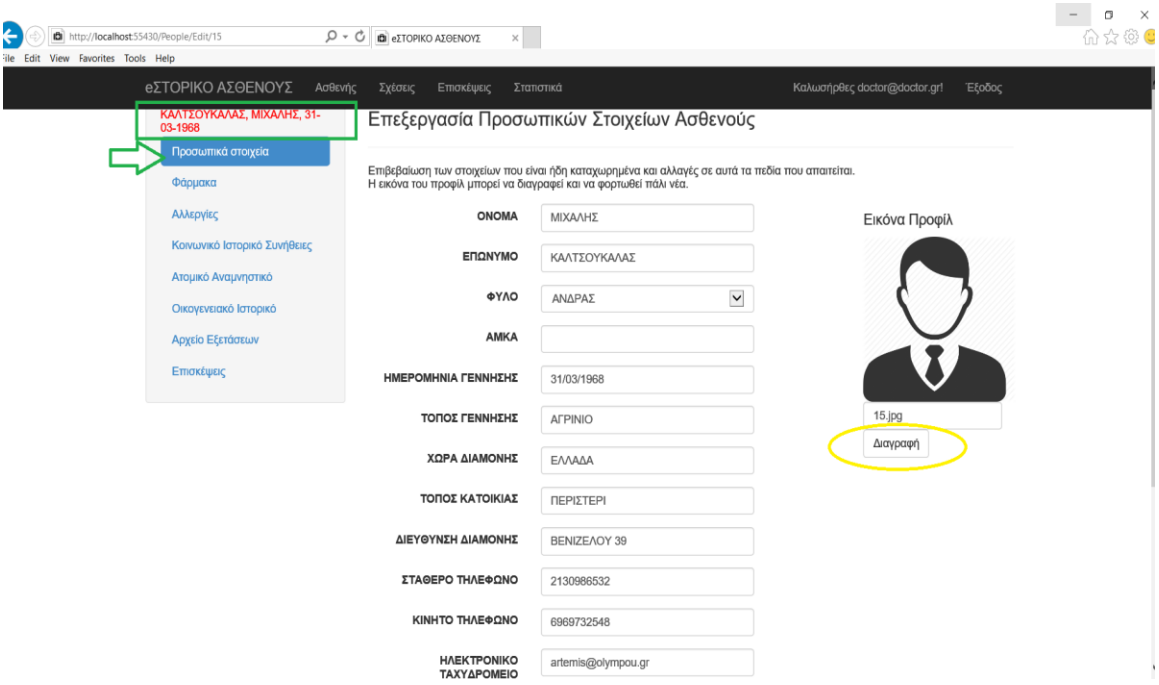


Τέλος μπορεί να διαγράψει το συγκεκριμένο άτομο αν το επιθυμεί οριστικά. Αυτό θα έχει ως αποτέλεσμα να χαθούν και όλες οι επισκέψεις που τυχόν έχουν καταγραφεί. Βέβαια για λόγους ασφαλείας και για να γίνεται ένας επιπλέον έλεγχος σε περίπτωση που υπάρχουν σχέσεις συγγένειας στις οποίες συμμετέχει θα πρέπει πρώτα να τις διαγράψει και μετά να προχωρήσει στη διαγραφή.

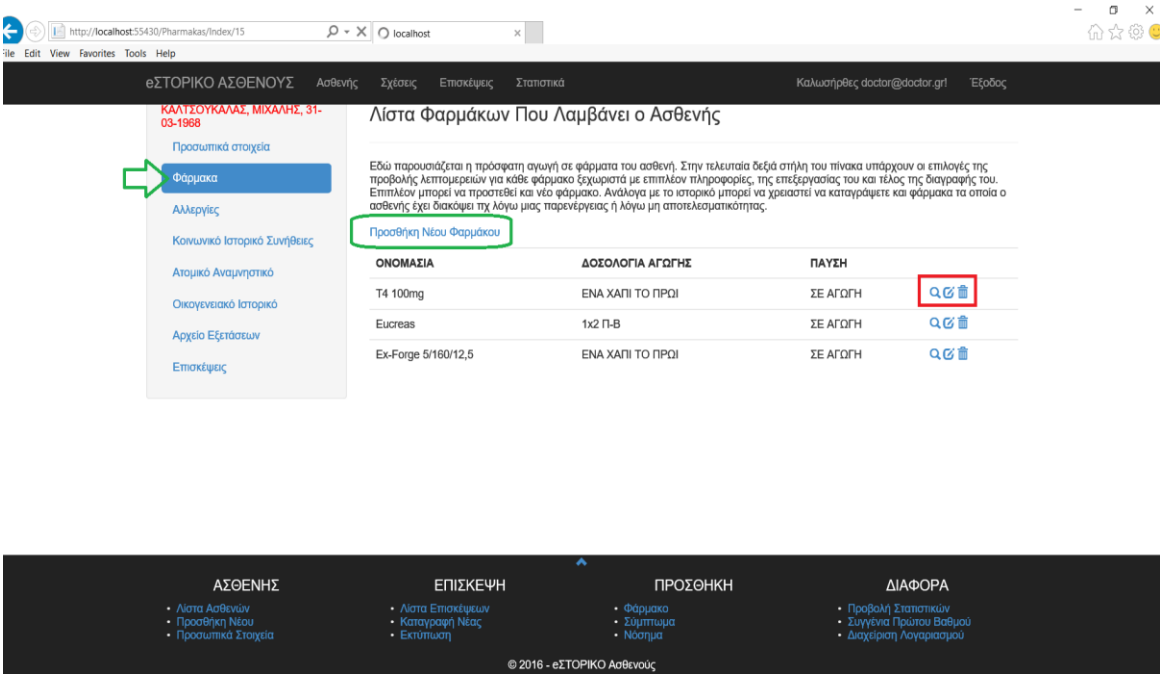


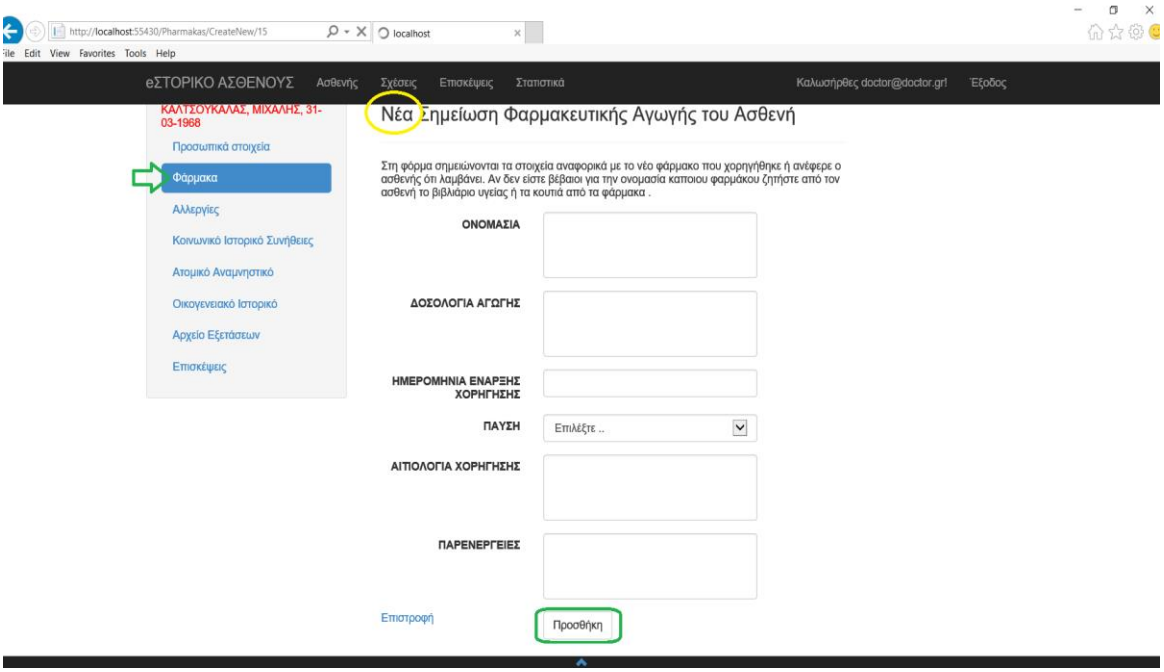
12.3 Καρτέλες Ασθενούς

Στην καρτέλα των προσωπικών στοιχείων του ατόμου, συμπληρώνονται κάποια πεδία υποχρεωτικά και μπορεί να φορτωθεί εικόνα ώστε η αναγνώριση του ατόμου να είναι πιο εύκολη.

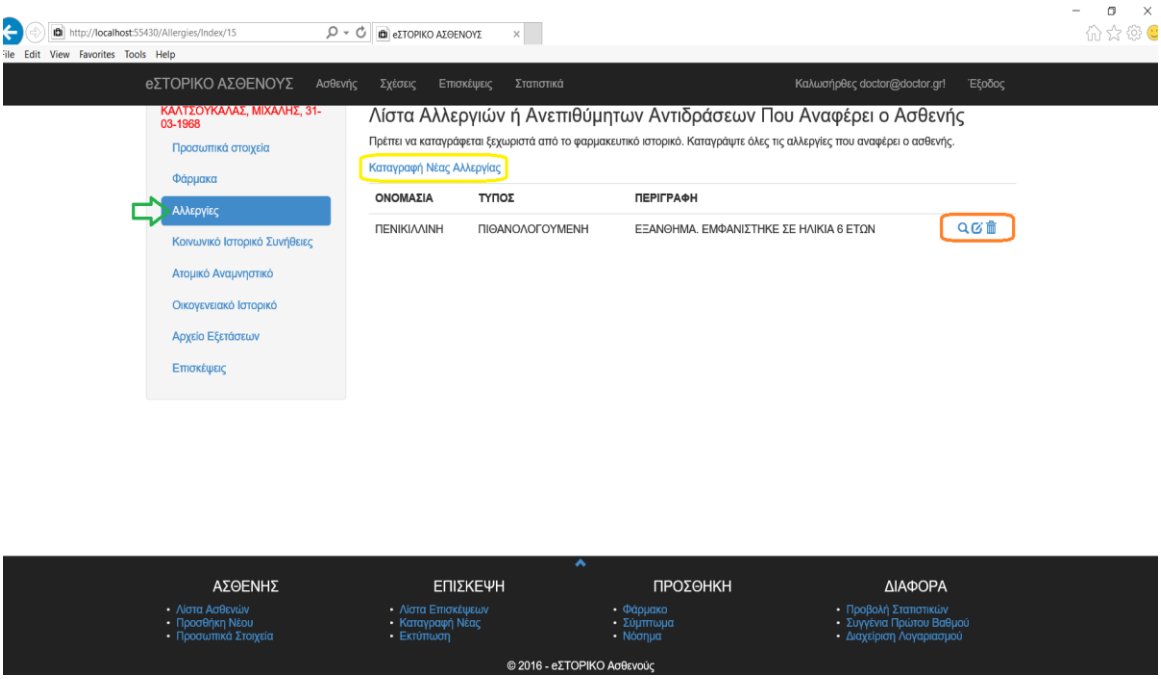


Στην καρτέλα με τα φάρμακα, καταχωρείται η φαρμακευτική αγωγή που ακολουθεί ο ασθενής, περιγράφονται λεπτομέρειες για το είδος του φαρμάκου, το λόγο, την δοσολογία και το αν είναι σε ισχύ.

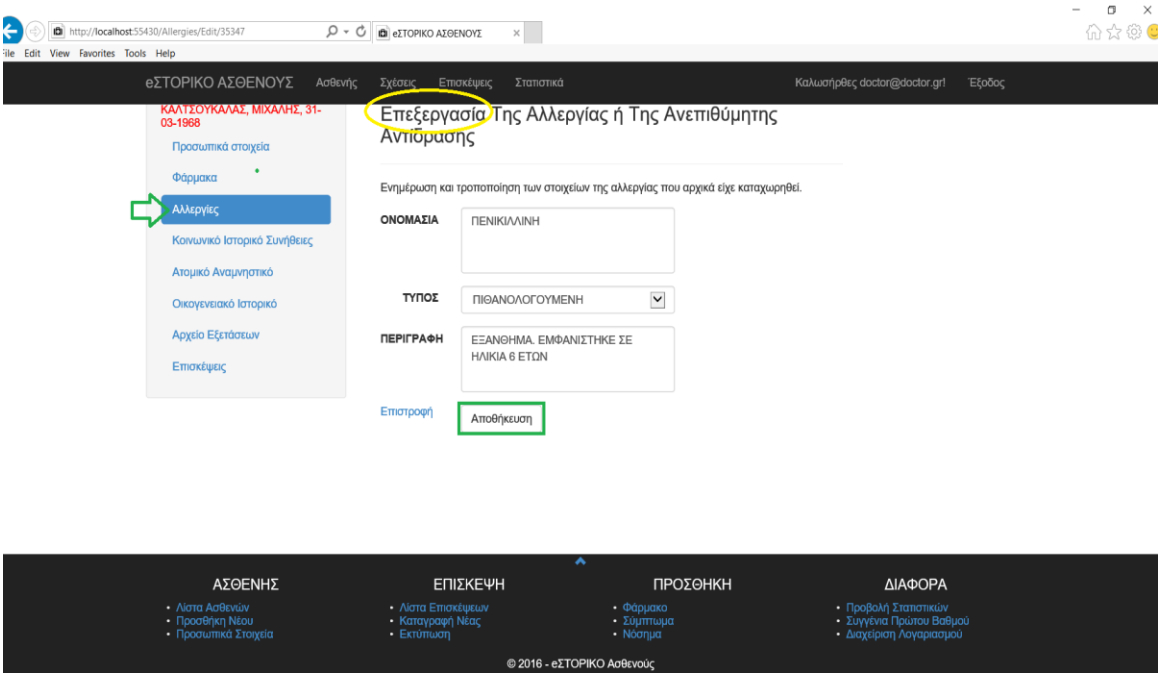




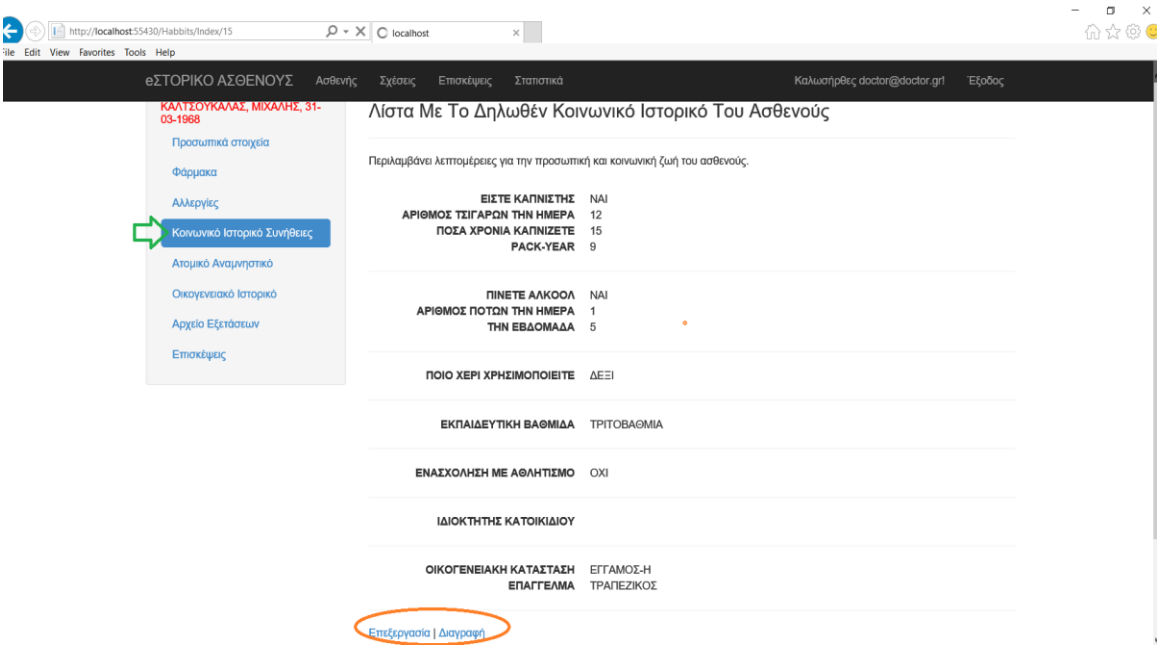
Στην καρτέλα με τις αλλεργίες του ασθενούς καταχωρούνται όλες οι αλλεργίες που έχει. Για κάθε μία πρέπει να τις χαρακτηρίσει ως βεβαιωμένες ή πιθανολογούμενες ώστε να μπορεί ο γιατρός να τις λάβει υπόψιν του κατά τις διαγνώσεις.



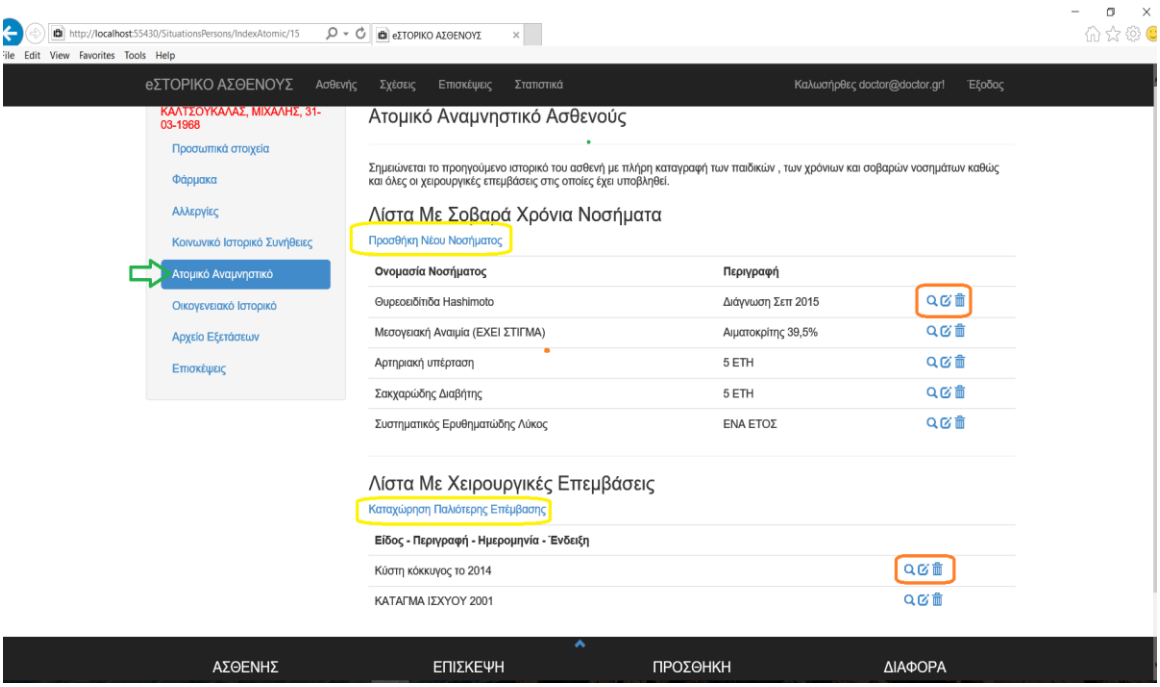
Για κάθε εγγραφή αλλεργίας που πραγματοποιεί, μπορεί στην συνέχεια να την τροποποιήσει, να την διαγράψει ώστε να έχει μία σωστή και ακριβή εικόνα για τον ασθενή του.



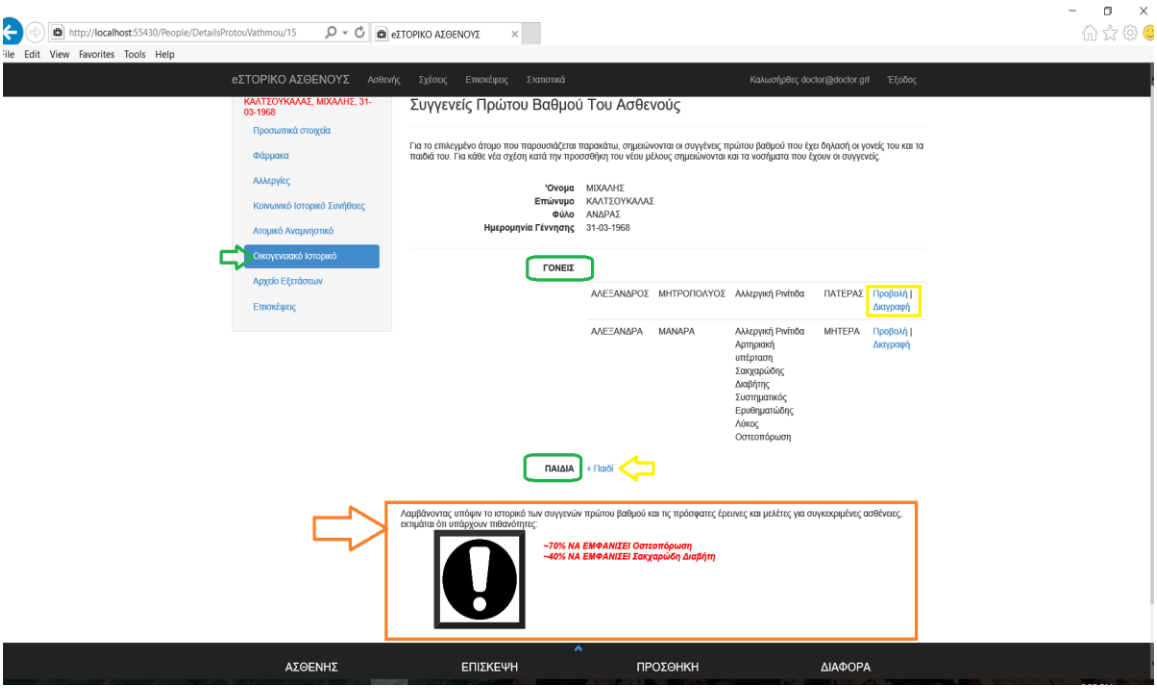
Στην καρτέλα με το κοινωνικό ιστορικό καταγράφονται λεπτομέρειες για την προσωπική ζωή του ασθενούς. Οι πληροφορίες αυτές είναι χρήσιμες και σημαντικές. Βέβαια καθώς μπορεί να αλλάξουν θα πρέπει ο γιατρός όποτε τον επισκέπτεται ο ασθενής του να την ενημερώνει την καρτέλα.



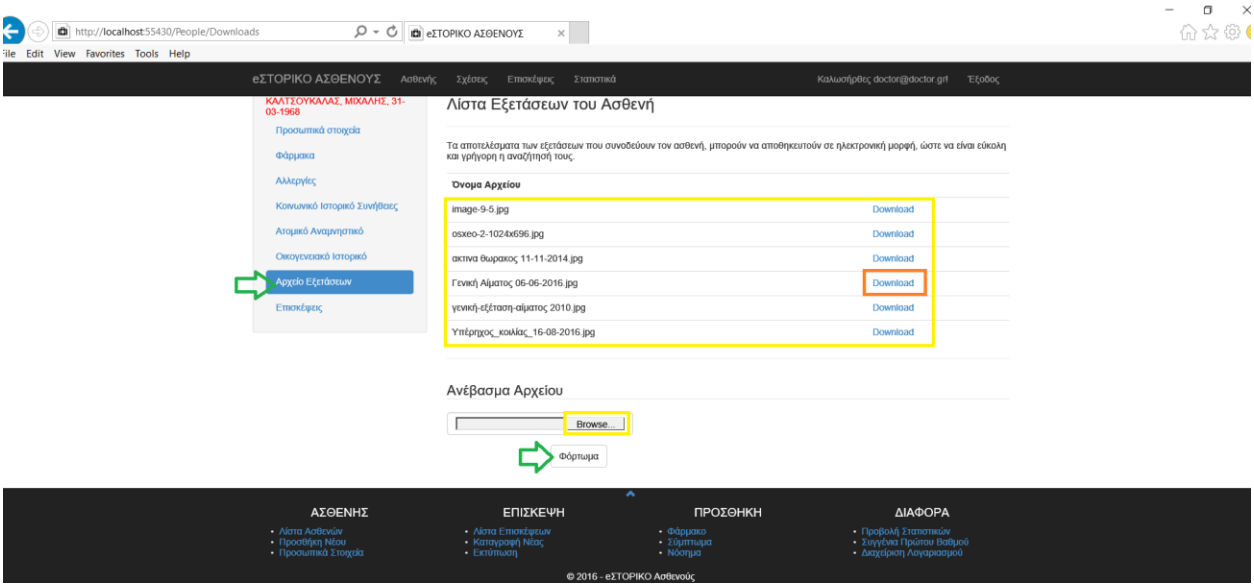
Στην καρτέλα αυτή ο γιατρός μπορεί να εισάγει πληροφορίες για σοβαρές ασθένειες που έχει νοσήσει ο ασθενής καθώς και να σημειώσει χειρουργικές επεμβάσεις τις οποίες έχει πραγματοποιήσει. Είναι σημαντικό στην περιγραφή να περαστεί το πότε, το γιατί και τυχόν περιπλοκές.



Στην καρτέλα με το οικογενειακό ιστορικό, μπορεί ο γιατρός να εισάγει τα άτομα που έχουν συγγένεια πρώτου βαθμού με τον ασθενή του. Κατά την προσθήκη μέλους δίνεται η δυνατότητα να επιλεχθεί κάποιο από τα ήδη καταχωρημένα άτομα που σημαίνει ότι και αυτό το μέλος είναι ασθενής που παρακολουθείτε από τον συγκεκριμένο γιατρό ή να καταγραφούν κάποια βασικά στοιχεία του νέου μέλους. Οι πληροφορίες αυτές περιλαμβάνουν εκτός από τα δημογραφικά στοιχεία και πληροφορίες για σοβαρά νοσήματα που έχει περάσει ώστε να ληφθούν υπόψιν σε περίπτωση που είναι κληρονομικά. Στο κάτω μέρος σε περίπτωση που υπάρχει κληρονομικό νόσημα, εμφανίζεται μία προειδοποίηση για να ληφθεί υπόψιν από τον γιατρό κατά την εξέταση.

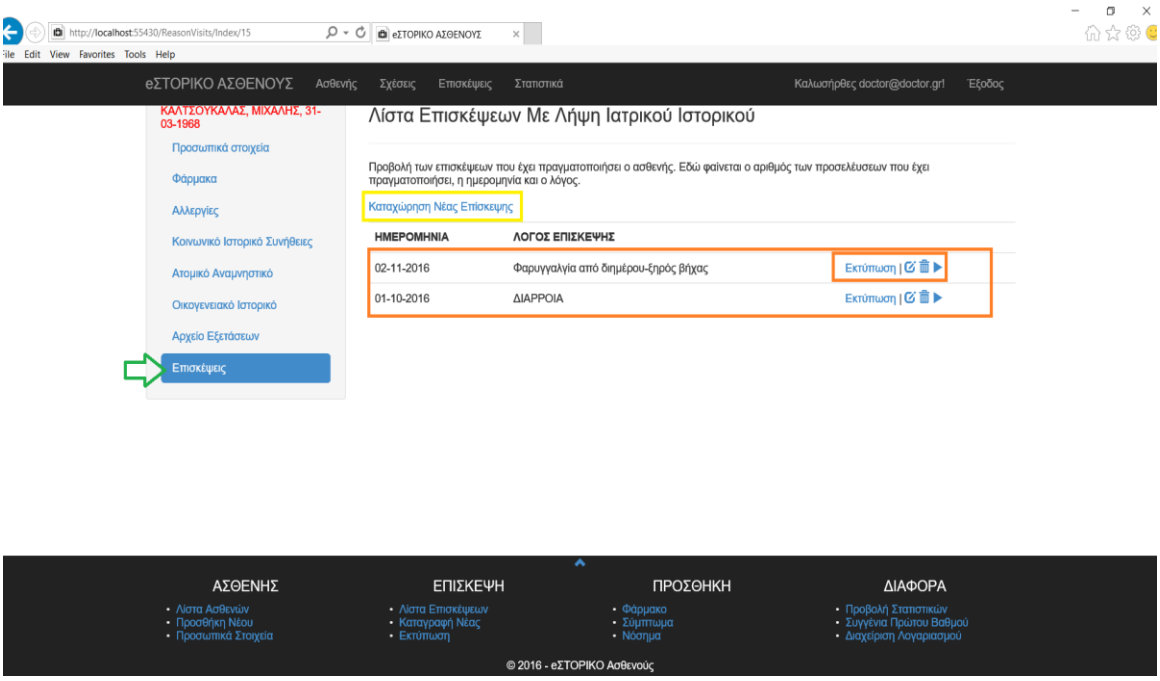


Τυχόν νέες εξετάσεις που πραγματοποιούνται και παλιότερες που υπάρχουν σε ηλεκτρονική μορφή, μπορούν να αποθηκευτούν και να εμφανίζονται σε αυτήν την καρτέλα.

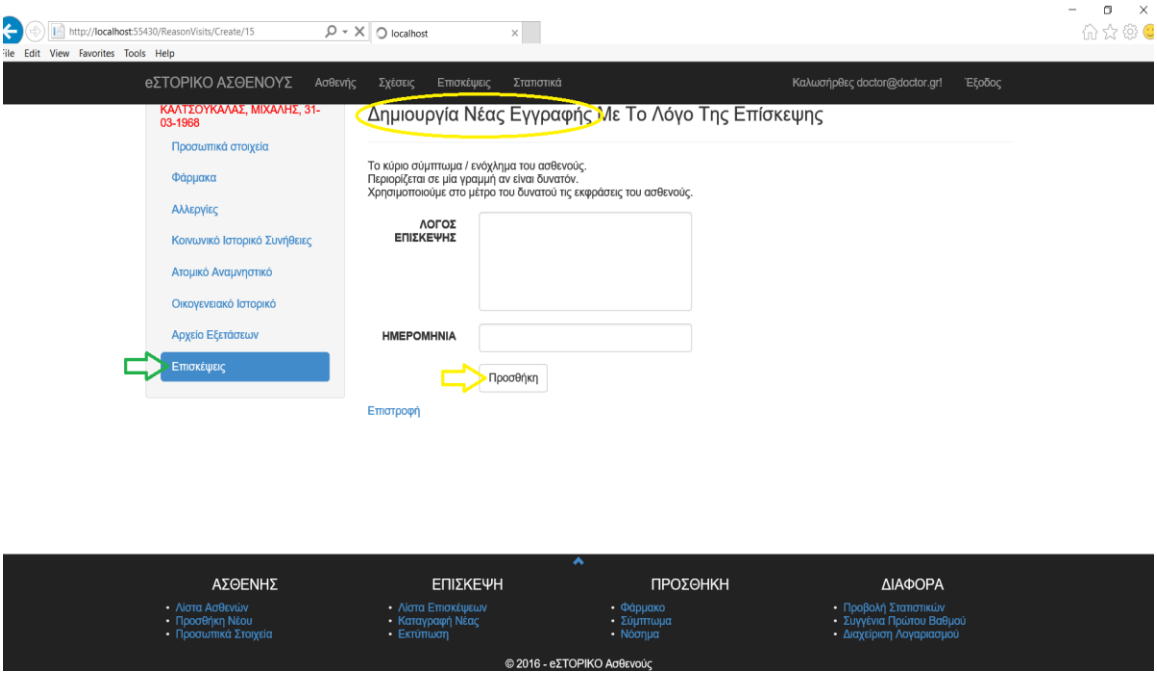


12.4 Λίστα Επισκέψεων και Προσθήκη Νέας

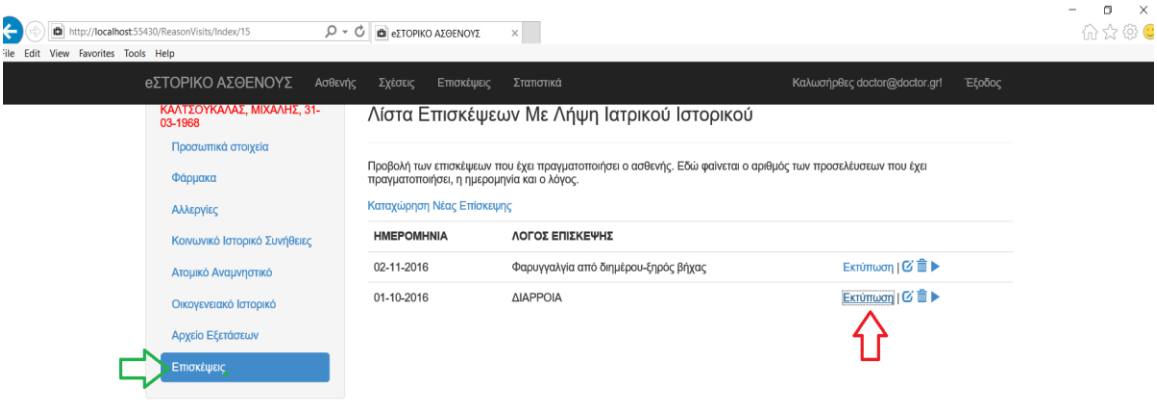
Για το επιλεγμένο άτομο στην παρακάτω καρτέλα βλέπουμε την λίστα με τις επισκέψεις που έχει πραγματοποιήσει και έχουν καταγραφεί. Τα δύο στοιχεία που χαρακτηρίζουν μία επίσκεψη είναι η μέρα και ο λόγος. Για κάθε μία, υπάρχουν διαθέσιμες τέσσερις λειτουργίες. Η «ΕΚΤΥΠΩΣΗ» που γίνεται εξαγωγή σε pdf, «ΕΠΕΞΕΡΓΑΣΙΑ» για τροποποίηση, «ΔΙΑΓΡΑΦΗ» και «ΕΠΙΛΟΓΗ» για προβολή των αναλυτικών καρτελών.

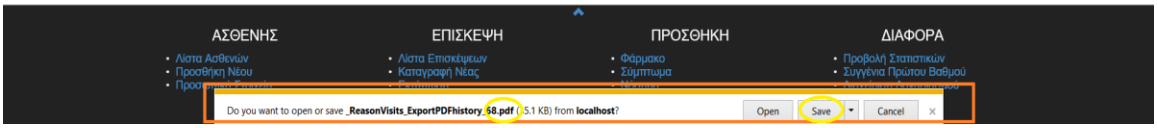


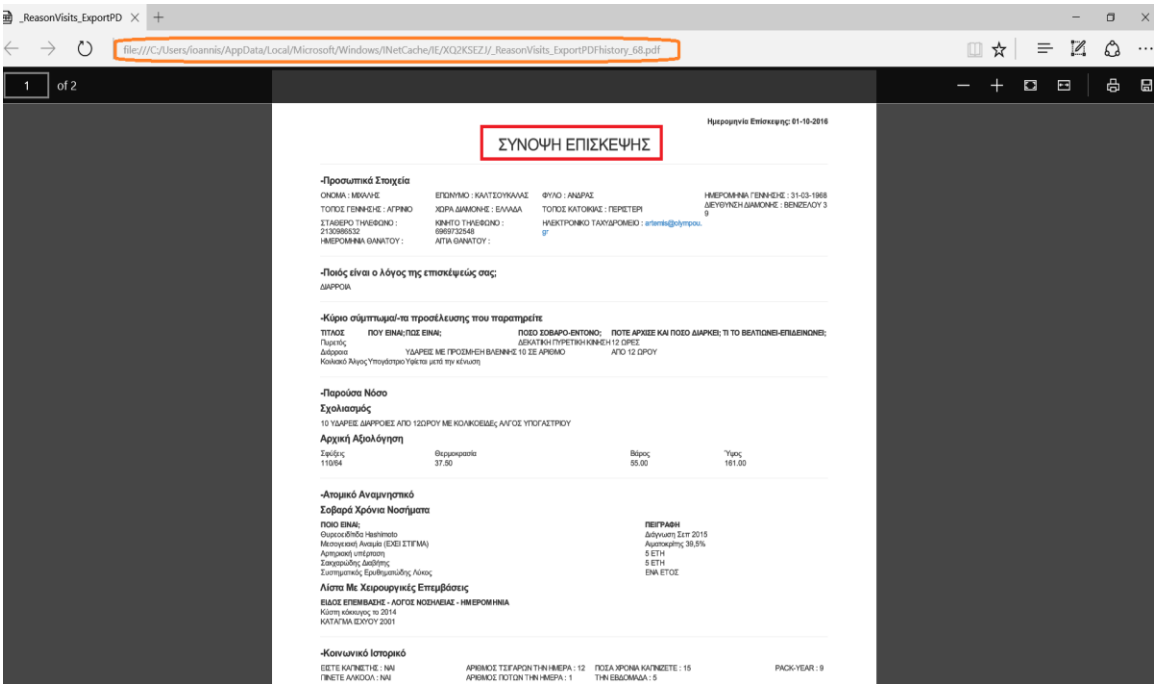
Ο γιατρός σημειώνει το λόγο που τον επισκέφτηκε ο ασθενής και αφού δημιουργήσει την επίσκεψη στην συνέχεια περνάει και τα χαρακτηριστικά στις διαθέσιμες καρτέλες της.



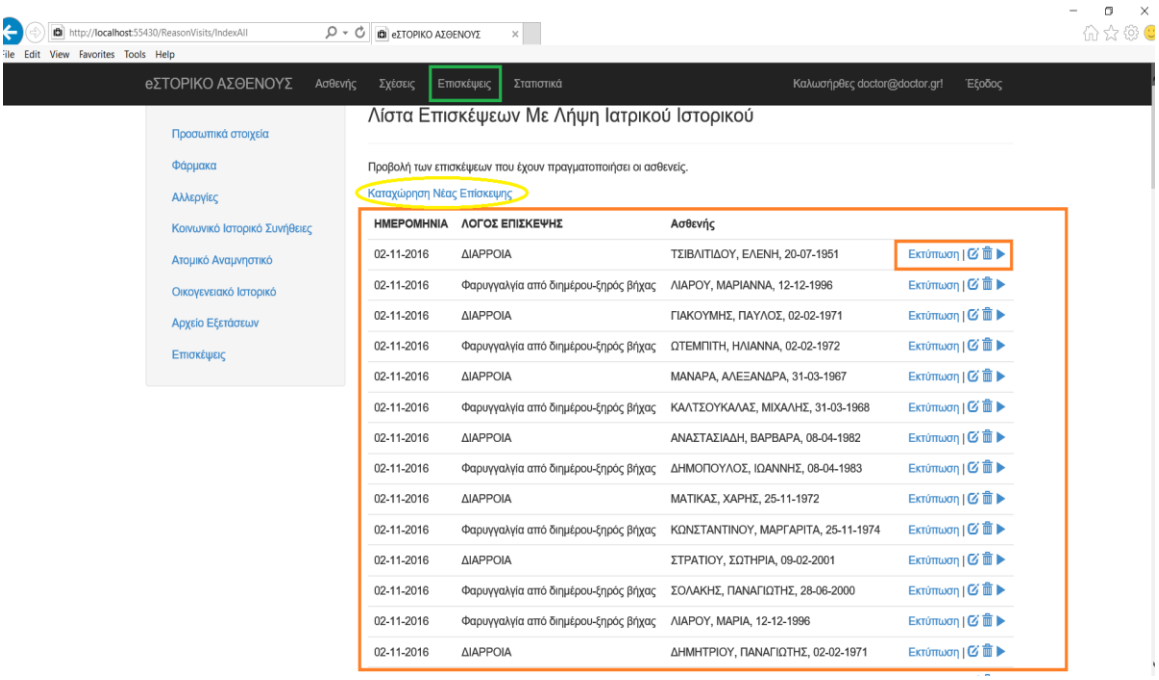
Επιλέγοντας την «ΕΚΤΥΠΩΣΗ», ο γιατρός μπορεί να εξάγει σε ηλεκτρονική μορφή και μετέπειτα να εκτυπώσει την αντίστοιχη επίσκεψη με τα στοιχεία που την συνοδεύουν και κάποιες γενικές πληροφορίες από την εικόνα του ασθενούς.



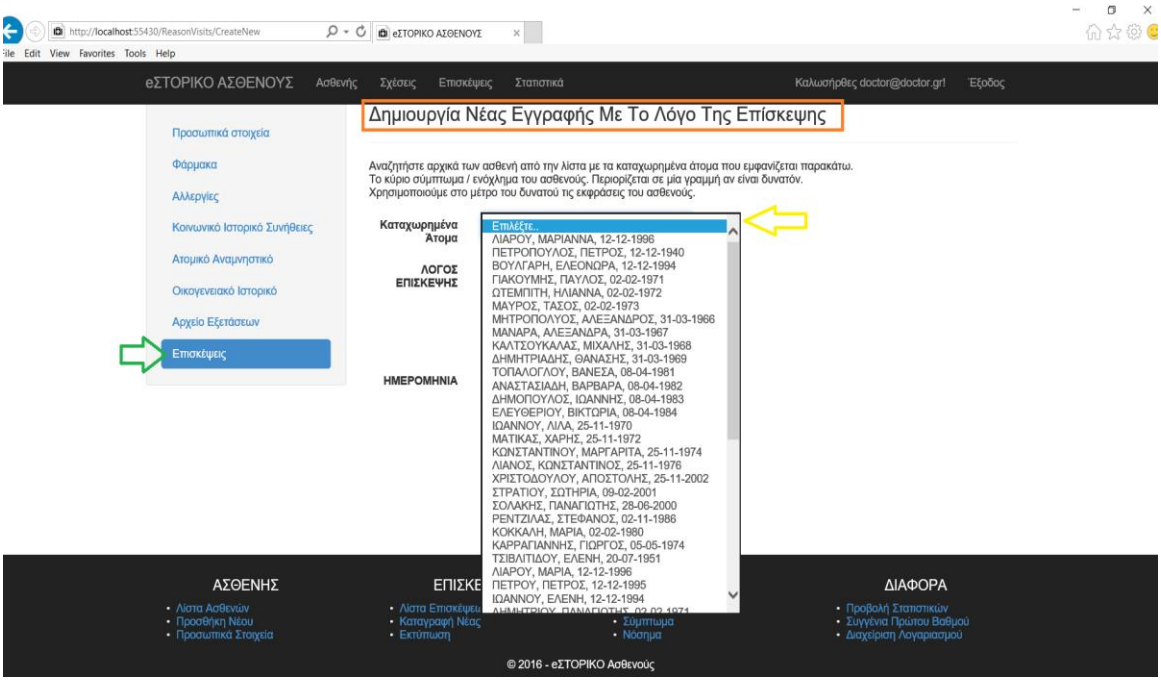




Από το κεντρικό μενού δίνεται η δυνατότητα να εμφανιστεί όλη η λίστα με τις επισκέψεις που έχουν καταγραφεί από τον συγκεκριμένο γιατρό. Για κάθε μία μπορεί να επιλέξει από τις διαθέσιμες λειτουργίες.

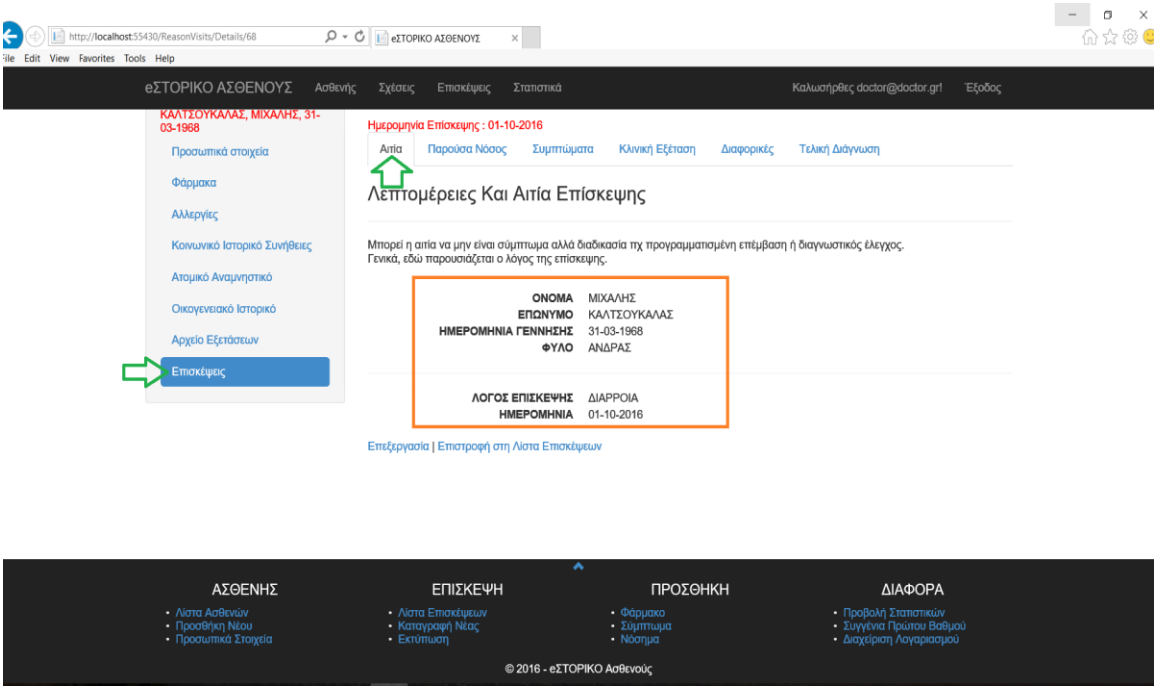


Επιπλέον η εφαρμογή του δίνει την δυνατότητα να δημιουργήσει απευθείας νέα επίσκεψη κατευθείαν για κάποιο από τα καταχωρημένα άτομα.

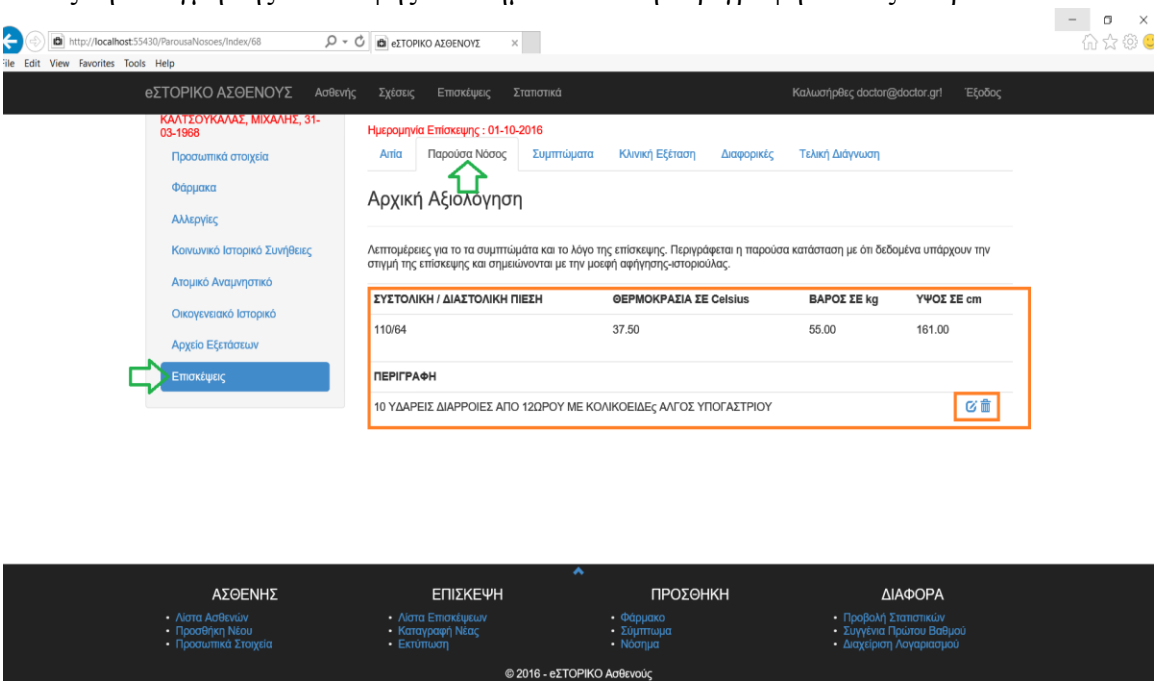


12.5 Καρτέλες Επίσκεψης

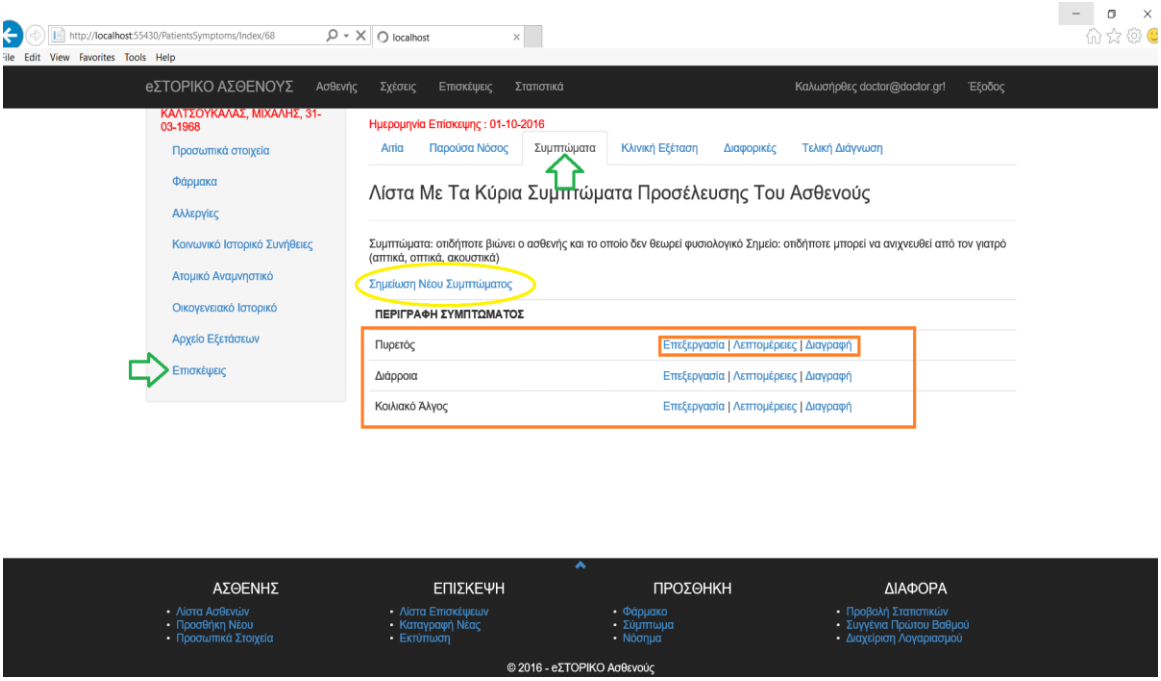
Όταν γίνει η δημιουργία μιας νέας επίσκεψης ή επιλέγετε μια συγκεκριμένης επίσκεψη ενός ασθενούς, κάτω από το κεντρικό οριζόντιο μενού, εμφανίζεται ένα νέο το οποίο περιέχει καρτέλες με τις διαθέσιμες λειτουργίες που την αφορούν. Στην πρώτη καρτέλα «ΑΙΤΙΑ», προβάλλονται κάποιες γενικές πληροφορίες.



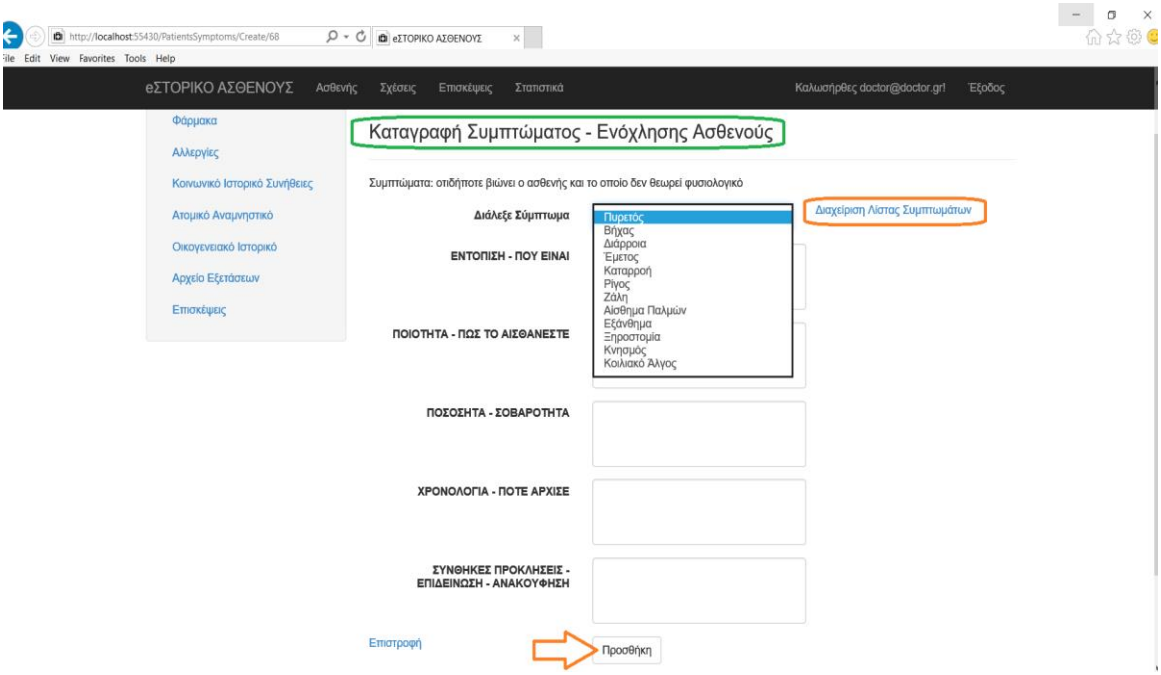
Στην δεύτερη καρτέλα «ΠΑΡΟΥΣΑ ΝΟΣΟ», σημειώνεται η εικόνα της κατάστασης του ασθενούς την στιγμή της επίσκεψης και σημειώνεται η περιγραφή που εξιστορεί.



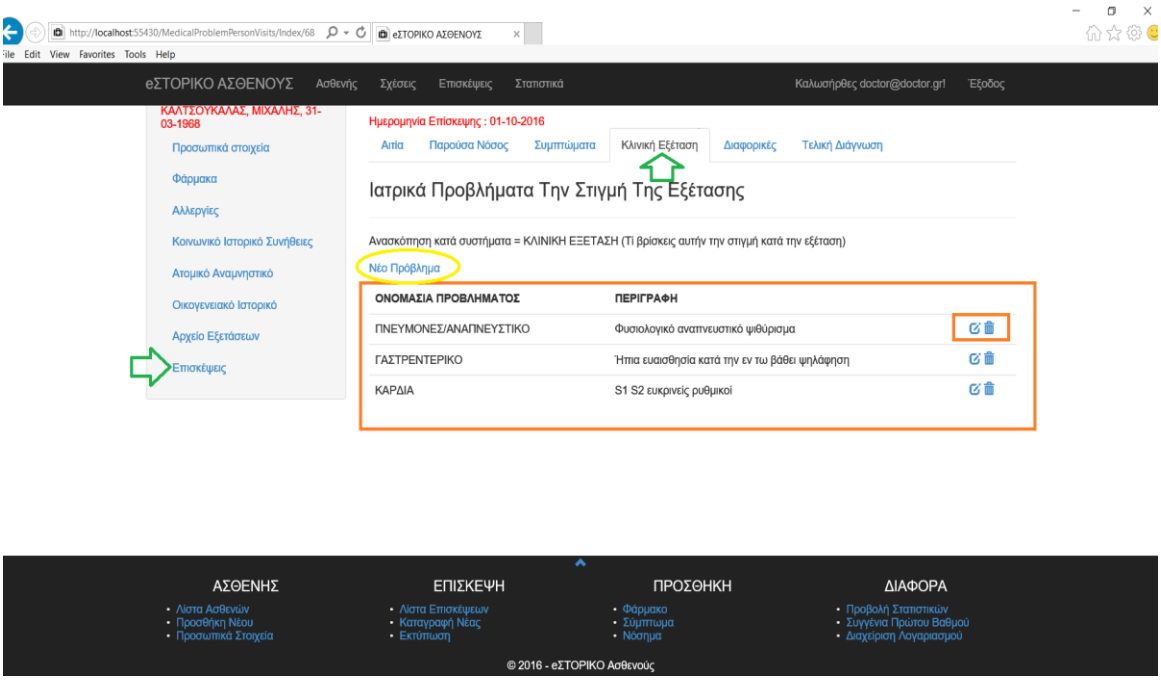
Το σημαντικότερο είναι η καταγραφή όλων των συμπτωμάτων που έχει παρατηρήσει ο ασθενής και τον έχουν οδηγήσει στο να έρθει στο γιατρό του.



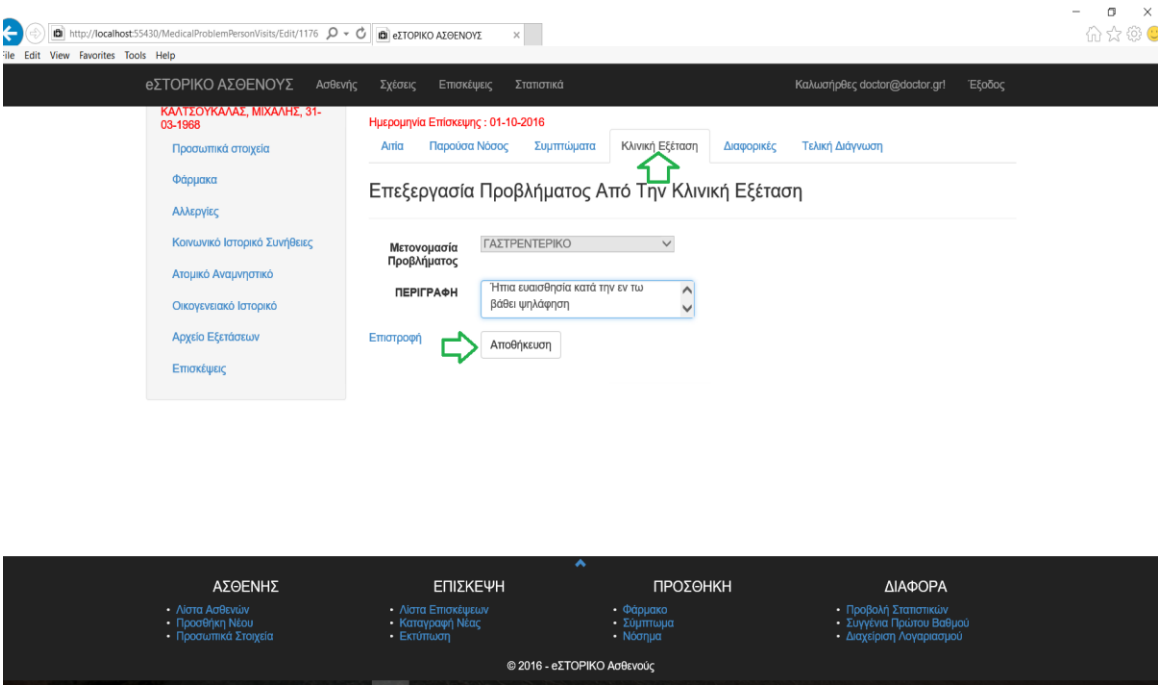
Για κάθε ένα σύμπτωμα πρέπει να σημειωθούν αρκετές πληροφορίες ώστε να μπορούν να αξιολογηθούν στην πορεία. Επιπλέον μπορεί να προσθέσει και κάποιο σύμπτωμα που δεν υπάρχει ήδη, ώστε να είναι διαθέσιμο και σε μελλοντική επίσκεψη.



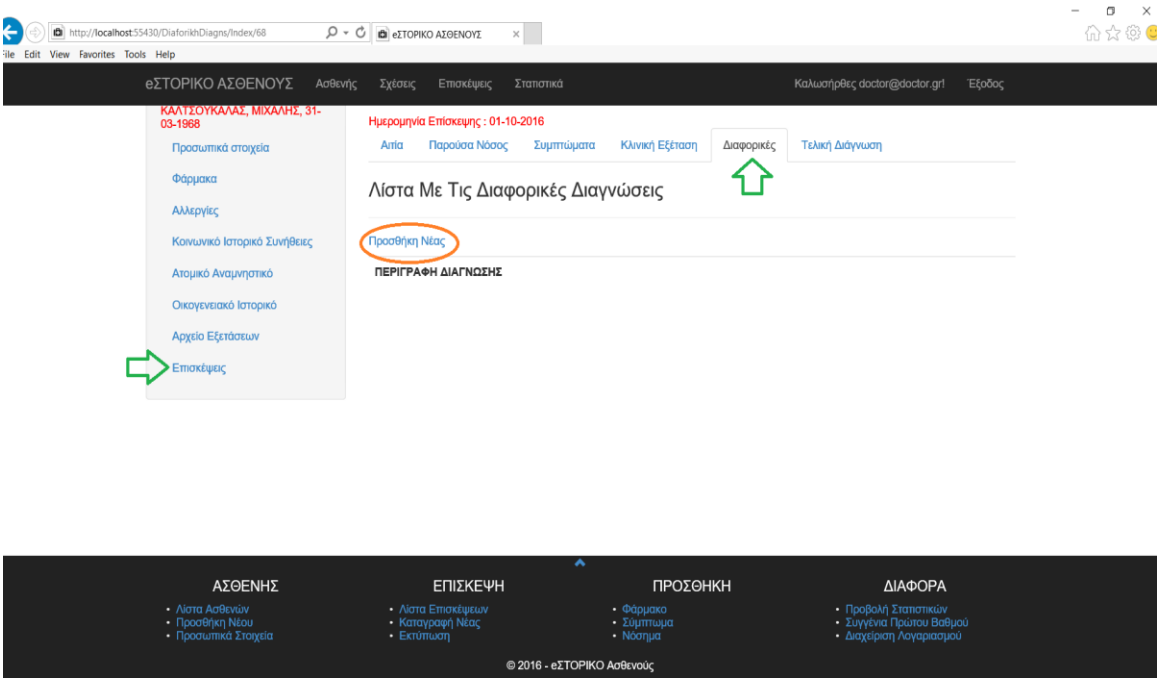
Στην καρτέλα «ΚΛΙΝΙΚΗ ΕΞΕΤΑΣΗ» της επίσκεψης, θα πρέπει να σημειωθούν πληροφορίες για τα συστήματα που υπάρχει ενόχληση κατά τον ασθενή ή που έχει εξετάσει ο γιατρός ώστε να εξάγει συμπεράσματα. Το κάθε πρόβλημα που σημειώνει στην συνέχεια μπορεί να το επεξεργαστεί ή ακόμη και να το διαγράψει απευθείας.



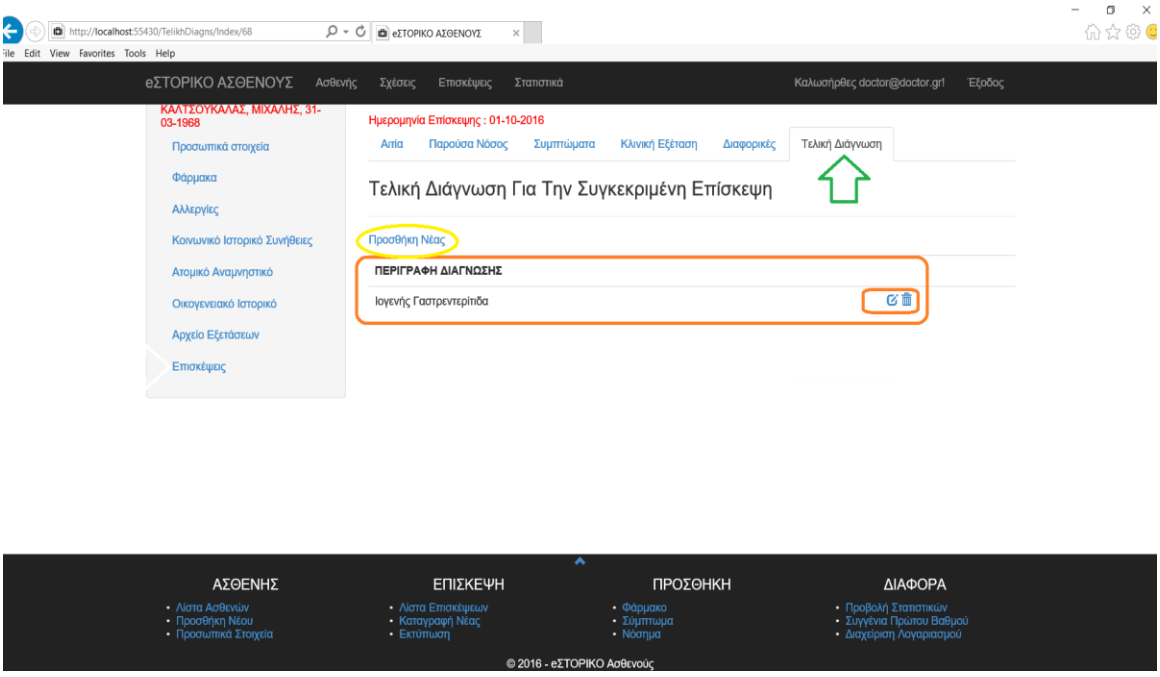
Μετά από οποιαδήποτε επεξεργασία, πρέπει να επιλέξει αποθήκευση ώστε να διατηρηθούν οι αλλαγές.



Ο γιατρός επειδή μπορεί να μην εξάγει απευθείας διάγνωση για το συγκεκριμένο περιστατικό, έχει την δυνατότητα να σημειώσει εδώ πέρα την εκτίμηση που έχει κάνει και την μέχρι στιγμής διάγνωσή του.

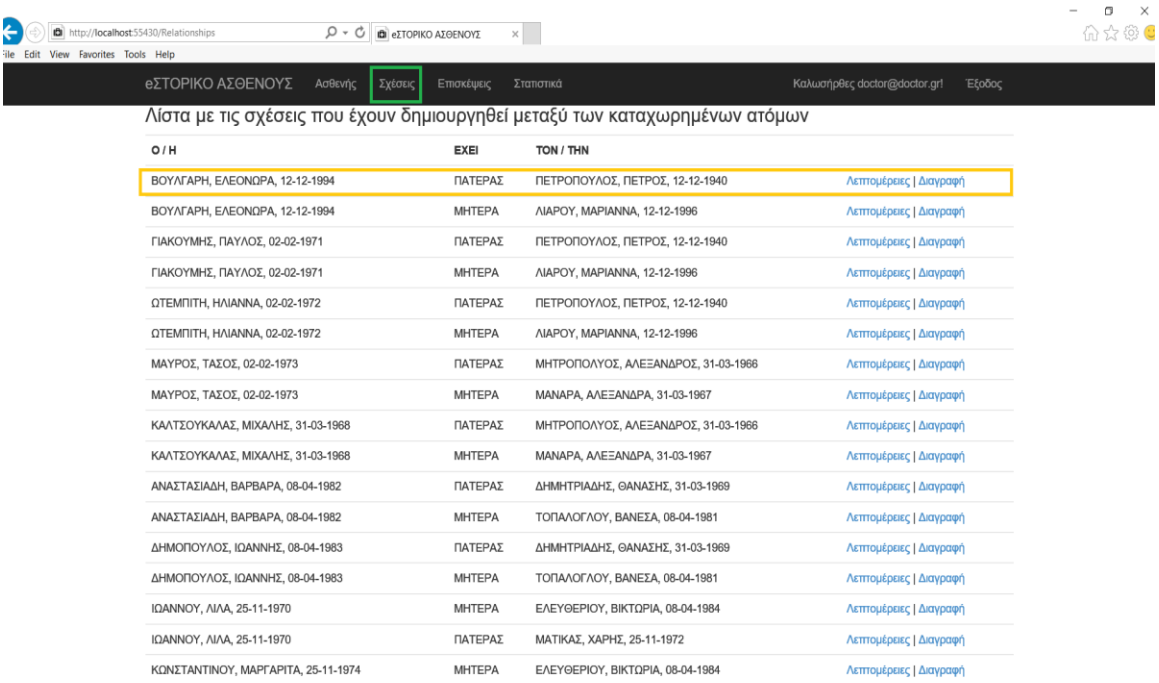


Τέλος καταγράφει την τελική διάγνωση με την οποία και ολοκληρώνεται κατά κύριο ποσοστό και η συγκεκριμένη επίσκεψη. Το πεδίο είναι ελεύθερο και ο γιατρός μπορεί να σημειώσει οτιδήποτε θα τον βοηθήσει αργότερα να θυμηθεί για το περιστατικό.



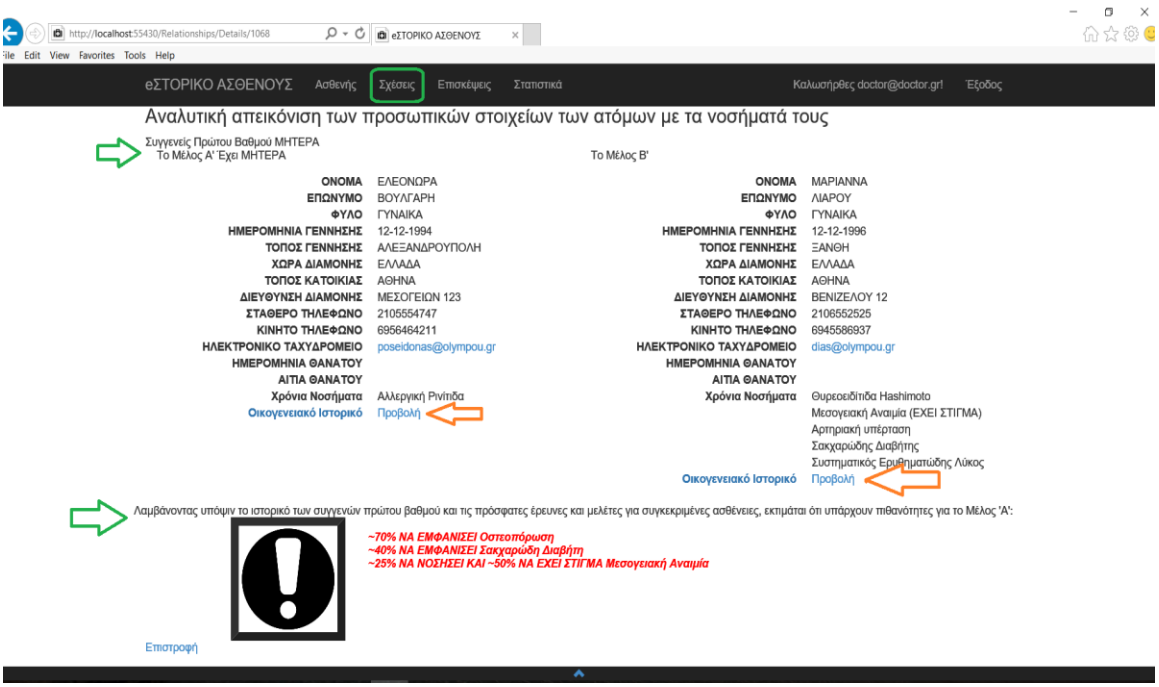
12.6 Σχέσεις Συγγένειας

Από το κεντρικό μενού ο γιατρός μπορεί να κάνει προβολή όλων των σχέσεων συγγένειας που έχει δημιουργήσει μεταξύ των καταχωρημένων ασθενών του. Την κάθε σχέση μπορεί να την επεξεργαστεί και φυσικά να κάνει προβολή λεπτομερειών ώστε να εξάγει κάποια συμπεράσματα.



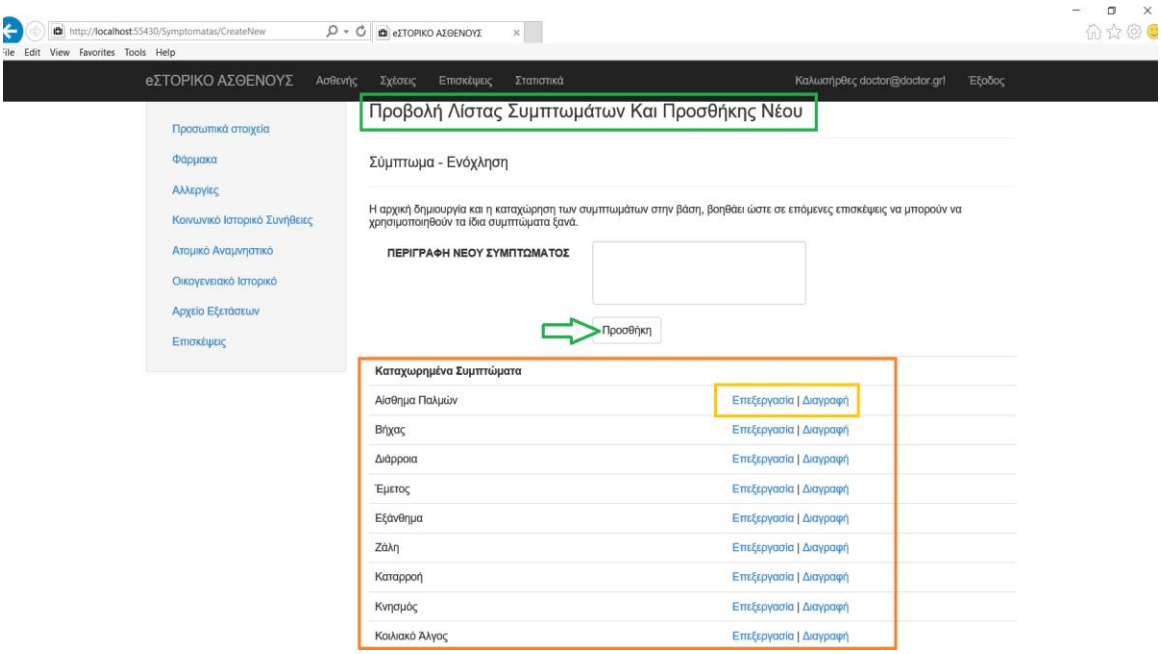
Ο / Η	ΕΧΕΙ	ΤΟΝ / ΤΗΝ	
ΒΟΥΛΓΑΡΗ, ΕΛΕΟΝΩΡΑ, 12-12-1994	ΠΑΤΕΡΑΣ	ΠΕΤΡΟΠΟΥΛΟΣ, ΠΕΤΡΟΣ, 12-12-1940	Λεπτομέρειες | Διαγραφή
ΒΟΥΛΓΑΡΗ, ΕΛΕΟΝΩΡΑ, 12-12-1994	ΜΗΤΕΡΑ	ΛΙΑΡΟΥ, ΜΑΡΙΑΝΝΑ, 12-12-1996	Λεπτομέρειες | Διαγραφή
ΓΙΑΚΟΥΜΗΣ, ΠΑΥΛΟΣ, 02-02-1971	ΠΑΤΕΡΑΣ	ΠΕΤΡΟΠΟΥΛΟΣ, ΠΕΤΡΟΣ, 12-12-1940	Λεπτομέρειες | Διαγραφή
ΓΙΑΚΟΥΜΗΣ, ΠΑΥΛΟΣ, 02-02-1971	ΜΗΤΕΡΑ	ΛΙΑΡΟΥ, ΜΑΡΙΑΝΝΑ, 12-12-1996	Λεπτομέρειες | Διαγραφή
ΩΤΕΜΠΙΤΗ, ΗΛΙΑΝΝΑ, 02-02-1972	ΠΑΤΕΡΑΣ	ΠΕΤΡΟΠΟΥΛΟΣ, ΠΕΤΡΟΣ, 12-12-1940	Λεπτομέρειες | Διαγραφή
ΩΤΕΜΠΙΤΗ, ΗΛΙΑΝΝΑ, 02-02-1972	ΜΗΤΕΡΑ	ΛΙΑΡΟΥ, ΜΑΡΙΑΝΝΑ, 12-12-1996	Λεπτομέρειες | Διαγραφή
ΜΑΥΡΟΣ, ΤΑΣΟΣ, 02-02-1973	ΠΑΤΕΡΑΣ	ΜΗΤΡΟΠΟΛΥΟΣ, ΑΛΕΞΑΝΔΡΟΣ, 31-03-1966	Λεπτομέρειες | Διαγραφή
ΜΑΥΡΟΣ, ΤΑΣΟΣ, 02-02-1973	ΜΗΤΕΡΑ	ΜΑΝΑΡΑ, ΑΛΕΞΑΝΔΡΑ, 31-03-1967	Λεπτομέρειες | Διαγραφή
ΚΑΛΤΣΟΥΚΑΛΑΣ, ΜΙΧΑΛΗΣ, 31-03-1968	ΠΑΤΕΡΑΣ	ΜΗΤΡΟΠΟΛΥΟΣ, ΑΛΕΞΑΝΔΡΟΣ, 31-03-1966	Λεπτομέρειες | Διαγραφή
ΚΑΛΤΣΟΥΚΑΛΑΣ, ΜΙΧΑΛΗΣ, 31-03-1968	ΜΗΤΕΡΑ	ΜΑΝΑΡΑ, ΑΛΕΞΑΝΔΡΑ, 31-03-1967	Λεπτομέρειες | Διαγραφή
ΑΝΑΣΤΑΣΙΑΔΗ, ΒΑΡΒΑΡΑ, 08-04-1982	ΠΑΤΕΡΑΣ	ΔΗΜΗΤΡΙΑΔΗΣ, ΘΑΝΑΣΗΣ, 31-03-1969	Λεπτομέρειες | Διαγραφή
ΑΝΑΣΤΑΣΙΑΔΗ, ΒΑΡΒΑΡΑ, 08-04-1982	ΜΗΤΕΡΑ	ΤΟΠΑΛΟΓΛΟΥ, ΒΑΝΕΣΑ, 08-04-1981	Λεπτομέρειες | Διαγραφή
ΔΗΜΟΠΟΥΛΟΣ, ΙΩΑΝΝΗΣ, 08-04-1983	ΠΑΤΕΡΑΣ	ΔΗΜΗΤΡΙΑΔΗΣ, ΘΑΝΑΣΗΣ, 31-03-1969	Λεπτομέρειες | Διαγραφή
ΔΗΜΟΠΟΥΛΟΣ, ΙΩΑΝΝΗΣ, 08-04-1983	ΜΗΤΕΡΑ	ΤΟΠΑΛΟΓΛΟΥ, ΒΑΝΕΣΑ, 08-04-1981	Λεπτομέρειες | Διαγραφή
ΙΩΑΝΝΟΥ, ΛΙΛΑ, 25-11-1970	ΜΗΤΕΡΑ	ΕΛΕΥΘΕΡΙΟΥ, ΒΙΚΤΩΡΙΑ, 08-04-1984	Λεπτομέρειες | Διαγραφή
ΙΩΑΝΝΟΥ, ΛΙΛΑ, 25-11-1970	ΠΑΤΕΡΑΣ	ΜΑΤΙΚΑΣ, ΧΑΡΗΣ, 25-11-1972	Λεπτομέρειες | Διαγραφή
ΚΩΝΣΤΑΝΤΙΝΟΥ, ΜΑΡΓΑΡΙΤΑ, 25-11-1974	ΜΗΤΕΡΑ	ΕΛΕΥΘΕΡΙΟΥ, ΒΙΚΤΩΡΙΑ, 08-04-1984	Λεπτομέρειες | Διαγραφή

Βοηθητικά εμφανίζονται πληροφορίες για κάθε σχέση. Τα προσωπικά στοιχεία και μία λίστα με τα νοσήματα που έχει αναφέρει ή έχει διαγνωστεί ο κάθε ασθενής. Συμπληρωματικά στο τέλος προβάλλεται και μία ενδεχόμενη προειδοποίηση στην περίπτωση που λόγω συγγένειας και ιατρικού ιστορικό, υπάρχει πιθανότητα για κάποιο από τα μέλη να εμφανίσει κάποια ασθένεια.



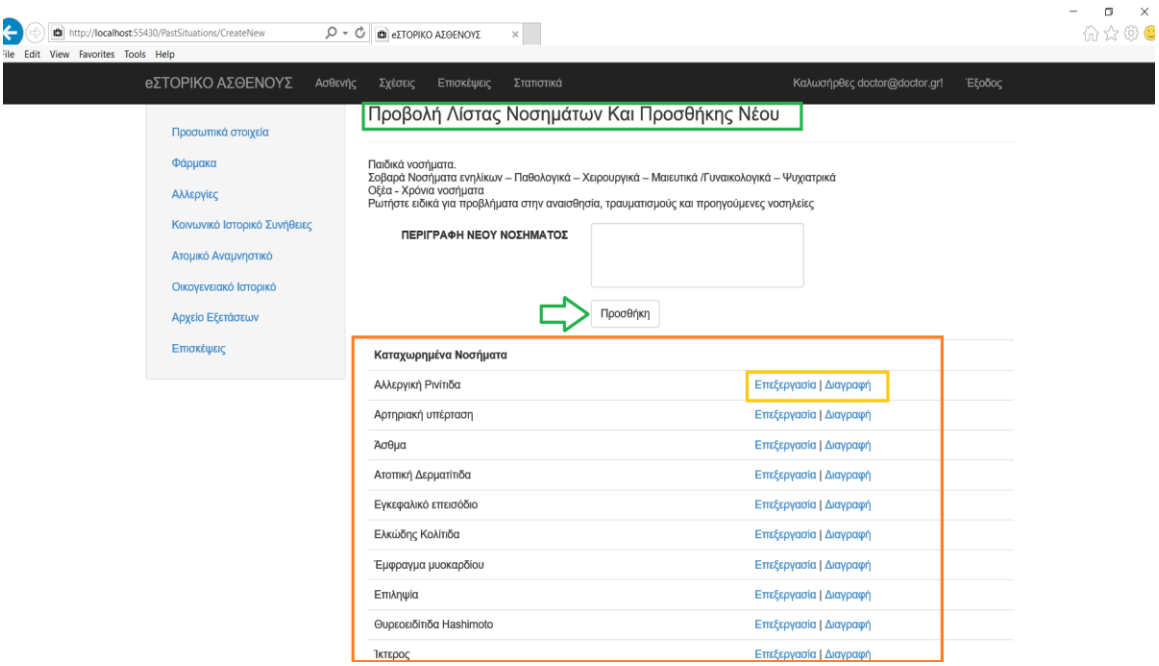
12.7 Λίστα Συμπτωμάτων και Προσθήκη Νέου

Από την παρακάτω καρτέλα μπορεί ο χειριστής ιατρός να καταχωρήσει νέα συμπτώματα, ώστε με αυτό τον τρόπο να ενημερώσει την λίστα του και να έχει σε μελλοντικές καταγραφές μία ενημερωμένη λίστα για εύκολη και γρήγορη αναζήτηση συμπτώματος.



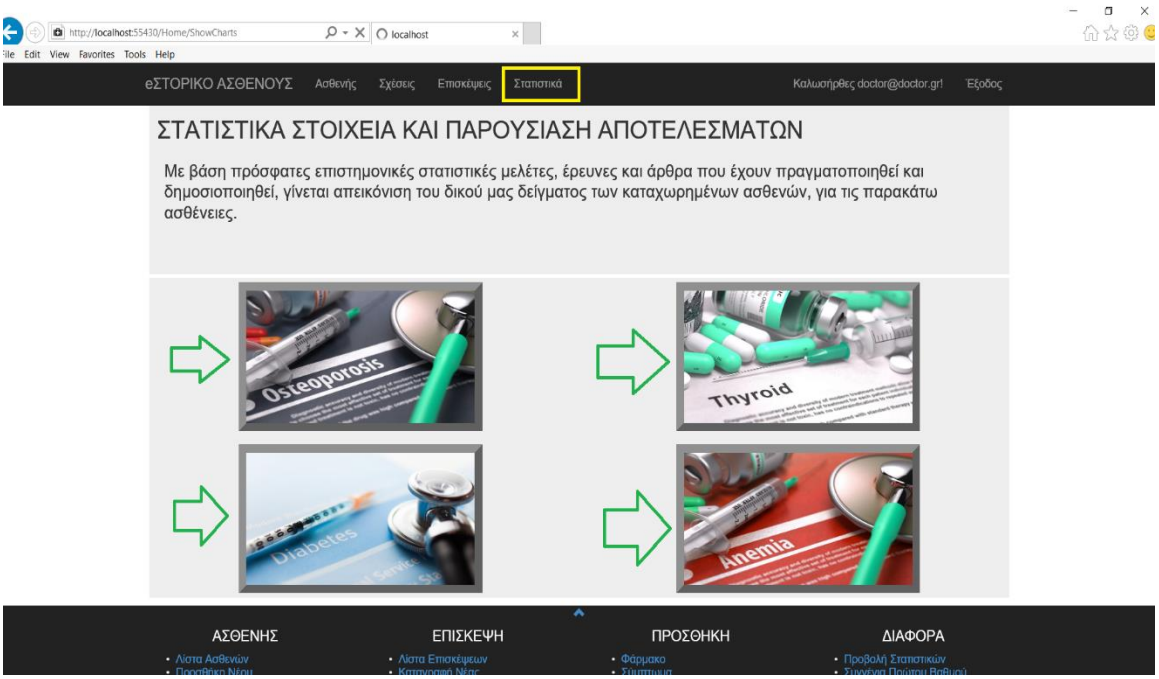
12.8 Λίστα Νοσημάτων και Προσθήκη Νέου

Από την παρακάτω καρτέλα μπορεί ο χειριστής ιατρός να καταχωρήσει νέα νοσήματα, ώστε με αυτό τον τρόπο να ενημερώσει την λίστα του και να έχει σε μελλοντικές καταγραφές μία ενημερωμένη λίστα για εύκολη και γρήγορη αναζήτηση νοσημάτων.

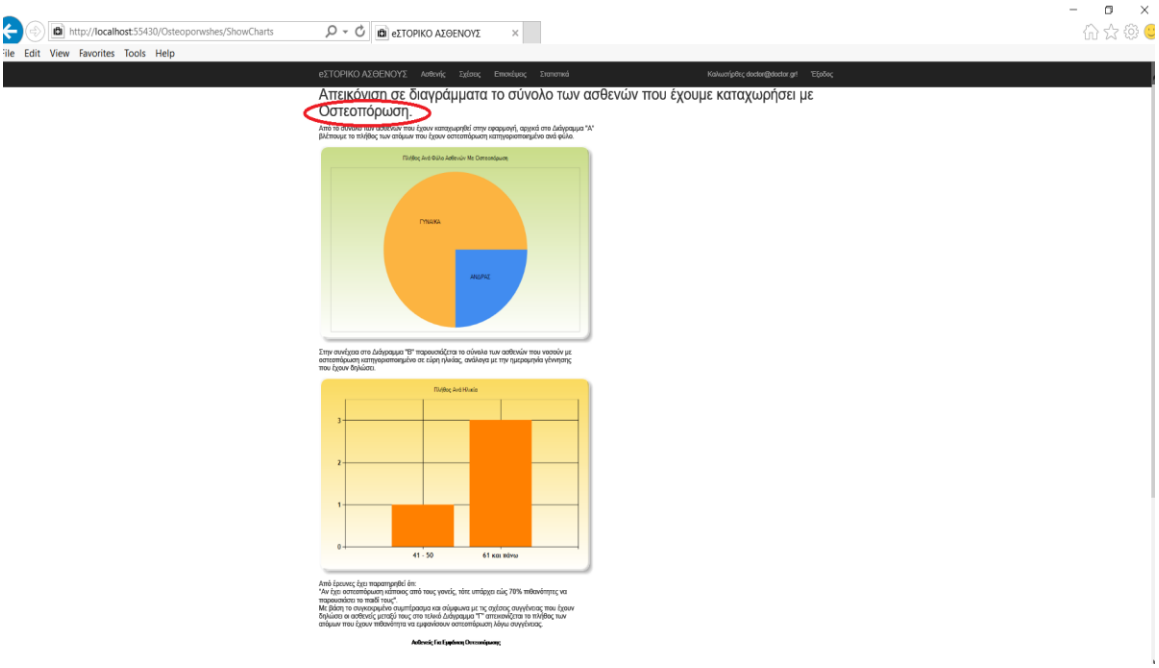


12.9 Στατιστικά για τα Καταχωρημένα Άτομα

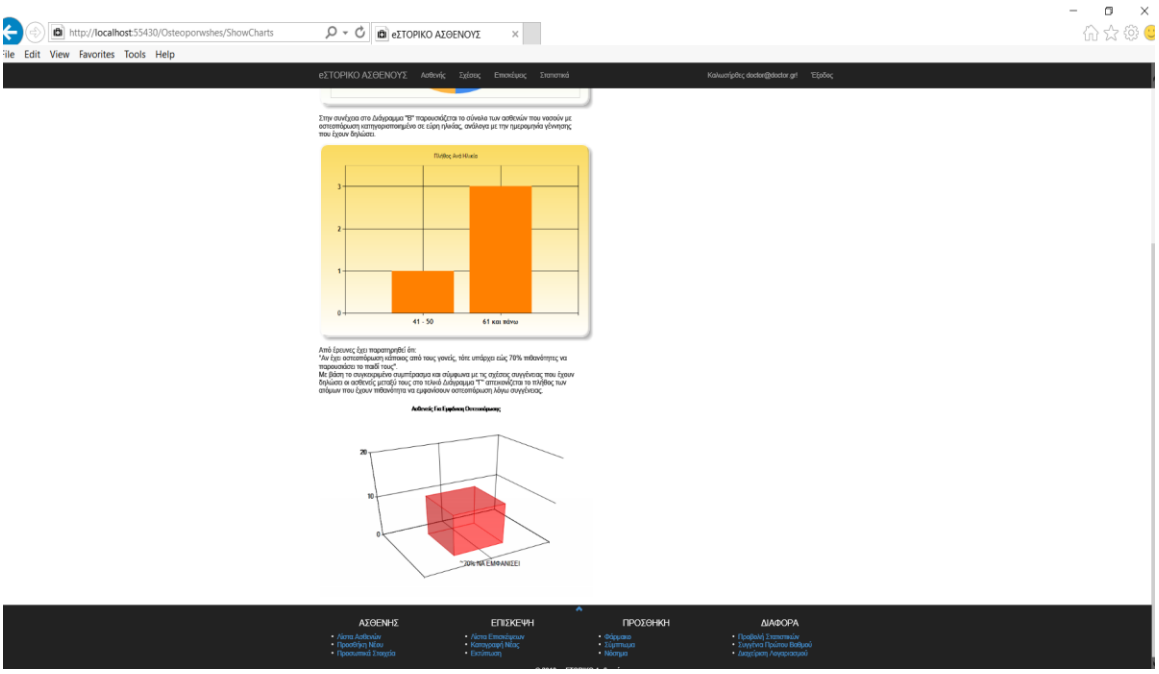
Η εφαρμογή μας δίνει την δυνατότητα με βάση πρόσφατες μελέτες και έρευνες για κάποιες κληρονομικές ασθένειες, να εξετάσει τις σχέσεις συγγένειας που έχουν δημιουργηθεί και να προβάλλει στατιστικά αλλά και μία εκτίμηση για πιθανότητες εμφάνισης ασθενειών σε άτομα που έχει εξετάσει ο γιατρός.



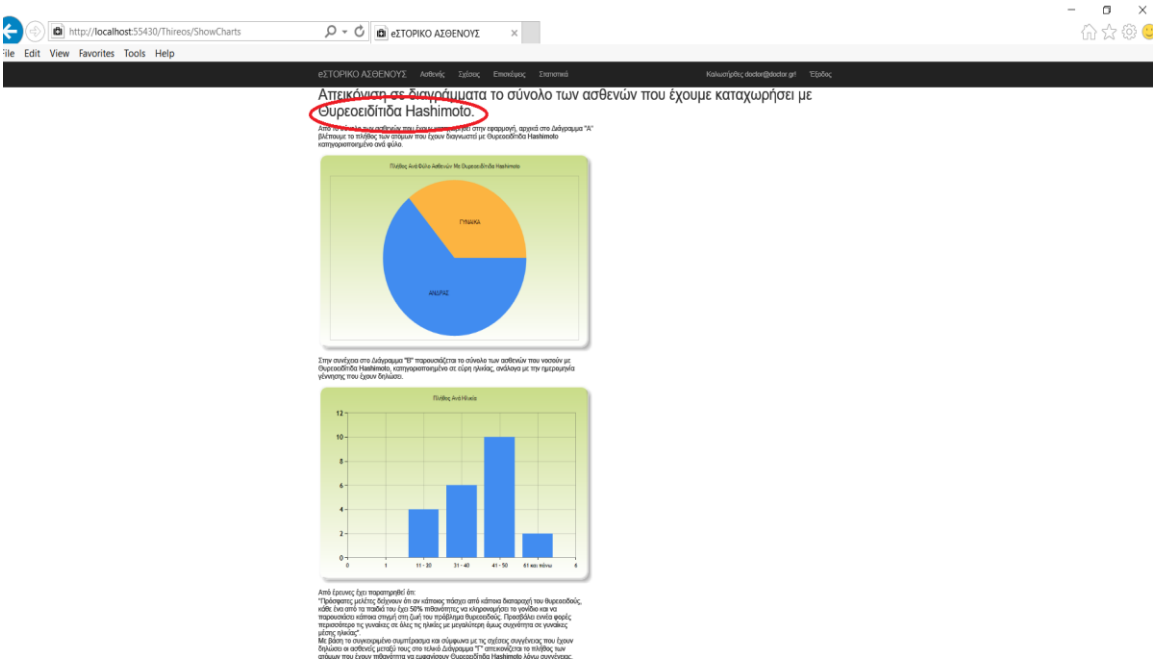
Αρχικά για την οστεοπόρωση προβάλλεται το πλήθος των ατόμων ομαδοποιημένο με βάση το φύλο τους. Στην συνέχεια με βάση το εύρος ηλικίας τους.



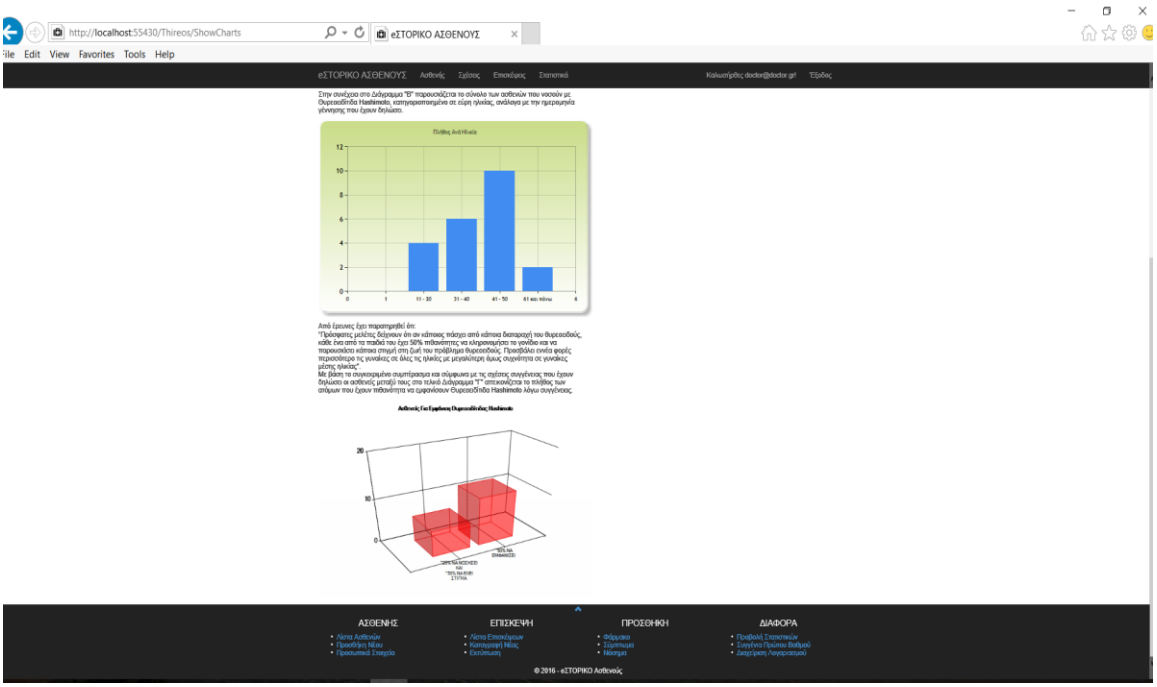
Τέλος προβάλλεται το πλήθος των ατόμων που λόγω ¨κακού¨ ιατρικού ιστορικού έχουν πιθανότητες να εμφανίσουν οστεοπόρωση.



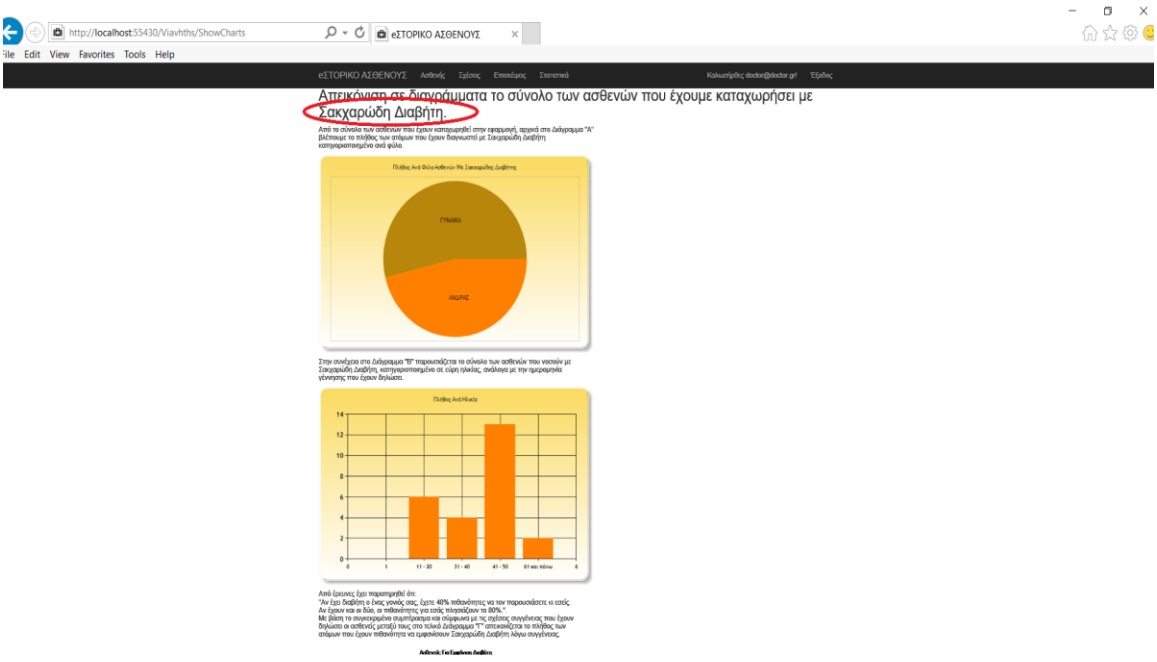
Στην συνέχεια για το αυτοάνοσο νόσημα της θυρεοειδίτιδας Hashimoto, προβάλλεται το πλήθος των ατόμων ομαδοποιημένο με βάση το φύλο τους και από κάτω με βάση το εύρος ηλικίας τους.



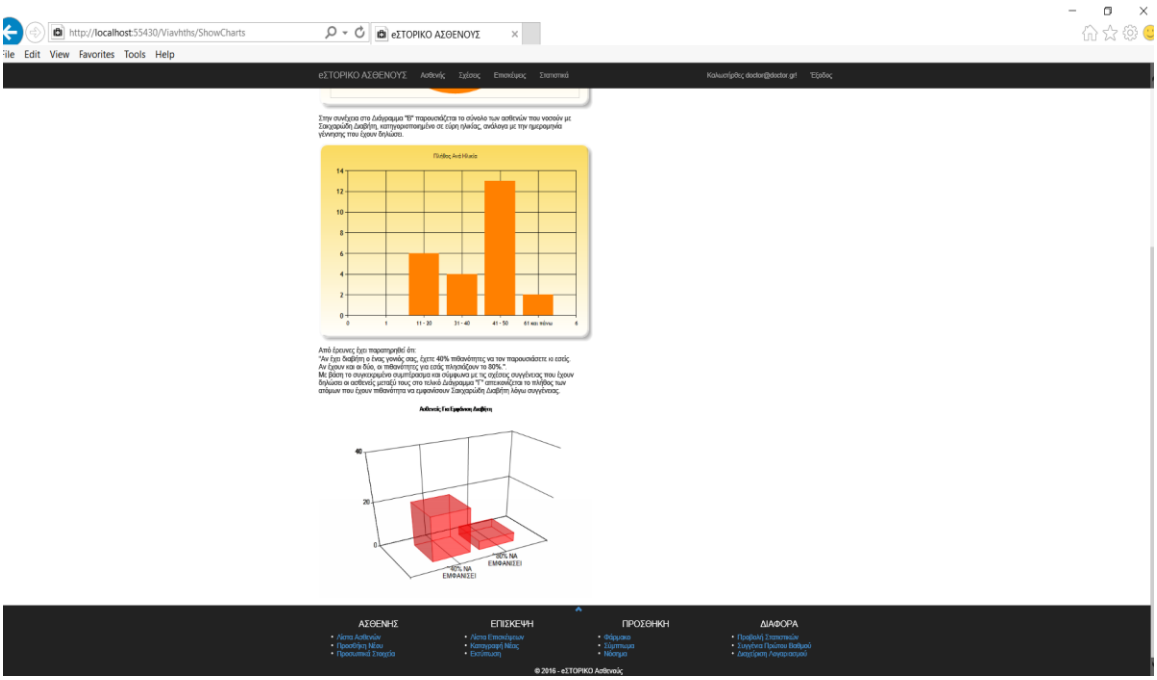
Τέλος προβάλλεται το πλήθος των ατόμων που λόγω ¨κακού¨ ιατρικού ιστορικού έχουν πιθανότητες να εμφανίσουν θυρεοειδίτιδα Hashimoto.



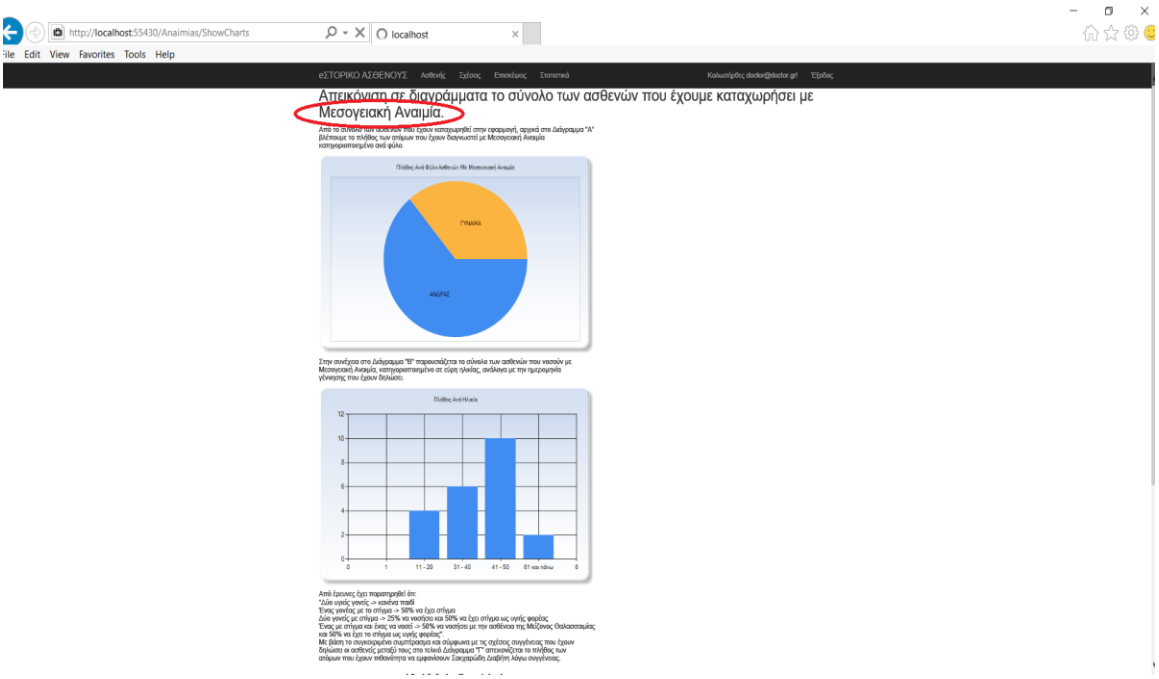
Στην συνέχεια για το νόσημα του Σακχαρώδη Διαβήτη, προβάλλεται το πλήθος των ατόμων ομαδοποιημένο με βάση το φύλο τους και από κάτω με βάση το εύρος ηλικίας τους.



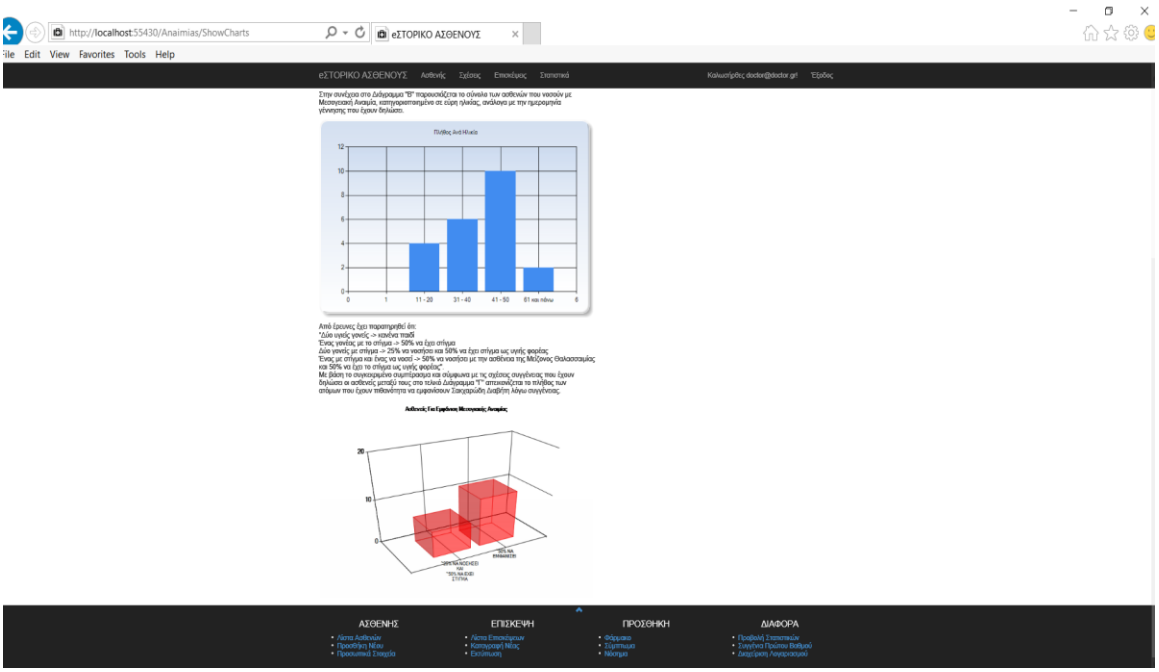
Τέλος προβάλλεται το πλήθος των ατόμων που λόγω ¨κακού¨ ιατρικού ιστορικού έχουν πιθανότητες να εμφανίσουν Σακχαρώδη Διαβήτη.



Στην συνέχεια για το νόσημα της Μεσογειακής Αναιμίας, προβάλλεται το πλήθος των ατόμων ομαδοποιημένο με βάση το φύλο τους και από κάτω με βάση το εύρος ηλικίας τους.



Τέλος προβάλλεται το πλήθος των ατόμων που λόγω ¨κακού¨ ιατρικού ιστορικού έχουν πιθανότητες να εμφανίσουν Μεσογειακή Αναιμία.



13 ΠΑΡΑΡΤΗΜΑΤΑ

13.1 Παράρτημα Α’

Παρακάτω παρατίθενται οι εντολές SQL από το αρχείο scriptsPatientHistory.sql για την δημιουργία όλων των αντικειμένων της βάσης δεδομένων μας.

```
USE [master]
GO
/****** Object:  Database [PatientHistory]    Script Date: 18-Nov-16 4:07:31 PM ******/
CREATE DATABASE [PatientHistory]
 CONTAINMENT = NONE
 ON  PRIMARY
( NAME = N'PatientHistory', FILENAME = N'C:\Program Files\Microsoft SQL
Server\MSSQL12.MSSQLSERVER\MSSQL\DATA\PatientHistory.mdf' , SIZE = 10240KB , MAXSIZE =
UNLIMITED, FILEGROWTH = 1024KB )
 LOG ON
( NAME = N'PatientHistory_log', FILENAME = N'C:\Program Files\Microsoft SQL
Server\MSSQL12.MSSQLSERVER\MSSQL\DATA\PatientHistory_log.ldf' , SIZE = 112384KB , MAXSIZE =
2048GB , FILEGROWTH = 10%)
GO
ALTER DATABASE [PatientHistory] SET COMPATIBILITY_LEVEL = 120
GO
IF (1 = FULLTEXTSERVICEPROPERTY('IsFullTextInstalled'))
begin
EXEC [PatientHistory].[dbo].[sp_fulltext_database] @action = 'enable'
end
GO
ALTER DATABASE [PatientHistory] SET ANSI_NULL_DEFAULT OFF
GO
ALTER DATABASE [PatientHistory] SET ANSI_NULLS OFF
GO
ALTER DATABASE [PatientHistory] SET ANSI_PADDING OFF
GO
ALTER DATABASE [PatientHistory] SET ANSI_WARNINGS OFF
GO
ALTER DATABASE [PatientHistory] SET ARITHABORT OFF
GO
ALTER DATABASE [PatientHistory] SET AUTO_CLOSE OFF
GO
ALTER DATABASE [PatientHistory] SET AUTO_SHRINK OFF
GO
ALTER DATABASE [PatientHistory] SET AUTO_UPDATE_STATISTICS ON
GO
ALTER DATABASE [PatientHistory] SET CURSOR_CLOSE_ON_COMMIT OFF
GO
ALTER DATABASE [PatientHistory] SET CURSOR_DEFAULT  GLOBAL
GO
ALTER DATABASE [PatientHistory] SET CONCAT_NULL_YIELDS_NULL OFF
GO
ALTER DATABASE [PatientHistory] SET NUMERIC_ROUNDABORT OFF
GO
ALTER DATABASE [PatientHistory] SET QUOTED_IDENTIFIER OFF
GO
ALTER DATABASE [PatientHistory] SET RECURSIVE_TRIGGERS OFF
GO
```

```
ALTER DATABASE [PatientHistory] SET  DISABLE_BROKER
GO
ALTER DATABASE [PatientHistory] SET AUTO_UPDATE_STATISTICS_ASYNC OFF
GO
ALTER DATABASE [PatientHistory] SET DATE_CORRELATION_OPTIMIZATION OFF
GO
ALTER DATABASE [PatientHistory] SET TRUSTWORTHY OFF
GO
ALTER DATABASE [PatientHistory] SET ALLOW_SNAPSHOT_ISOLATION OFF
GO
ALTER DATABASE [PatientHistory] SET PARAMETERIZATION SIMPLE
GO
ALTER DATABASE [PatientHistory] SET READ_COMMITTED_SNAPSHOT OFF
GO
ALTER DATABASE [PatientHistory] SET HONOR_BROKER_PRIORITY OFF
GO
ALTER DATABASE [PatientHistory] SET RECOVERY FULL
GO
ALTER DATABASE [PatientHistory] SET  MULTI_USER
GO
ALTER DATABASE [PatientHistory] SET PAGE_VERIFY CHECKSUM
GO
ALTER DATABASE [PatientHistory] SET DB_CHAINING OFF
GO
ALTER DATABASE [PatientHistory] SET FILESTREAM( NON_TRANSACTED_ACCESS = OFF )
GO
ALTER DATABASE [PatientHistory] SET TARGET_RECOVERY_TIME = 0 SECONDS
GO
ALTER DATABASE [PatientHistory] SET DELAYED_DURABILITY = DISABLED
GO
EXEC sys.sp_db_vardecimal_storage_format N'PatientHistory', N'ON'
GO
USE [PatientHistory]
GO
/****** Object:  Table [dbo].[__MigrationHistory]    Script Date: 18-Nov-16 4:07:31 PM ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_PADDING ON
GO
CREATE TABLE [dbo].[__MigrationHistory](
	[MigrationId] [nvarchar](150) NOT NULL,
	[ContextKey] [nvarchar](300) NOT NULL,
	[Model] [varbinary](max) NOT NULL,
	[ProductVersion] [nvarchar](32) NOT NULL,
 CONSTRAINT [PK_dbo.__MigrationHistory] PRIMARY KEY CLUSTERED
(
	[MigrationId] ASC,
	[ContextKey] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]

GO
SET ANSI_PADDING OFF
```

```
GO
/****** Object:  Table [dbo].[Allergy]    Script Date: 18-Nov-16 4:07:31 PM ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Allergy](
	[AllergyID] [int] IDENTITY(1,1) NOT NULL,
	[PersonId] [int] NOT NULL,
	[NameAllergy] [nvarchar](50) NULL,
	[TypeAllergy] [nvarchar](max) NULL,
	[DescrAllergy] [nvarchar](max) NULL,
 CONSTRAINT [PK_Allergy] PRIMARY KEY CLUSTERED
(
	[AllergyID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]

GO
/****** Object:  Table [dbo].[Anaimia]    Script Date: 18-Nov-16 4:07:31 PM ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Anaimia](
	[AnaimiaID] [int] IDENTITY(1,1) NOT NULL,
	[MumId] [int] NULL,
	[MumHas] [nvarchar](max) NULL,
	[DadId] [int] NULL,
	[DadHas] [nvarchar](max) NULL,
	[KidId] [int] NOT NULL,
	[KidHas] [nvarchar](max) NULL,
	[PososTOCHILDREN] [nvarchar](max) NULL,
 CONSTRAINT [PK_Anaimia] PRIMARY KEY CLUSTERED
(
	[AnaimiaID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]

GO
/****** Object:  Table [dbo].[AspNetRoles]    Script Date: 18-Nov-16 4:07:31 PM ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[AspNetRoles](
	[Id] [nvarchar](128) NOT NULL,
	[Name] [nvarchar](256) NOT NULL,
 CONSTRAINT [PK_dbo.AspNetRoles] PRIMARY KEY CLUSTERED
(
	[Id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
```

```
GO
/****** Object:  Table [dbo].[AspNetUserClaims]    Script Date: 18-Nov-16 4:07:31 PM ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[AspNetUserClaims](
	[Id] [int] IDENTITY(1,1) NOT NULL,
	[UserId] [nvarchar](128) NOT NULL,
	[ClaimType] [nvarchar](max) NULL,
	[ClaimValue] [nvarchar](max) NULL,
 CONSTRAINT [PK_dbo.AspNetUserClaims] PRIMARY KEY CLUSTERED
(
	[Id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]

GO
/****** Object:  Table [dbo].[AspNetUserLogins]    Script Date: 18-Nov-16 4:07:31 PM ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[AspNetUserLogins](
	[LoginProvider] [nvarchar](128) NOT NULL,
	[ProviderKey] [nvarchar](128) NOT NULL,
	[UserId] [nvarchar](128) NOT NULL,
 CONSTRAINT [PK_dbo.AspNetUserLogins] PRIMARY KEY CLUSTERED
(
	[LoginProvider] ASC,
	[ProviderKey] ASC,
	[UserId] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]

GO
/****** Object:  Table [dbo].[AspNetUserRoles]    Script Date: 18-Nov-16 4:07:31 PM ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[AspNetUserRoles](
	[UserId] [nvarchar](128) NOT NULL,
	[RoleId] [nvarchar](128) NOT NULL,
 CONSTRAINT [PK_dbo.AspNetUserRoles] PRIMARY KEY CLUSTERED
(
	[UserId] ASC,
	[RoleId] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]

GO
```

```
/****** Object:  Table [dbo].[AspNetUsers]    Script Date: 18-Nov-16 4:07:31 PM ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[AspNetUsers](
	[Id] [nvarchar](128) NOT NULL,
	[Email] [nvarchar](256) NULL,
	[EmailConfirmed] [bit] NOT NULL,
	[PasswordHash] [nvarchar](max) NULL,
	[SecurityStamp] [nvarchar](max) NULL,
	[PhoneNumber] [nvarchar](max) NULL,
	[PhoneNumberConfirmed] [bit] NOT NULL,
	[TwoFactorEnabled] [bit] NOT NULL,
	[LockoutEndDateUtc] [datetime] NULL,
	[LockoutEnabled] [bit] NOT NULL,
	[AccessFailedCount] [int] NOT NULL,
	[UserName] [nvarchar](256) NOT NULL,
 CONSTRAINT [PK_dbo.AspNetUsers] PRIMARY KEY CLUSTERED
(
	[Id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]

GO
/****** Object:  Table [dbo].[DiaforikhDiagn]    Script Date: 18-Nov-16 4:07:31 PM ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[DiaforikhDiagn](
	[DiaforikhDiagnID] [int] IDENTITY(1,1) NOT NULL,
	[ReasonVisitId] [int] NOT NULL,
	[DiaforikhDescr] [nvarchar](max) NULL,
 CONSTRAINT [PK_DiaforikhDiagn] PRIMARY KEY CLUSTERED
(
	[DiaforikhDiagnID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]

GO
/****** Object:  Table [dbo].[DiseaseAdult]    Script Date: 18-Nov-16 4:07:31 PM ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[DiseaseAdult](
	[DiseaseAdultID] [int] IDENTITY(1,1) NOT NULL,
	[PersonId] [int] NOT NULL,
	[DescriptionAdultDisease] [nvarchar](max) NOT NULL,
 CONSTRAINT [PK_DiseaseAdult] PRIMARY KEY CLUSTERED
(
	[DiseaseAdultID] ASC
```

```
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]

GO
/****** Object:  Table [dbo].[DiseaseChild]    Script Date: 18-Nov-16 4:07:31 PM ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[DiseaseChild](
	[DiseaseChildID] [int] IDENTITY(1,1) NOT NULL,
	[PersonId] [int] NOT NULL,
	[DescriptionChildDisease] [nvarchar](max) NOT NULL,
 CONSTRAINT [PK_DiseaseChild] PRIMARY KEY CLUSTERED
(
	[DiseaseChildID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]

GO
/****** Object:  Table [dbo].[Family]    Script Date: 18-Nov-16 4:07:31 PM ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Family](
	[FamilyID] [int] IDENTITY(1,1) NOT NULL,
	[MotherId] [int] NULL,
	[FatherId] [int] NULL,
	[ChildId] [int] NOT NULL,
 CONSTRAINT [PK_Family] PRIMARY KEY CLUSTERED
(
	[FamilyID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]

GO
/****** Object:  Table [dbo].[FamilyHierarchy]    Script Date: 18-Nov-16 4:07:31 PM ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[FamilyHierarchy](
	[FamilyId] [int] IDENTITY(1,1) NOT NULL,
	[MemberId] [int] NOT NULL,
	[ParentId] [int] NOT NULL,
	[Name] [nvarchar](max) NULL,
	[Parent2] [int] NOT NULL,
 CONSTRAINT [PK_dbo.FamilyHierarchy] PRIMARY KEY CLUSTERED
(
	[FamilyId] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
```

```
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]

GO
/****** Object:  Table [dbo].[Habbit]    Script Date: 18-Nov-16 4:07:31 PM ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Habbit](
	[HabbitUNQ] [int] IDENTITY(1,1) NOT NULL,
	[PersonId] [int] NOT NULL,
	[Smoking] [nvarchar](50) NULL,
	[CigaretteDay] [nvarchar](50) NULL,
	[CigaretteYears] [nvarchar](50) NULL,
	[Alcohol] [nvarchar](50) NULL,
	[AlcoholDay] [nvarchar](50) NULL,
	[AlcoholWeek] [nvarchar](50) NULL,
	[WriteHand] [nvarchar](50) NULL,
	[Education] [nvarchar](50) NULL,
	[Sports] [nvarchar](50) NULL,
	[Pets] [nvarchar](50) NULL,
	[FamilyStatus] [nvarchar](50) NULL,
	[JobName] [nvarchar](max) NULL,
	[PacketYearCigaret] [nvarchar](50) NULL,
 CONSTRAINT [PK_Habbit_1] PRIMARY KEY CLUSTERED
(
	[PersonId] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]

GO
/****** Object:  Table [dbo].[MedicalProblem]    Script Date: 18-Nov-16 4:07:31 PM ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[MedicalProblem](
	[MedicalProblemID] [int] IDENTITY(1,1) NOT NULL,
	[NameMedicProblem] [nvarchar](max) NOT NULL,
 CONSTRAINT [PK_MedicalProblem] PRIMARY KEY CLUSTERED
(
	[MedicalProblemID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]

GO
/****** Object:  Table [dbo].[MedicalProblemPersonVisit]    Script Date: 18-Nov-16 4:07:31 PM ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[MedicalProblemPersonVisit](
	[MedicalProblemPersonVisitID] [int] IDENTITY(1,1) NOT NULL,
	[MedicalProblemId] [int] NOT NULL,
```

```
	[ReasonVisitId] [int] NOT NULL,
	[DescrMedicProblem] [nvarchar](max) NULL,
 CONSTRAINT [PK_MedicalProblemPersonVisit] PRIMARY KEY CLUSTERED
(
	[MedicalProblemPersonVisitID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]

GO
/****** Object:  Table [dbo].[Osteoporwsh]    Script Date: 18-Nov-16 4:07:31 PM ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Osteoporwsh](
	[OsteoporwshID] [int] IDENTITY(1,1) NOT NULL,
	[MumId] [int] NULL,
	[MumHas] [nvarchar](max) NULL,
	[DadId] [int] NULL,
	[DadHas] [nvarchar](max) NULL,
	[KidId] [int] NOT NULL,
	[KidHas] [nvarchar](max) NULL,
	[PososTOCHILDREN] [nvarchar](max) NULL,
 CONSTRAINT [PK_Osteoporwsh] PRIMARY KEY CLUSTERED
(
	[OsteoporwshID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]

GO
/****** Object:  Table [dbo].[ParousaNoso]    Script Date: 18-Nov-16 4:07:31 PM ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[ParousaNoso](
	[ParousaNosoUNQ] [int] IDENTITY(1,1) NOT NULL,
	[ReasonVisitId] [int] NOT NULL,
	[Sfykseis] [nvarchar](50) NULL,
	[Thermokrasia] [decimal](5, 2) NULL,
	[Baros] [decimal](5, 2) NULL,
	[Ypsos] [decimal](5, 2) NULL,
	[ExtraDescr] [nvarchar](max) NULL,
 CONSTRAINT [PK_ParousaNoso] PRIMARY KEY CLUSTERED
(
	[ReasonVisitId] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]

GO
/****** Object:  Table [dbo].[PastSituation]    Script Date: 18-Nov-16 4:07:31 PM ******/
SET ANSI_NULLS ON
GO
```

```
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[PastSituation](
	[PastSituationID] [int] IDENTITY(1,1) NOT NULL,
	[SitDescr] [nvarchar](max) NOT NULL,
 CONSTRAINT [PK_PastSituation] PRIMARY KEY CLUSTERED
(
	[PastSituationID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]

GO
/****** Object:  Table [dbo].[PatientsSymptom]    Script Date: 18-Nov-16 4:07:31 PM ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[PatientsSymptom](
	[PatientsSymptomID] [int] IDENTITY(1,1) NOT NULL,
	[SymptomaId] [int] NOT NULL,
	[ReasonVisitId] [int] NOT NULL,
	[WhereIs] [nvarchar](max) NULL,
	[HowFeel] [nvarchar](max) NULL,
	[HowMuch] [nvarchar](max) NULL,
	[WhenStart] [nvarchar](max) NULL,
	[BetterWhen] [nvarchar](max) NULL,
 CONSTRAINT [PK_PatientsSymptom] PRIMARY KEY CLUSTERED
(
	[PatientsSymptomID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]

GO
/****** Object:  Table [dbo].[Person]    Script Date: 18-Nov-16 4:07:31 PM ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_PADDING ON
GO
CREATE TABLE [dbo].[Person](
	[PersonID] [int] IDENTITY(1,1) NOT NULL,
	[First_Name] [nvarchar](50) NOT NULL,
	[Last_Name] [nvarchar](50) NOT NULL,
	[Sex] [nvarchar](50) NOT NULL,
	[Birth_Date] [datetime] NULL,
	[Birth_City] [nvarchar](50) NULL,
	[Town_Live] [nvarchar](50) NULL,
	[Adreess_Live] [nvarchar](50) NULL,
	[Phone] [nvarchar](max) NULL,
	[Mobile] [nvarchar](max) NULL,
	[Mail] [nvarchar](max) NULL,
	[Death_Date] [datetime] NULL,
	[WhyDied] [nvarchar](max) NULL,
```

```
	[Country_Live] [nvarchar](50) NULL,
	[userCreator] [nvarchar](max) NOT NULL,
	[amka] [varchar](50) NULL,
 CONSTRAINT [PK_dbo.Person] PRIMARY KEY CLUSTERED
(
	[PersonID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]

GO
SET ANSI_PADDING OFF
GO
/****** Object:  Table [dbo].[Pharmaka]    Script Date: 18-Nov-16 4:07:31 PM ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Pharmaka](
	[PharmakoID] [int] IDENTITY(1,1) NOT NULL,
	[PersonId] [int] NOT NULL,
	[NamePha] [nvarchar](50) NOT NULL,
	[DosologiaPha] [nvarchar](50) NULL,
	[StartPha] [datetime] NULL,
	[ReasonPha] [nvarchar](max) NULL,
	[Pausi] [nvarchar](10) NULL,
	[Parenergeies] [nvarchar](max) NULL,
 CONSTRAINT [PK_Pharmaka] PRIMARY KEY CLUSTERED
(
	[PharmakoID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]

GO
/****** Object:  Table [dbo].[ReasonVisit]    Script Date: 18-Nov-16 4:07:31 PM ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[ReasonVisit](
	[ReasonVisitID] [int] IDENTITY(1,1) NOT NULL,
	[PersonId] [int] NOT NULL,
	[ReasonDescr] [nvarchar](max) NULL,
	[DateVisit] [datetime] NOT NULL,
 CONSTRAINT [PK_ReasonVisit] PRIMARY KEY CLUSTERED
(
	[ReasonVisitID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]

GO
/****** Object:  Table [dbo].[Relationship]    Script Date: 18-Nov-16 4:07:31 PM ******/
SET ANSI_NULLS ON
GO
```

```
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Relationship](
	[RelationshipID] [int] IDENTITY(1,1) NOT NULL,
	[Person1Id] [int] NOT NULL,
	[Person2Id] [int] NOT NULL,
	[RelatypeId] [int] NOT NULL,
 CONSTRAINT [PK_dbo.Relationship] PRIMARY KEY CLUSTERED
(
	[RelationshipID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]

GO
/****** Object:  Table [dbo].[Relatype]    Script Date: 18-Nov-16 4:07:31 PM ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Relatype](
	[RelatypeID] [int] IDENTITY(1,1) NOT NULL,
	[Type] [nvarchar](max) NOT NULL,
	[Group] [nvarchar](max) NOT NULL,
 CONSTRAINT [PK_dbo.Relatype] PRIMARY KEY CLUSTERED
(
	[RelatypeID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]

GO
/****** Object:  Table [dbo].[SituationsPerson]    Script Date: 18-Nov-16 4:07:31 PM ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[SituationsPerson](
	[SituationsPersonID] [int] IDENTITY(1,1) NOT NULL,
	[PersonId] [int] NOT NULL,
	[PastSituationId] [int] NOT NULL,
	[Notes] [nvarchar](max) NULL,
 CONSTRAINT [PK_SituationsPerson] PRIMARY KEY CLUSTERED
(
	[SituationsPersonID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]

GO
/****** Object:  Table [dbo].[Surgery]    Script Date: 18-Nov-16 4:07:31 PM ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Surgery](
```

```
	[SurgeryID] [int] IDENTITY(1,1) NOT NULL,
	[PersonId] [int] NOT NULL,
	[DescriptionSurgery] [nvarchar](max) NOT NULL,
 CONSTRAINT [PK_Surgery] PRIMARY KEY CLUSTERED
(
	[SurgeryID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]

GO
/****** Object:  Table [dbo].[Symptomata]    Script Date: 18-Nov-16 4:07:31 PM ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Symptomata](
	[SymptomaID] [int] IDENTITY(1,1) NOT NULL,
	[Description] [nvarchar](50) NOT NULL,
 CONSTRAINT [PK_Symptomata] PRIMARY KEY CLUSTERED
(
	[SymptomaID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]

GO
/****** Object:  Table [dbo].[TelikhDiagn]    Script Date: 18-Nov-16 4:07:31 PM ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[TelikhDiagn](
	[TelikhDiagnID] [int] IDENTITY(1,1) NOT NULL,
	[ReasonVisitId] [int] NOT NULL,
	[TelikhDescr] [nvarchar](max) NULL,
 CONSTRAINT [PK_TelikhDiagn] PRIMARY KEY CLUSTERED
(
	[TelikhDiagnID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]

GO
/****** Object:  Table [dbo].[Thireo]    Script Date: 18-Nov-16 4:07:31 PM ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Thireo](
	[ThireoID] [int] IDENTITY(1,1) NOT NULL,
	[MumId] [int] NULL,
	[MumHas] [nvarchar](max) NULL,
	[DadId] [int] NULL,
	[DadHas] [nvarchar](max) NULL,
	[KidId] [int] NOT NULL,
```

```
	[KidHas] [nvarchar](max) NULL,
	[PososotoCHILDREN] [nvarchar](max) NULL,
 CONSTRAINT [PK_Thireo] PRIMARY KEY CLUSTERED
(
	[ThireoID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]

GO
/****** Object:  Table [dbo].[tmp]    Script Date: 18-Nov-16 4:07:31 PM ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[tmp](
	[fname] [nvarchar](50) NOT NULL
) ON [PRIMARY]

GO
/****** Object:  Table [dbo].[Viavhths]    Script Date: 18-Nov-16 4:07:31 PM ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Viavhths](
	[ViavhthsID] [int] IDENTITY(1,1) NOT NULL,
	[MumId] [int] NULL,
	[MumHas] [nvarchar](max) NULL,
	[DadId] [int] NULL,
	[DadHas] [nvarchar](max) NULL,
	[KidId] [int] NOT NULL,
	[KidHas] [nvarchar](max) NULL,
	[PososotoCHILDREN] [nvarchar](max) NULL,
 CONSTRAINT [PK_Viavhths] PRIMARY KEY CLUSTERED
(
	[ViavhthsID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]

GO
/****** Object:  View [dbo].[AgePersonNoshmaView]    Script Date: 18-Nov-16 4:07:31 PM ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE VIEW [dbo].[AgePersonNoshmaView]
AS
SELECT dbo.Person.PersonID, dbo.Person.Sex, (CONVERT(int, CONVERT(char(8), GETDATE(), 112)) -
CONVERT(char(8), dbo.Person.Birth_Date, 112)) / 10000 AS AGE, dbo.PastSituation.SitDescr
FROM	dbo.PastSituation INNER JOIN
		dbo.SituationsPerson ON dbo.PastSituation.PastSituationID = dbo.SituationsPerson.PastSituationId
INNER JOIN
		dbo.Person ON dbo.SituationsPerson.PersonId = dbo.Person.PersonID
```

```
GO
/****** Object:  View [dbo].[NoshmataPersonView]    Script Date: 18-Nov-16 4:07:31 PM ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE VIEW [dbo].[NoshmataPersonView]
AS
SELECT dbo.PastSituation.SitDescr, dbo.SituationsPerson.PersonId
FROM     dbo.PastSituation INNER JOIN
             dbo.SituationsPerson ON dbo.PastSituation.PastSituationID = dbo.SituationsPerson.PastSituationId

GO
SET ANSI_PADDING ON

GO
/****** Object:  Index [RoleNameIndex]    Script Date: 18-Nov-16 4:07:31 PM ******/
CREATE UNIQUE NONCLUSTERED INDEX [RoleNameIndex] ON [dbo].[AspNetRoles]
(
	[Name] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, SORT_IN_TEMPDB = OFF,
IGNORE_DUP_KEY = OFF, DROP_EXISTING = OFF, ONLINE = OFF, ALLOW_ROW_LOCKS = ON,
ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
GO
SET ANSI_PADDING ON

GO
/****** Object:  Index [IX_UserId]    Script Date: 18-Nov-16 4:07:31 PM ******/
CREATE NONCLUSTERED INDEX [IX_UserId] ON [dbo].[AspNetUserClaims]
(
	[UserId] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, SORT_IN_TEMPDB = OFF,
DROP_EXISTING = OFF, ONLINE = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON)
ON [PRIMARY]
GO
SET ANSI_PADDING ON

GO
/****** Object:  Index [IX_UserId]    Script Date: 18-Nov-16 4:07:31 PM ******/
CREATE NONCLUSTERED INDEX [IX_UserId] ON [dbo].[AspNetUserLogins]
(
	[UserId] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, SORT_IN_TEMPDB = OFF,
DROP_EXISTING = OFF, ONLINE = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON)
ON [PRIMARY]
GO
SET ANSI_PADDING ON

GO
/****** Object:  Index [IX_RoleId]    Script Date: 18-Nov-16 4:07:31 PM ******/
CREATE NONCLUSTERED INDEX [IX_RoleId] ON [dbo].[AspNetUserRoles]
(
	[RoleId] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, SORT_IN_TEMPDB = OFF,
DROP_EXISTING = OFF, ONLINE = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON)
ON [PRIMARY]
```

```
GO
SET ANSI_PADDING ON

GO
/****** Object:  Index [IX_UserId]    Script Date: 18-Nov-16 4:07:31 PM ******/
CREATE NONCLUSTERED INDEX [IX_UserId] ON [dbo].[AspNetUserRoles]
(
	[UserId] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, SORT_IN_TEMPDB = OFF,
DROP_EXISTING = OFF, ONLINE = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON)
ON [PRIMARY]
GO
SET ANSI_PADDING ON

GO
/****** Object:  Index [UserNameIndex]    Script Date: 18-Nov-16 4:07:31 PM ******/
CREATE UNIQUE NONCLUSTERED INDEX [UserNameIndex] ON [dbo].[AspNetUsers]
(
	[UserName] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, SORT_IN_TEMPDB = OFF,
IGNORE_DUP_KEY = OFF, DROP_EXISTING = OFF, ONLINE = OFF, ALLOW_ROW_LOCKS = ON,
ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
GO
/****** Object:  Index [IX_Person1Id]    Script Date: 18-Nov-16 4:07:31 PM ******/
CREATE NONCLUSTERED INDEX [IX_Person1Id] ON [dbo].[Relationship]
(
	[Person1Id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, SORT_IN_TEMPDB = OFF,
DROP_EXISTING = OFF, ONLINE = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON)
ON [PRIMARY]
GO
/****** Object:  Index [IX_Person2Id]    Script Date: 18-Nov-16 4:07:31 PM ******/
CREATE NONCLUSTERED INDEX [IX_Person2Id] ON [dbo].[Relationship]
(
	[Person2Id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, SORT_IN_TEMPDB = OFF,
DROP_EXISTING = OFF, ONLINE = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON)
ON [PRIMARY]
GO
/****** Object:  Index [IX_RelatypeId]    Script Date: 18-Nov-16 4:07:31 PM ******/
CREATE NONCLUSTERED INDEX [IX_RelatypeId] ON [dbo].[Relationship]
(
	[RelatypeId] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, SORT_IN_TEMPDB = OFF,
DROP_EXISTING = OFF, ONLINE = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON)
ON [PRIMARY]
GO
ALTER TABLE [dbo].[Allergy]  WITH CHECK ADD  CONSTRAINT [FK_Allergy_Person] FOREIGN
KEY([PersonId])
REFERENCES [dbo].[Person] ([PersonID])
ON DELETE CASCADE
GO
ALTER TABLE [dbo].[Allergy] CHECK CONSTRAINT [FK_Allergy_Person]
GO
ALTER TABLE [dbo].[Anaimia]  WITH CHECK ADD  CONSTRAINT [FK_Anaimia_Person_Dad]
FOREIGN KEY([DadId])
```

```
REFERENCES [dbo].[Person] ([PersonID])
GO
ALTER TABLE [dbo].[Anaimia] CHECK CONSTRAINT [FK_Anaimia_Person_Dad]
GO
ALTER TABLE [dbo].[Anaimia]  WITH CHECK ADD  CONSTRAINT [FK_Anaimia_Person_Kid]
FOREIGN KEY([KidId])
REFERENCES [dbo].[Person] ([PersonID])
GO
ALTER TABLE [dbo].[Anaimia] CHECK CONSTRAINT [FK_Anaimia_Person_Kid]
GO
ALTER TABLE [dbo].[Anaimia]  WITH CHECK ADD  CONSTRAINT [FK_Anaimia_Person_Mum]
FOREIGN KEY([MumId])
REFERENCES [dbo].[Person] ([PersonID])
GO
ALTER TABLE [dbo].[Anaimia] CHECK CONSTRAINT [FK_Anaimia_Person_Mum]
GO
ALTER TABLE [dbo].[AspNetUserClaims]  WITH CHECK ADD  CONSTRAINT
[FK_dbo.AspNetUserClaims_dbo.AspNetUsers_UserId] FOREIGN KEY([UserId])
REFERENCES [dbo].[AspNetUsers] ([Id])
ON DELETE CASCADE
GO
ALTER TABLE [dbo].[AspNetUserClaims] CHECK CONSTRAINT
[FK_dbo.AspNetUserClaims_dbo.AspNetUsers_UserId]
GO
ALTER TABLE [dbo].[AspNetUserLogins]  WITH CHECK ADD  CONSTRAINT
[FK_dbo.AspNetUserLogins_dbo.AspNetUsers_UserId] FOREIGN KEY([UserId])
REFERENCES [dbo].[AspNetUsers] ([Id])
ON DELETE CASCADE
GO
ALTER TABLE [dbo].[AspNetUserLogins] CHECK CONSTRAINT
[FK_dbo.AspNetUserLogins_dbo.AspNetUsers_UserId]
GO
ALTER TABLE [dbo].[AspNetUserRoles]  WITH CHECK ADD  CONSTRAINT
[FK_dbo.AspNetUserRoles_dbo.AspNetRoles_RoleId] FOREIGN KEY([RoleId])
REFERENCES [dbo].[AspNetRoles] ([Id])
ON DELETE CASCADE
GO
ALTER TABLE [dbo].[AspNetUserRoles] CHECK CONSTRAINT
[FK_dbo.AspNetUserRoles_dbo.AspNetRoles_RoleId]
GO
ALTER TABLE [dbo].[AspNetUserRoles]  WITH CHECK ADD  CONSTRAINT
[FK_dbo.AspNetUserRoles_dbo.AspNetUsers_UserId] FOREIGN KEY([UserId])
REFERENCES [dbo].[AspNetUsers] ([Id])
ON DELETE CASCADE
GO
ALTER TABLE [dbo].[AspNetUserRoles] CHECK CONSTRAINT
[FK_dbo.AspNetUserRoles_dbo.AspNetUsers_UserId]
GO
ALTER TABLE [dbo].[DiaforikhDiagn]  WITH CHECK ADD  CONSTRAINT
[FK_DiaforikhDiagn_ReasonVisit] FOREIGN KEY([ReasonVisitId])
REFERENCES [dbo].[ReasonVisit] ([ReasonVisitID])
ON DELETE CASCADE
GO
ALTER TABLE [dbo].[DiaforikhDiagn] CHECK CONSTRAINT [FK_DiaforikhDiagn_ReasonVisit]
GO
```

```
ALTER TABLE [dbo].[DiseaseAdult]  WITH CHECK ADD  CONSTRAINT [FK_DiseaseAdult_Person]
FOREIGN KEY([PersonId])
REFERENCES [dbo].[Person] ([PersonID])
GO
ALTER TABLE [dbo].[DiseaseAdult] CHECK CONSTRAINT [FK_DiseaseAdult_Person]
GO
ALTER TABLE [dbo].[DiseaseChild]  WITH CHECK ADD  CONSTRAINT [FK_DiseaseChild_Person]
FOREIGN KEY([PersonId])
REFERENCES [dbo].[Person] ([PersonID])
GO
ALTER TABLE [dbo].[DiseaseChild] CHECK CONSTRAINT [FK_DiseaseChild_Person]
GO
ALTER TABLE [dbo].[Family]  WITH CHECK ADD  CONSTRAINT [FK_Family_Person_Child] FOREIGN
KEY([ChildId])
REFERENCES [dbo].[Person] ([PersonID])
GO
ALTER TABLE [dbo].[Family] CHECK CONSTRAINT [FK_Family_Person_Child]
GO
ALTER TABLE [dbo].[Family]  WITH CHECK ADD  CONSTRAINT [FK_Family_Person_Father]
FOREIGN KEY([FatherId])
REFERENCES [dbo].[Person] ([PersonID])
GO
ALTER TABLE [dbo].[Family] CHECK CONSTRAINT [FK_Family_Person_Father]
GO
ALTER TABLE [dbo].[Family]  WITH CHECK ADD  CONSTRAINT [FK_Family_Person_Mother]
FOREIGN KEY([MotherId])
REFERENCES [dbo].[Person] ([PersonID])
GO
ALTER TABLE [dbo].[Family] CHECK CONSTRAINT [FK_Family_Person_Mother]
GO
ALTER TABLE [dbo].[Habbit]  WITH CHECK ADD  CONSTRAINT [FK_Habbit_Person] FOREIGN
KEY([PersonId])
REFERENCES [dbo].[Person] ([PersonID])
ON DELETE CASCADE
GO
ALTER TABLE [dbo].[Habbit] CHECK CONSTRAINT [FK_Habbit_Person]
GO
ALTER TABLE [dbo].[MedicalProblemPersonVisit]  WITH CHECK ADD  CONSTRAINT
[FK_MedicalProblem_ReasonVisit] FOREIGN KEY([ReasonVisitId])
REFERENCES [dbo].[ReasonVisit] ([ReasonVisitID])
ON DELETE CASCADE
GO
ALTER TABLE [dbo].[MedicalProblemPersonVisit] CHECK CONSTRAINT
[FK_MedicalProblem_ReasonVisit]
GO
ALTER TABLE [dbo].[MedicalProblemPersonVisit]  WITH CHECK ADD  CONSTRAINT
[FK_MedicalProblemPersonVisit_MedicalProblem] FOREIGN KEY([MedicalProblemId])
REFERENCES [dbo].[MedicalProblem] ([MedicalProblemID])
GO
ALTER TABLE [dbo].[MedicalProblemPersonVisit] CHECK CONSTRAINT
[FK_MedicalProblemPersonVisit_MedicalProblem]
GO
ALTER TABLE [dbo].[Osteoporwsh]  WITH CHECK ADD  CONSTRAINT [FK_Osteoporwsh_Person_Dad]
FOREIGN KEY([DadId])
REFERENCES [dbo].[Person] ([PersonID])
GO
```

```
ALTER TABLE [dbo].[Osteoporwsh] CHECK CONSTRAINT [FK_Osteoporwsh_Person_Dad]
GO
ALTER TABLE [dbo].[Osteoporwsh]  WITH CHECK ADD  CONSTRAINT [FK_Osteoporwsh_Person_Kid]
FOREIGN KEY([KidId])
REFERENCES [dbo].[Person] ([PersonID])
GO
ALTER TABLE [dbo].[Osteoporwsh] CHECK CONSTRAINT [FK_Osteoporwsh_Person_Kid]
GO
ALTER TABLE [dbo].[Osteoporwsh]  WITH CHECK ADD  CONSTRAINT [FK_Osteoporwsh_Person_Mum]
FOREIGN KEY([MumId])
REFERENCES [dbo].[Person] ([PersonID])
GO
ALTER TABLE [dbo].[Osteoporwsh] CHECK CONSTRAINT [FK_Osteoporwsh_Person_Mum]
GO
ALTER TABLE [dbo].[ParousaNoso]  WITH CHECK ADD  CONSTRAINT [FK_ParousaNosos_ReasonVisit]
FOREIGN KEY([ReasonVisitId])
REFERENCES [dbo].[ReasonVisit] ([ReasonVisitID])
ON DELETE CASCADE
GO
ALTER TABLE [dbo].[ParousaNoso] CHECK CONSTRAINT [FK_ParousaNosos_ReasonVisit]
GO
ALTER TABLE [dbo].[PatientsSymptom]  WITH CHECK ADD  CONSTRAINT
[FK_PatientsSymptoms_ReasonVisit] FOREIGN KEY([ReasonVisitId])
REFERENCES [dbo].[ReasonVisit] ([ReasonVisitID])
ON DELETE CASCADE
GO
ALTER TABLE [dbo].[PatientsSymptom] CHECK CONSTRAINT [FK_PatientsSymptoms_ReasonVisit]
GO
ALTER TABLE [dbo].[PatientsSymptom]  WITH CHECK ADD  CONSTRAINT
[FK_PatientsSymptoms_Symptomata] FOREIGN KEY([SymptomaId])
REFERENCES [dbo].[Symptomata] ([SymptomaID])
GO
ALTER TABLE [dbo].[PatientsSymptom] CHECK CONSTRAINT [FK_PatientsSymptoms_Symptomata]
GO
ALTER TABLE [dbo].[Pharmaka]  WITH CHECK ADD  CONSTRAINT [FK_Pharmaka_Person] FOREIGN
KEY([PersonId])
REFERENCES [dbo].[Person] ([PersonID])
ON DELETE CASCADE
GO
ALTER TABLE [dbo].[Pharmaka] CHECK CONSTRAINT [FK_Pharmaka_Person]
GO
ALTER TABLE [dbo].[ReasonVisit]  WITH CHECK ADD  CONSTRAINT [FK_ReasonVisit_Person]
FOREIGN KEY([PersonId])
REFERENCES [dbo].[Person] ([PersonID])
ON DELETE CASCADE
GO
ALTER TABLE [dbo].[ReasonVisit] CHECK CONSTRAINT [FK_ReasonVisit_Person]
GO
ALTER TABLE [dbo].[Relationship]  WITH CHECK ADD  CONSTRAINT
[FK_dbo.Relationship_dbo.Person_Person1Id] FOREIGN KEY([Person1Id])
REFERENCES [dbo].[Person] ([PersonID])
GO
ALTER TABLE [dbo].[Relationship] CHECK CONSTRAINT [FK_dbo.Relationship_dbo.Person_Person1Id]
GO
ALTER TABLE [dbo].[Relationship]  WITH CHECK ADD  CONSTRAINT
[FK_dbo.Relationship_dbo.Person_Person2Id] FOREIGN KEY([Person2Id])
```

```
REFERENCES [dbo].[Person] ([PersonID])
GO
ALTER TABLE [dbo].[Relationship] CHECK CONSTRAINT [FK_dbo.Relationship_dbo.Person_Person2Id]
GO
ALTER TABLE [dbo].[Relationship]  WITH CHECK ADD  CONSTRAINT
[FK_dbo.Relationship_dbo.Relatype_RelatypeId] FOREIGN KEY([RelatypeId])
REFERENCES [dbo].[Relatype] ([RelatypeID])
GO
ALTER TABLE [dbo].[Relationship] CHECK CONSTRAINT [FK_dbo.Relationship_dbo.Relatype_RelatypeId]
GO
ALTER TABLE [dbo].[SituationsPerson]  WITH CHECK ADD  CONSTRAINT
[FK_SituationsPerson_PastSituation] FOREIGN KEY([PastSituationId])
REFERENCES [dbo].[PastSituation] ([PastSituationID])
GO
ALTER TABLE [dbo].[SituationsPerson] CHECK CONSTRAINT [FK_SituationsPerson_PastSituation]
GO
ALTER TABLE [dbo].[SituationsPerson]  WITH CHECK ADD  CONSTRAINT [FK_SituationsPerson_Person]
FOREIGN KEY([PersonId])
REFERENCES [dbo].[Person] ([PersonID])
ON DELETE CASCADE
GO
ALTER TABLE [dbo].[SituationsPerson] CHECK CONSTRAINT [FK_SituationsPerson_Person]
GO
ALTER TABLE [dbo].[Surgery]  WITH CHECK ADD  CONSTRAINT [FK_Surgery_Person] FOREIGN
KEY([PersonId])
REFERENCES [dbo].[Person] ([PersonID])
ON DELETE CASCADE
GO
ALTER TABLE [dbo].[Surgery] CHECK CONSTRAINT [FK_Surgery_Person]
GO
ALTER TABLE [dbo].[TelikhDiagn]  WITH CHECK ADD  CONSTRAINT [FK_TelikhDiagn_ReasonVisit]
FOREIGN KEY([ReasonVisitId])
REFERENCES [dbo].[ReasonVisit] ([ReasonVisitID])
ON DELETE CASCADE
GO
ALTER TABLE [dbo].[TelikhDiagn] CHECK CONSTRAINT [FK_TelikhDiagn_ReasonVisit]
GO
ALTER TABLE [dbo].[Thireo]  WITH CHECK ADD  CONSTRAINT [FK_Thireo_Person_Dad] FOREIGN
KEY([DadId])
REFERENCES [dbo].[Person] ([PersonID])
GO
ALTER TABLE [dbo].[Thireo] CHECK CONSTRAINT [FK_Thireo_Person_Dad]
GO
ALTER TABLE [dbo].[Thireo]  WITH CHECK ADD  CONSTRAINT [FK_Thireo_Person_Kid] FOREIGN
KEY([KidId])
REFERENCES [dbo].[Person] ([PersonID])
GO
ALTER TABLE [dbo].[Thireo] CHECK CONSTRAINT [FK_Thireo_Person_Kid]
GO
ALTER TABLE [dbo].[Thireo]  WITH CHECK ADD  CONSTRAINT [FK_Thireo_Person_Mum] FOREIGN
KEY([MumId])
REFERENCES [dbo].[Person] ([PersonID])
GO
ALTER TABLE [dbo].[Thireo] CHECK CONSTRAINT [FK_Thireo_Person_Mum]
GO
```

```
ALTER TABLE [dbo].[Viavhths]  WITH CHECK ADD  CONSTRAINT [FK_Viavhths_Person_Dad]
FOREIGN KEY([DadId])
REFERENCES [dbo].[Person] ([PersonID])
GO
ALTER TABLE [dbo].[Viavhths] CHECK CONSTRAINT [FK_Viavhths_Person_Dad]
GO
ALTER TABLE [dbo].[Viavhths]  WITH CHECK ADD  CONSTRAINT [FK_Viavhths_Person_Kid]
FOREIGN KEY([KidId])
REFERENCES [dbo].[Person] ([PersonID])
GO
ALTER TABLE [dbo].[Viavhths] CHECK CONSTRAINT [FK_Viavhths_Person_Kid]
GO
ALTER TABLE [dbo].[Viavhths]  WITH CHECK ADD  CONSTRAINT [FK_Viavhths_Person_Mum]
FOREIGN KEY([MumId])
REFERENCES [dbo].[Person] ([PersonID])
GO
ALTER TABLE [dbo].[Viavhths] CHECK CONSTRAINT [FK_Viavhths_Person_Mum]
GO
/****** Object:  StoredProcedure [dbo].[getLastName]    Script Date: 18-Nov-16 4:07:31 PM ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO


CREATE PROCEDURE [dbo].[getLastName] @City nvarchar(30)
AS
SELECT [PersonID]
    ,[First_Name]
    ,[Last_Name]
    ,[Sex]
    ,[Birth_Date]
    ,[Birth_City]
    ,[Town_Live]
    ,[Adreess_Live]
    ,[Phone]
    ,[Mobile]
    ,[Mail]
    ,[Death_Date]
    ,[WhyDied]
    ,[Country_Live]
    ,[userCreator]
 FROM [PatientHistory].[dbo].[Person]
WHERE [Last_Name] = @City

GO
/****** Object:  StoredProcedure [dbo].[INSERTDATA]    Script Date: 18-Nov-16 4:07:31 PM ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO

CREATE PROCEDURE [dbo].[INSERTDATA]

AS
BEGIN
```

```
	DECLARE @FatherID int

	DECLARE Find_Father_CUR CURSOR
	LOCAL STATIC READ_ONLY FORWARD_ONLY
	FOR
SELECT DISTINCT [PersonId]  FROM [PatientHistory].[dbo].[ReasonVisit]  WHERE
[ReasonDescr]='ΔΙΑΡΡΟΙΑ'
	OPEN Find_Father_CUR
	FETCH NEXT FROM Find_Father_CUR INTO @FatherID
	WHILE @@FETCH_STATUS = 0
	BEGIN

INSERT [dbo].[Surgery] ( [PersonId], [DescriptionSurgery]) VALUES ( @FatherID, N'ΚΑΤΑΓΜΑ ΙΣΧΥΟΥ
2001')

		FETCH NEXT FROM Find_Father_CUR INTO @FatherID
	END
	CLOSE Find_Father_CUR
	DEALLOCATE Find_Father_CUR
END
GO
/****** Object:  StoredProcedure [dbo].[parents]    Script Date: 18-Nov-16 4:07:31 PM ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO

CREATE PROCEDURE [dbo].[parents]

AS
BEGIN
	SET NOCOUNT ON;

	DELETE [PatientHistory].[dbo].[Family]

	--DECLARE @ChildID INT;
	--SELECT @ChildID = [Person1Id]  FROM [PatientHistory].[dbo].[Relationship]

DECLARE @FatherID int
DECLARE @ChildID int
DECLARE @MotherID int
DECLARE @ChildExists int

	--SXESEIS PAIDIOY PATERA
	DECLARE Find_Father_CUR CURSOR
	LOCAL STATIC READ_ONLY FORWARD_ONLY
	FOR
	SELECT [Person1Id],[Person2Id]  FROM [PatientHistory].[dbo].[Relationship] WHERE [RelatypeId]=2
	OPEN Find_Father_CUR
	FETCH NEXT FROM Find_Father_CUR INTO @ChildID,@FatherID
	WHILE @@FETCH_STATUS = 0
	BEGIN
		INSERT INTO [PatientHistory].[dbo].[Family] ([MotherId],[FatherId],[ChildId])
						VALUES	(null,@FatherID,@ChildID)
```

```
		FETCH NEXT FROM Find_Father_CUR INTO @ChildID,@FatherID
	END
	CLOSE Find_Father_CUR
	DEALLOCATE Find_Father_CUR

	--SXESEIS PAIDIOY MHTERAS
	DECLARE Find_Mother_CUR CURSOR
	LOCAL STATIC READ_ONLY FORWARD_ONLY
	FOR
	SELECT [Person1Id],[Person2Id]  FROM [PatientHistory].[dbo].[Relationship] WHERE [RelatypeId]=1
	OPEN Find_Mother_CUR
	FETCH NEXT FROM Find_Mother_CUR INTO @ChildID,@MotherID
	WHILE @@FETCH_STATUS = 0
	BEGIN
		SET @ChildExists = (SELECT COUNT(*) FROM [PatientHistory].[dbo].[Family] WHERE
[ChildId]=@ChildID)

		--AN YPARXEI IDI TO PAIDI ME TON PATERA TOU
		IF @ChildExists > 0
			UPDATE [PatientHistory].[dbo].[Family] SET [MotherId] = @MotherID WHERE
[ChildId]=@ChildID;
		ELSE
			INSERT INTO [PatientHistory].[dbo].[Family] ([MotherId],[FatherId],[ChildId])
								VALUES	(@MotherID,null,@ChildID) ;

		FETCH NEXT FROM Find_Mother_CUR INTO @ChildID,@MotherID
	END
	CLOSE Find_Mother_CUR
	DEALLOCATE Find_Mother_CUR
END

GO
/****** Object:  StoredProcedure [dbo].[parentsANAIMIA]	Script Date: 18-Nov-16 4:07:31 PM ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO

CREATE PROCEDURE [dbo].[parentsANAIMIA]

AS
BEGIN
	SET NOCOUNT ON;

	DELETE [PatientHistory].[dbo].[Anaimia]

	DECLARE @FatherID int
	DECLARE @ChildID int
	DECLARE @MotherID int
	DECLARE @ChildExists int
	DECLARE @AsthenisID int
	DECLARE @MamHas varchar(60)
	DECLARE @DadHas varchar(60)

BEGIN --CREATE FAMILY
	--SXESEIS PAIDIOY PATERA
```

```
	DECLARE Find_Father_CUR CURSOR
	LOCAL STATIC READ_ONLY FORWARD_ONLY
	FOR
	SELECT [Person1Id],[Person2Id]  FROM [PatientHistory].[dbo].[Relationship] WHERE [RelatypeId]=2
	OPEN Find_Father_CUR
	FETCH NEXT FROM Find_Father_CUR INTO @ChildID,@FatherID
	WHILE @@FETCH_STATUS = 0
	BEGIN
		INSERT INTO [PatientHistory].[dbo].[Anaimia] ([MumId],[DadId],[KidId])
						VALUES	(null,@FatherID,@ChildID)

		FETCH NEXT FROM Find_Father_CUR INTO @ChildID,@FatherID
	END
	CLOSE Find_Father_CUR
	DEALLOCATE Find_Father_CUR

	--SXESEIS PAIDIOY MHTERAS
	DECLARE Find_Mother_CUR CURSOR
	LOCAL STATIC READ_ONLY FORWARD_ONLY
	FOR
	SELECT [Person1Id],[Person2Id]  FROM [PatientHistory].[dbo].[Relationship] WHERE [RelatypeId]=1
	OPEN Find_Mother_CUR
	FETCH NEXT FROM Find_Mother_CUR INTO @ChildID,@MotherID
	WHILE @@FETCH_STATUS = 0
	BEGIN
		SET @ChildExists = (SELECT COUNT(*) FROM [PatientHistory].[dbo].[Anaimia] WHERE
[KidId]=@ChildID)

		--AN YPARXEI IDI TO PAIDI ME TON PATERA TOU
		IF @ChildExists > 0
			UPDATE [PatientHistory].[dbo].[Anaimia] SET [MumId] = @MotherID WHERE
[KidId]=@ChildID;
		ELSE
			INSERT INTO [PatientHistory].[dbo].[Anaimia] ([MumId],[DadId],[KidId])
						VALUES	(@MotherID,null,@ChildID) ;

		FETCH NEXT FROM Find_Mother_CUR INTO @ChildID,@MotherID
	END
	CLOSE Find_Mother_CUR
	DEALLOCATE Find_Mother_CUR
END--CREATE FAMILY

BEGIN --ASTHENEIS ME ANAIMIA STIGMA
	--ATOMA ME ANAIMIA
	DECLARE Find_ASTHENEIS_CUR CURSOR
	LOCAL STATIC READ_ONLY FORWARD_ONLY
	FOR
	SELECT DISTINCT [PersonId] FROM [PatientHistory].[dbo].[NoshmataPersonView]
								WHERE upper([SitDescr]) LIKE
'%ΜΕΣΟΓ%ΑΝΑΙΜ%ΣΤ%ΓΜΑ%'
	OPEN Find_ASTHENEIS_CUR
	FETCH NEXT FROM Find_ASTHENEIS_CUR INTO @AsthenisID
	WHILE @@FETCH_STATUS = 0
	BEGIN
		--ENHMERVSH TOY NOSHMATOS TOU STHN OIKOGENEIA
```

```
                UPDATE [PatientHistory].[dbo].[Anaimia] SET [MumHas]='ΣΤΙΓΜΑ' WHERE
[MumId]=@AsthenisID;
                UPDATE [PatientHistory].[dbo].[Anaimia] SET [DadHas]='ΣΤΙΓΜΑ' WHERE
[DadId]=@AsthenisID;
                UPDATE [PatientHistory].[dbo].[Anaimia] SET [KidHas]='ΣΤΙΓΜΑ' WHERE
[KidId]=@AsthenisID;

                FETCH NEXT FROM Find_ASTHENEIS_CUR INTO @AsthenisID
        END
        CLOSE Find_ASTHENEIS_CUR
        DEALLOCATE Find_ASTHENEIS_CUR
END --ASTHENEIS ME ANAIMIA ΣΤΙΓΜΑ

BEGIN --ASTHENEIS ME ANAIMIA ΝΟΣΟΥΝ
        --ΑΤΟΜΑ ΜΕ ΑΝΑΙΜΙΑ ΠΟΥ ΝΟΣΟΣΥΝ
        DECLARE Find_ASTHENEISnosun_CUR CURSOR
        LOCAL STATIC READ_ONLY FORWARD_ONLY
        FOR
        SELECT DISTINCT [PersonId] FROM [PatientHistory].[dbo].[NoshmataPersonView]
                                                        WHERE upper([SitDescr]) LIKE
'%ΜΕΣΟΓ%ΑΝΑΙΜ%ΝΟΣΕ%%'
        OPEN Find_ASTHENEISnosun_CUR
        FETCH NEXT FROM Find_ASTHENEISnosun_CUR INTO @AsthenisID
        WHILE @@FETCH_STATUS = 0
        BEGIN
                --ENHMERVSH TOY NOSHMATOS TOU STHN OIKOGENEIA
                UPDATE [PatientHistory].[dbo].[Anaimia] SET [MumHas]='ΝΟΣΕΙ' WHERE
[MumId]=@AsthenisID;
                UPDATE [PatientHistory].[dbo].[Anaimia] SET [DadHas]='ΝΟΣΕΙ' WHERE
[DadId]=@AsthenisID;
                UPDATE [PatientHistory].[dbo].[Anaimia] SET [KidHas]='ΝΟΣΕΙ' WHERE
[KidId]=@AsthenisID;

                FETCH NEXT FROM Find_ASTHENEISnosun_CUR INTO @AsthenisID
        END
        CLOSE Find_ASTHENEISnosun_CUR
        DEALLOCATE Find_ASTHENEISnosun_CUR
END --ASTHENEIS ME ANAIMIA ΝΟΣΟΥΝ

BEGIN --PITHANOTHTES PAIDIOU
        DECLARE Find_POSOSTO_CUR CURSOR
        LOCAL STATIC READ_ONLY FORWARD_ONLY
        FOR
SELECT [MumHas]
    ,[DadHas]
    ,[KidId]
 FROM [PatientHistory].[dbo].[Anaimia]

        OPEN Find_POSOSTO_CUR
        FETCH NEXT FROM Find_POSOSTO_CUR INTO @MamHas,@DadHas,@ChildID
        WHILE @@FETCH_STATUS = 0
        BEGIN
                --DIAMORFWSH POSOSTWN
                --Δύο υγιείς γονείς -> κανένα παιδί
                --Ένας γονέας με το στίγμα -> 50% να έχει στίγμα
                --Δύο γονείς με στίγμα -> 25% να νοσήσει και 50% να έχει στίγμα ως υγιής φορέας
```

```
                --Ένας με στίγμα και ένας να νοσεί -> 50% να νοσήσει με την ασθένεια της Μείζονος
                --Θαλασσαιμίας και 50% να έχει το στίγμα ως υγιής φορέας

                IF (((@MamHas='ΣΤΙΓΜΑ') AND (@DadHas IS NULL)) OR ((@MamHas IS NULL) AND
(@DadHas='ΣΤΙΓΜΑ')))
                UPDATE [PatientHistory].[dbo].[Anaimia] SET [Pososto CHILDREN]='~50% ΝΑ ΕΜΦΑΝΙΣΕΙ'
WHERE [KidId]=@ChildID;
                ELSE IF ((@MamHas='ΣΤΙΓΜΑ') AND (@DadHas='ΣΤΙΓΜΑ'))
                UPDATE [PatientHistory].[dbo].[Anaimia] SET [Pososto CHILDREN]='~25% ΝΑ ΝΟΣΗΣΕΙ
ΚΑΙ ~50% ΝΑ ΕΧΕΙ ΣΤΙΓΜΑ' WHERE [KidId]=@ChildID;
                ELSE IF (((@MamHas='ΣΤΙΓΜΑ') AND (@DadHas='ΝΟΣΕΙ')) OR ((@MamHas='ΝΟΣΕΙ')
AND (@DadHas='ΣΤΙΓΜΑ')))
                UPDATE [PatientHistory].[dbo].[Anaimia] SET [Pososto CHILDREN]='~50% ΝΑ ΝΟΣΗΣΕΙ
ΚΑΙ ~50% ΝΑ ΕΧΕΙ ΣΤΙΓΜΑ' WHERE [KidId]=@ChildID;

                FETCH NEXT FROM Find_POSOSTO_CUR INTO @MamHas,@DadHas,@ChildID
        END
        CLOSE Find_POSOSTO_CUR
        DEALLOCATE Find_POSOSTO_CUR
END --PITHANOTHTES PAIDIOU

END

GO
/****** Object:  StoredProcedure [dbo].[parentsDIAVITIS]    Script Date: 18-Nov-16 4:07:31 PM ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO

CREATE PROCEDURE [dbo].[parentsDIAVITIS]

AS
BEGIN
        SET NOCOUNT ON;

        DELETE [PatientHistory].[dbo].[Viavhths]

        DECLARE @FatherID int
        DECLARE @ChildID int
        DECLARE @MotherID int
        DECLARE @ChildExists int
        DECLARE @AsthenisID int
        DECLARE @MamHas varchar(60)
        DECLARE @DadHas varchar(60)

BEGIN --CREATE FAMILY
        --SXESEIS PAIDIOY PATERA
        DECLARE Find_Father_CUR CURSOR
        LOCAL STATIC READ_ONLY FORWARD_ONLY
        FOR
        SELECT [Person1Id],[Person2Id]  FROM [PatientHistory].[dbo].[Relationship] WHERE [RelatypeId]=2
        OPEN Find_Father_CUR
        FETCH NEXT FROM Find_Father_CUR INTO @ChildID,@FatherID
        WHILE @@FETCH_STATUS = 0
        BEGIN
```

```
		INSERT INTO [PatientHistory].[dbo].[Viavhths] ([MumId],[DadId],[KidId])
								VALUES	(null,@FatherID,@ChildID)

		FETCH NEXT FROM Find_Father_CUR INTO @ChildID,@FatherID
	END
	CLOSE Find_Father_CUR
	DEALLOCATE Find_Father_CUR

	--SXESEIS PAIDIOY MHTERAS
	DECLARE Find_Mother_CUR CURSOR
	LOCAL STATIC READ_ONLY FORWARD_ONLY
	FOR
	SELECT [Person1Id],[Person2Id]  FROM [PatientHistory].[dbo].[Relationship] WHERE [RelatypeId]=1
	OPEN Find_Mother_CUR
	FETCH NEXT FROM Find_Mother_CUR INTO @ChildID,@MotherID
	WHILE @@FETCH_STATUS = 0
	BEGIN
		SET @ChildExists = (SELECT COUNT(*) FROM [PatientHistory].[dbo].[Viavhths] WHERE
[KidId]=@ChildID)

		--AN YPARXEI IDI TO PAIDI ME TON PATERA TOU
		IF @ChildExists > 0
			UPDATE [PatientHistory].[dbo].[Viavhths] SET [MumId] = @MotherID WHERE
[KidId]=@ChildID;
		ELSE
			INSERT INTO [PatientHistory].[dbo].[Viavhths] ([MumId],[DadId],[KidId])
								VALUES	(@MotherID,null,@ChildID) ;

		FETCH NEXT FROM Find_Mother_CUR INTO @ChildID,@MotherID
	END
	CLOSE Find_Mother_CUR
	DEALLOCATE Find_Mother_CUR
END--CREATE FAMILY

BEGIN --ASTHENEIS ME VIAVHTH
	--ATOMA ME VIAVHTH
	DECLARE Find_ASTHENEIS_CUR CURSOR
	LOCAL STATIC READ_ONLY FORWARD_ONLY
	FOR
	SELECT DISTINCT [PersonId] FROM [PatientHistory].[dbo].[NoshmataPersonView]
								WHERE upper([SitDescr]) LIKE
'%ΔΙΑΒ%ΤΗΣ%'
	OPEN Find_ASTHENEIS_CUR
	FETCH NEXT FROM Find_ASTHENEIS_CUR INTO @AsthenisID
	WHILE @@FETCH_STATUS = 0
	BEGIN
		--ENHMERVSH TOY NOSHMATOS TOU STHN OIKOGENEIA
		UPDATE [PatientHistory].[dbo].[Viavhths] SET [MumHas]='Σακχαρώδης Διαβήτης' WHERE
[MumId]=@AsthenisID;
		UPDATE [PatientHistory].[dbo].[Viavhths] SET [DadHas]='Σακχαρώδης Διαβήτης' WHERE
[DadId]=@AsthenisID;
		UPDATE [PatientHistory].[dbo].[Viavhths] SET [KidHas]='Σακχαρώδης Διαβήτης' WHERE
[KidId]=@AsthenisID;

		FETCH NEXT FROM Find_ASTHENEIS_CUR INTO @AsthenisID
	END
```

```
        CLOSE Find_ASTHENEIS_CUR
        DEALLOCATE Find_ASTHENEIS_CUR
END --ASTHENEIS ME DIAVHTH

BEGIN --PITHANOTHTES PAIDIOU
        DECLARE Find_POSOSTO_CUR CURSOR
        LOCAL STATIC READ_ONLY FORWARD_ONLY
        FOR
SELECT [MumHas]
    ,[DadHas]
    ,[KidId]
 FROM [PatientHistory].[dbo].[Viavhths]

        OPEN Find_POSOSTO_CUR
        FETCH NEXT FROM Find_POSOSTO_CUR INTO @MamHas,@DadHas,@ChildID
        WHILE @@FETCH_STATUS = 0
        BEGIN
                --DIAMORFWSH POSOSTWN
                --Αν έχει διαβήτη ο ένας γονιός σας, έχετε 40% πιθανότητες να τον παρουσιάσετε κι εσείς.
                --Αν έχουν και οι δύο, οι πιθανότητες για εσάς πλησιάζουν το 80%.
                IF ((@MamHas='Σακχαρώδης Διαβήτης')and(@DadHas='Σακχαρώδης Διαβήτης'))
                UPDATE [PatientHistory].[dbo].[Viavhths] SET [PososTOCHILDREN]='~80% NA
ΕΜΦΑΝΙΣΕΙ' WHERE [KidId]=@ChildID;
                ELSE IF ((@MamHas IS NULL) AND (@DadHas IS NULL))
                UPDATE [PatientHistory].[dbo].[Viavhths] SET [PososTOCHILDREN]=NULL WHERE
[KidId]=@ChildID;
                ELSE
                UPDATE [PatientHistory].[dbo].[Viavhths] SET [PososTOCHILDREN]='~40% NA
ΕΜΦΑΝΙΣΕΙ' WHERE [KidId]=@ChildID;

                FETCH NEXT FROM Find_POSOSTO_CUR INTO @MamHas,@DadHas,@ChildID
        END
        CLOSE Find_POSOSTO_CUR
        DEALLOCATE Find_POSOSTO_CUR
END --PITHANOTHTES PAIDIOU

END

GO
/****** Object:  StoredProcedure [dbo].[parentsOSTEOPORWSH]    Script Date: 18-Nov-16 4:07:31 PM ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO

CREATE PROCEDURE [dbo].[parentsOSTEOPORWSH]

AS
BEGIN
        SET NOCOUNT ON;

        DELETE [PatientHistory].[dbo].[Osteoporwsh]

        DECLARE @FatherID int
        DECLARE @ChildID int
        DECLARE @MotherID int
```

```
	DECLARE @ChildExists int
	DECLARE @AsthenisID int
	DECLARE @MamHas varchar(60)
	DECLARE @DadHas varchar(60)

BEGIN --CREATE FAMILY
	--SXESEIS PAIDIOY PATERA
	DECLARE Find_Father_CUR CURSOR
	LOCAL STATIC READ_ONLY FORWARD_ONLY
	FOR
	SELECT [Person1Id],[Person2Id]  FROM [PatientHistory].[dbo].[Relationship] WHERE [RelatypeId]=2
	OPEN Find_Father_CUR
	FETCH NEXT FROM Find_Father_CUR INTO @ChildID,@FatherID
	WHILE @@FETCH_STATUS = 0
	BEGIN
		INSERT INTO [PatientHistory].[dbo].[Osteoporwsh] ([MumId],[DadId],[KidId])
						VALUES	(null,@FatherID,@ChildID)

		FETCH NEXT FROM Find_Father_CUR INTO @ChildID,@FatherID
	END
	CLOSE Find_Father_CUR
	DEALLOCATE Find_Father_CUR

	--SXESEIS PAIDIOY MHTERAS
	DECLARE Find_Mother_CUR CURSOR
	LOCAL STATIC READ_ONLY FORWARD_ONLY
	FOR
	SELECT [Person1Id],[Person2Id]  FROM [PatientHistory].[dbo].[Relationship] WHERE [RelatypeId]=1
	OPEN Find_Mother_CUR
	FETCH NEXT FROM Find_Mother_CUR INTO @ChildID,@MotherID
	WHILE @@FETCH_STATUS = 0
	BEGIN
		SET @ChildExists = (SELECT COUNT(*) FROM [PatientHistory].[dbo].[Osteoporwsh]
WHERE [KidId]=@ChildID)

		--AN YPARXEI IDI TO PAIDI ME TON PATERA TOU
		IF @ChildExists > 0
			UPDATE [PatientHistory].[dbo].[Osteoporwsh] SET [MumId] = @MotherID WHERE
[KidId]=@ChildID;
		ELSE
			INSERT INTO [PatientHistory].[dbo].[Osteoporwsh] ([MumId],[DadId],[KidId])
						VALUES	(@MotherID,null,@ChildID) ;

		FETCH NEXT FROM Find_Mother_CUR INTO @ChildID,@MotherID
	END
	CLOSE Find_Mother_CUR
	DEALLOCATE Find_Mother_CUR
END--CREATE FAMILY

BEGIN --ASTHENEIS ME VIAVHTH
	--ATOMA ME VIAVHTH
	DECLARE Find_ASTHENEIS_CUR CURSOR
	LOCAL STATIC READ_ONLY FORWARD_ONLY
	FOR
	SELECT DISTINCT [PersonId] FROM [PatientHistory].[dbo].[NoshmataPersonView]
```

```
                                                                WHERE upper([SitDescr]) LIKE
'%ΟΣΤΕΟΠ%ΡΩΣΗ%'
        OPEN Find_ASTHENEIS_CUR
        FETCH NEXT FROM Find_ASTHENEIS_CUR INTO @AsthenisID
        WHILE @@FETCH_STATUS = 0
        BEGIN
                --ENHMERVSH TOY NOSHMATOS TOU STHN OIKOGENEIA
                UPDATE [PatientHistory].[dbo].[Osteoporwsh] SET [MumHas]='Οστεοπόρωση' WHERE
[MumId]=@AsthenisID;
                UPDATE [PatientHistory].[dbo].[Osteoporwsh] SET [DadHas]='Οστεοπόρωση' WHERE
[DadId]=@AsthenisID;
                UPDATE [PatientHistory].[dbo].[Osteoporwsh] SET [KidHas]='Οστεοπόρωση' WHERE
[KidId]=@AsthenisID;

                FETCH NEXT FROM Find_ASTHENEIS_CUR INTO @AsthenisID
        END
        CLOSE Find_ASTHENEIS_CUR
        DEALLOCATE Find_ASTHENEIS_CUR
END --ASTHENEIS ME DIAVHTH

BEGIN --PITHANOTHTES PAIDIOU
        DECLARE Find_POSOSTO_CUR CURSOR
        LOCAL STATIC READ_ONLY FORWARD_ONLY
        FOR
SELECT [MumHas]
    ,[DadHas]
    ,[KidId]
 FROM [PatientHistory].[dbo].[Osteoporwsh]

        OPEN Find_POSOSTO_CUR
        FETCH NEXT FROM Find_POSOSTO_CUR INTO @MamHas,@DadHas,@ChildID
        WHILE @@FETCH_STATUS = 0
        BEGIN
                --DIAMORFWSH POSOSTWN
                --Αν έχει οστεοπόρωση η μητέρα ή ο πατέρας σας, τότε έχετε 70% πιθανότητες να παρουσιάσετε
κι εσείς.
                --(Η οστεοπόρωση της γιαγιάς ή του παππού, όμως, δεν φαίνεται στατιστικά να σας αφορά
ιδιαίτερα.)
                IF (@MamHas=@DadHas)
                UPDATE [PatientHistory].[dbo].[Osteoporwsh] SET [PososTOCHILDREN]='~70% ΝΑ
ΕΜΦΑΝΙΣΕΙ' WHERE [KidId]=@ChildID;
                ELSE IF ((@MamHas IS NULL) AND (@DadHas IS NULL))
                UPDATE [PatientHistory].[dbo].[Osteoporwsh] SET [PososTOCHILDREN]=NULL WHERE
[KidId]=@ChildID;
                ELSE
                UPDATE [PatientHistory].[dbo].[Osteoporwsh] SET [PososTOCHILDREN]='~70% ΝΑ
ΕΜΦΑΝΙΣΕΙ' WHERE [KidId]=@ChildID;

                FETCH NEXT FROM Find_POSOSTO_CUR INTO @MamHas,@DadHas,@ChildID
        END
        CLOSE Find_POSOSTO_CUR
        DEALLOCATE Find_POSOSTO_CUR
END --PITHANOTHTES PAIDIOU

END
```

```
GO
/****** Object:  StoredProcedure [dbo].[parentsTHIREO]    Script Date: 18-Nov-16 4:07:31 PM ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO

CREATE PROCEDURE [dbo].[parentsTHIREO]

AS
BEGIN
	SET NOCOUNT ON;

	DELETE [PatientHistory].[dbo].[Thireo]

	DECLARE @FatherID int
	DECLARE @ChildID int
	DECLARE @MotherID int
	DECLARE @ChildExists int
	DECLARE @AsthenisID int
	DECLARE @MamHas varchar(60)
	DECLARE @DadHas varchar(60)

BEGIN --CREATE FAMILY
	--SXESEIS PAIDIOY PATERA
	DECLARE Find_Father_CUR CURSOR
	LOCAL STATIC READ_ONLY FORWARD_ONLY
	FOR
	SELECT [Person1Id],[Person2Id]  FROM [PatientHistory].[dbo].[Relationship] WHERE [RelatypeId]=2
	OPEN Find_Father_CUR
	FETCH NEXT FROM Find_Father_CUR INTO @ChildID,@FatherID
	WHILE @@FETCH_STATUS = 0
	BEGIN
		INSERT INTO [PatientHistory].[dbo].[Thireo] ([MumId],[DadId],[KidId])
						VALUES	(null,@FatherID,@ChildID)

		FETCH NEXT FROM Find_Father_CUR INTO @ChildID,@FatherID
	END
	CLOSE Find_Father_CUR
	DEALLOCATE Find_Father_CUR

	--SXESEIS PAIDIOY MHTERAS
	DECLARE Find_Mother_CUR CURSOR
	LOCAL STATIC READ_ONLY FORWARD_ONLY
	FOR
	SELECT [Person1Id],[Person2Id]  FROM [PatientHistory].[dbo].[Relationship] WHERE [RelatypeId]=1
	OPEN Find_Mother_CUR
	FETCH NEXT FROM Find_Mother_CUR INTO @ChildID,@MotherID
	WHILE @@FETCH_STATUS = 0
	BEGIN
		SET @ChildExists = (SELECT COUNT(*) FROM [PatientHistory].[dbo].[Thireo] WHERE
[KidId]=@ChildID)

		--AN YPARXEI IDI TO PAIDI ME TON PATERA TOU
		IF @ChildExists > 0
```

```
                        UPDATE [PatientHistory].[dbo].[Thireo] SET [MumId] = @MotherID WHERE
[KidId]=@ChildID;
                ELSE
                        INSERT INTO [PatientHistory].[dbo].[Thireo] ([MumId],[DadId],[KidId])
                                                VALUES    (@MotherID,null,@ChildID) ;

                FETCH NEXT FROM Find_Mother_CUR INTO @ChildID,@MotherID
        END
        CLOSE Find_Mother_CUR
        DEALLOCATE Find_Mother_CUR
END--CREATE FAMILY

BEGIN --ASTHENEIS ME διαταραχή του θυρεοειδούς
        --ATOMA ME διαταραχή του θυρεοειδούς
        DECLARE Find_ASTHENEIS_CUR CURSOR
        LOCAL STATIC READ_ONLY FORWARD_ONLY
        FOR
        SELECT DISTINCT [PersonId] FROM [PatientHistory].[dbo].[NoshmataPersonView]
                                                        WHERE upper([SitDescr]) LIKE
'%ΘΥΡΕΟΕΙΔ%ΤΙΔΑ%HASHIMOTO%'
        OPEN Find_ASTHENEIS_CUR
        FETCH NEXT FROM Find_ASTHENEIS_CUR INTO @AsthenisID
        WHILE @@FETCH_STATUS = 0
        BEGIN
                --ENHMERVSH TOY NOSHMATOS TOU STHN OIKOGENEIA
                UPDATE [PatientHistory].[dbo].[Thireo] SET [MumHas]='HASHIMOTO' WHERE
[MumId]=@AsthenisID;
                UPDATE [PatientHistory].[dbo].[Thireo] SET [DadHas]='HASHIMOTO' WHERE
[DadId]=@AsthenisID;
                UPDATE [PatientHistory].[dbo].[Thireo] SET [KidHas]='HASHIMOTO' WHERE
[KidId]=@AsthenisID;

                FETCH NEXT FROM Find_ASTHENEIS_CUR INTO @AsthenisID
        END
        CLOSE Find_ASTHENEIS_CUR
        DEALLOCATE Find_ASTHENEIS_CUR
END --ASTHENEIS ME διαταραχή του θυρεοειδούς

BEGIN --PITHANOTHTES PAIDIOU
        DECLARE Find_POSOSTO_CUR CURSOR
        LOCAL STATIC READ_ONLY FORWARD_ONLY
        FOR
SELECT [MumHas]
    ,[DadHas]
    ,[KidId]
 FROM [PatientHistory].[dbo].[Thireo]

        OPEN Find_POSOSTO_CUR
        FETCH NEXT FROM Find_POSOSTO_CUR INTO @MamHas,@DadHas,@ChildID
        WHILE @@FETCH_STATUS = 0
        BEGIN
                --DIAMORFWSH POSOSTWN
                --Πρόσφατες μελέτες δείχνουν ότι αν κάποιος πάσχει από κάποια διαταραχή του θυρεοειδούς,
                --κάθε ένα από τα παιδιά του έχει 50% πιθανότητες να κληρονομήσει το γονίδιο
                --και να παρουσιάσει κάποια στιγμή στη ζωή του πρόβλημα θυρεοειδούς.
```

```
                IF ((@MamHas='HASHIMOTO')  OR  (@DadHas='HASHIMOTO'))
                UPDATE [PatientHistory].[dbo].[Thireo] SET [Pososto CHILDREN]='~25% ΝΑ ΝΟΣΗΣΕΙ ΚΑΙ
~50% ΝΑ ΕΜΦΑΝΙΣΕΙ ΔΙΑΤΑΡΑΧΗ ΤΟΥ ΘΥΡΕΟΕΙΔΟΥΣ' WHERE [KidId]=@ChildID;
                ELSE
                UPDATE [PatientHistory].[dbo].[Thireo] SET [Pososto CHILDREN]=NULL WHERE
[KidId]=@ChildID;

                FETCH NEXT FROM Find_POSOSTO_CUR INTO @MamHas,@DadHas,@ChildID
        END
        CLOSE Find_POSOSTO_CUR
        DEALLOCATE Find_POSOSTO_CUR
END --PITHANOTHTES PAIDIOU

END

GO
EXEC sys.sp_addextendedproperty @name=N'MS_DiagramPane1', @value=N'[0E232FF0-B466-11cf-A24F-
00AA00A3EFFF, 1.00]
Begin DesignProperties =
  Begin PaneConfigurations =
    Begin PaneConfiguration = 0
      NumPanes = 4
      Configuration = "(H (1[40] 4[20] 2[20] 3) )"
    End
    Begin PaneConfiguration = 1
      NumPanes = 3
      Configuration = "(H (1 [50] 4 [25] 3))"
    End
    Begin PaneConfiguration = 2
      NumPanes = 3
      Configuration = "(H (1 [50] 2 [25] 3))"
    End
    Begin PaneConfiguration = 3
      NumPanes = 3
      Configuration = "(H (4 [30] 2 [40] 3))"
    End
    Begin PaneConfiguration = 4
      NumPanes = 2
      Configuration = "(H (1 [56] 3))"
    End
    Begin PaneConfiguration = 5
      NumPanes = 2
      Configuration = "(H (2 [66] 3))"
    End
    Begin PaneConfiguration = 6
      NumPanes = 2
      Configuration = "(H (4 [50] 3))"
    End
    Begin PaneConfiguration = 7
      NumPanes = 1
      Configuration = "(V (3))"
    End
    Begin PaneConfiguration = 8
      NumPanes = 3
      Configuration = "(H (1[56] 4[18] 2) )"
    End
```

```
    Begin PaneConfiguration = 9
      NumPanes = 2
      Configuration = "(H (1 [75] 4))"
    End
    Begin PaneConfiguration = 10
      NumPanes = 2
      Configuration = "(H (1[66] 2) )"
    End
    Begin PaneConfiguration = 11
      NumPanes = 2
      Configuration = "(H (4 [60] 2))"
    End
    Begin PaneConfiguration = 12
      NumPanes = 1
      Configuration = "(H (1) )"
    End
    Begin PaneConfiguration = 13
      NumPanes = 1
      Configuration = "(V (4))"
    End
    Begin PaneConfiguration = 14
      NumPanes = 1
      Configuration = "(V (2))"
    End
    ActivePaneConfig = 0
  End
  Begin DiagramPane =
    Begin Origin =
      Top = 0
      Left = 0
    End
    Begin Tables =
      Begin Table = "PastSituation"
        Begin Extent =
          Top = 7
          Left = 48
          Bottom = 126
          Right = 244
        End
        DisplayFlags = 280
        TopColumn = 0
      End
      Begin Table = "SituationsPerson"
        Begin Extent =
          Top = 7
          Left = 292
          Bottom = 170
          Right = 511
        End
        DisplayFlags = 280
        TopColumn = 0
      End
      Begin Table = "Person"
        Begin Extent =
          Top = 7
          Left = 559
```

```
            Bottom = 170
            Right = 753
          End
          DisplayFlags = 280
          TopColumn = 1
        End
      End
    End
    Begin SQLPane =
    End
    Begin DataPane =
      Begin ParameterDefaults = ""
      End
      Begin ColumnWidths = 9
        Width = 284
        Width = 1200
        Width = 1200
        Width = 1200
        Width = 1200
        Width = 1200
        Width = 1200
        Width = 1200
        Width = 1200
      End
    End
    Begin CriteriaPane =
      Begin ColumnWidths = 11
        Column = 1440
        Alias = 900
        Table = 1170
        Output = 720
        Append = 1400
        NewValue = 1170
        SortType = 1350
        SortOrder = 1410
        GroupBy = 1350
        Filter = 1350
        Or = 1350
        Or = 1350
        Or = 1350
      End
    End
  End
' , @level0type=N'SCHEMA',@level0name=N'dbo',
@level1type=N'VIEW',@level1name=N'AgePersonNoshmaView'
GO
EXEC sys.sp_addextendedproperty @name=N'MS_DiagramPaneCount', @value=1 ,
@level0type=N'SCHEMA',@level0name=N'dbo',
@level1type=N'VIEW',@level1name=N'AgePersonNoshmaView'
GO
EXEC sys.sp_addextendedproperty @name=N'MS_DiagramPane1', @value=N'[0E232FF0-B466-11cf-A24F-
00AA00A3EFFF, 1.00]
Begin DesignProperties =
  Begin PaneConfigurations =
    Begin PaneConfiguration = 0
      NumPanes = 4
```

```
  Configuration = "(H (1[40] 4[20] 2[20] 3) )"
End
Begin PaneConfiguration = 1
  NumPanes = 3
  Configuration = "(H (1 [50] 4 [25] 3))"
End
Begin PaneConfiguration = 2
  NumPanes = 3
  Configuration = "(H (1 [50] 2 [25] 3))"
End
Begin PaneConfiguration = 3
  NumPanes = 3
  Configuration = "(H (4 [30] 2 [40] 3))"
End
Begin PaneConfiguration = 4
  NumPanes = 2
  Configuration = "(H (1 [56] 3))"
End
Begin PaneConfiguration = 5
  NumPanes = 2
  Configuration = "(H (2 [66] 3))"
End
Begin PaneConfiguration = 6
  NumPanes = 2
  Configuration = "(H (4 [50] 3))"
End
Begin PaneConfiguration = 7
  NumPanes = 1
  Configuration = "(V (3))"
End
Begin PaneConfiguration = 8
  NumPanes = 3
  Configuration = "(H (1[56] 4[18] 2) )"
End
Begin PaneConfiguration = 9
  NumPanes = 2
  Configuration = "(H (1 [75] 4))"
End
Begin PaneConfiguration = 10
  NumPanes = 2
  Configuration = "(H (1[66] 2) )"
End
Begin PaneConfiguration = 11
  NumPanes = 2
  Configuration = "(H (4 [60] 2))"
End
Begin PaneConfiguration = 12
  NumPanes = 1
  Configuration = "(H (1) )"
End
Begin PaneConfiguration = 13
  NumPanes = 1
  Configuration = "(V (4))"
End
Begin PaneConfiguration = 14
  NumPanes = 1
```

```
      Configuration = "(V (2))"
    End
    ActivePaneConfig = 0
  End
  Begin DiagramPane =
    Begin Origin =
      Top = 0
      Left = 0
    End
    Begin Tables =
      Begin Table = "PastSituation"
        Begin Extent =
          Top = 7
          Left = 48
          Bottom = 126
          Right = 244
        End
        DisplayFlags = 280
        TopColumn = 0
      End
      Begin Table = "SituationsPerson"
        Begin Extent =
          Top = 7
          Left = 292
          Bottom = 170
          Right = 511
        End
        DisplayFlags = 280
        TopColumn = 0
      End
    End
  End
  Begin SQLPane =
  End
  Begin DataPane =
    Begin ParameterDefaults = ""
    End
    Begin ColumnWidths = 9
      Width = 284
      Width = 1200
      Width = 1200
      Width = 1200
      Width = 1200
      Width = 1200
      Width = 1200
      Width = 1200
      Width = 1200
    End
  End
  Begin CriteriaPane =
    Begin ColumnWidths = 11
      Column = 1440
      Alias = 900
      Table = 1170
      Output = 720
      Append = 1400
```

```
        NewValue = 1170
        SortType = 1350
        SortOrder = 1410
        GroupBy = 1350
        Filter = 1350
        Or = 1350
        Or = 1350
        Or = 1350
      End
    End
End
' , @level0type=N'SCHEMA',@level0name=N'dbo',
@level1type=N'VIEW',@level1name=N'NoshmataPersonView'
GO
EXEC sys.sp_addextendedproperty @name=N'MS_DiagramPaneCount', @value=1 ,
@level0type=N'SCHEMA',@level0name=N'dbo', @level1type=N'VIEW',@level1name=N'NoshmataPersonView'
GO
USE [master]
GO
ALTER DATABASE [PatientHistory] SET  READ_WRITE
GO
```

13.2 Παράρτημα Β'

Παρακάτω παρουσιάζεται ο κώδικας από το web.config αρχείο του project.

```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
  <configSections>
    <section name="entityFramework"
type="System.Data.Entity.Internal.ConfigFile.EntityFrameworkSection, EntityFramework, Version=6.0.0.0,
Culture=neutral, PublicKeyToken=b77a5c561934e089" requirePermission="false" />
  </configSections>
  <connectionStrings>
    <add name="PatientHistoryEntities"
connectionString="metadata=res://*/PatientHistoryEntities.csdl|res://*/PatientHistoryEntities.ssdl|res://*/Pati
entHistoryEntities.msl;provider=System.Data.SqlClient;provider connection string=&quot;data
source=BEIMAN-SATELITE;initial catalog=PatientHistory;integrated
security=True;MultipleActiveResultSets=True;App=EntityFramework&quot;"
providerName="System.Data.EntityClient" />
    <add name="Default" connectionString="Data Source=BEIMAN-SATELITE;Initial
Catalog=PatientHistory;Integrated Security=True" providerName="System.Data.SqlClient" />
  </connectionStrings>
  <appSettings>
    <add key="webpages:Version" value="3.0.0.0" />
    <add key="webpages:Enabled" value="false" />
    <add key="ClientValidationEnabled" value="true" />
    <add key="UnobtrusiveJavaScriptEnabled" value="true" />
  </appSettings>
  <system.web>
    <authentication mode="None" />
    <compilation debug="true" targetFramework="4.5" />
    <httpRuntime targetFramework="4.5" />
    <!--<customErrors mode="On"/>-->
```

```
  </system.web>
  <system.webServer>
    <modules>
      <remove name="FormsAuthentication" />
    </modules>
  </system.webServer>
  <runtime>
    <assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1">
      <dependentAssembly>
        <assemblyIdentity name="Microsoft.Owin.Security" publicKeyToken="31bf3856ad364e35" />
        <bindingRedirect oldVersion="0.0.0.0-3.0.1.0" newVersion="3.0.1.0" />
      </dependentAssembly>
      <dependentAssembly>
        <assemblyIdentity name="Microsoft.Owin.Security.OAuth" publicKeyToken="31bf3856ad364e35" />
        <bindingRedirect oldVersion="0.0.0.0-3.0.1.0" newVersion="3.0.1.0" />
      </dependentAssembly>
      <dependentAssembly>
        <assemblyIdentity name="Microsoft.Owin.Security.Cookies" publicKeyToken="31bf3856ad364e35" />
        <bindingRedirect oldVersion="0.0.0.0-3.0.1.0" newVersion="3.0.1.0" />
      </dependentAssembly>
      <dependentAssembly>
        <assemblyIdentity name="Microsoft.Owin" publicKeyToken="31bf3856ad364e35" />
        <bindingRedirect oldVersion="0.0.0.0-3.0.1.0" newVersion="3.0.1.0" />
      </dependentAssembly>
      <dependentAssembly>
        <assemblyIdentity name="Newtonsoft.Json" culture="neutral" publicKeyToken="30ad4fe6b2a6aeed"
/>
        <bindingRedirect oldVersion="0.0.0.0-6.0.0.0" newVersion="6.0.0.0" />
      </dependentAssembly>
      <dependentAssembly>
        <assemblyIdentity name="System.Web.Optimization" publicKeyToken="31bf3856ad364e35" />
        <bindingRedirect oldVersion="1.0.0.0-1.1.0.0" newVersion="1.1.0.0" />
      </dependentAssembly>
      <dependentAssembly>
        <assemblyIdentity name="WebGrease" publicKeyToken="31bf3856ad364e35" />
        <bindingRedirect oldVersion="0.0.0.0-1.5.2.14234" newVersion="1.5.2.14234" />
      </dependentAssembly>
      <dependentAssembly>
        <assemblyIdentity name="System.Web.Helpers" publicKeyToken="31bf3856ad364e35" />
        <bindingRedirect oldVersion="1.0.0.0-3.0.0.0" newVersion="3.0.0.0" />
      </dependentAssembly>
      <dependentAssembly>
        <assemblyIdentity name="System.Web.Mvc" publicKeyToken="31bf3856ad364e35" />
        <bindingRedirect oldVersion="0.0.0.0-5.2.3.0" newVersion="5.2.3.0" />
      </dependentAssembly>
      <dependentAssembly>
        <assemblyIdentity name="System.Web.WebPages" publicKeyToken="31bf3856ad364e35" />
        <bindingRedirect oldVersion="0.0.0.0-3.0.0.0" newVersion="3.0.0.0" />
      </dependentAssembly>
    </assemblyBinding>
  </runtime>
  <entityFramework>
    <defaultConnectionFactory type="System.Data.Entity.Infrastructure.LocalDbConnectionFactory,
EntityFramework">
      <parameters>
        <parameter value="mssqllocaldb" />
```

```
    </parameters>
  </defaultConnectionFactory>
  <providers>
    <provider invariantName="System.Data.SqlClient"
type="System.Data.Entity.SqlServer.SqlProviderServices, EntityFramework.SqlServer" />
  </providers>
 </entityFramework>
    </configuration>
```

13.3 Παράρτημα Γ'

Παρακάτω παρουσιάζεται ο κώδικας από το PatientHistoryEntities.Context.cs αρχείο του project στο οποίο περιέχεται ο κώδικας για τον χειρισμό των οντοτήτων της βάσης δεδομένων μας από το Entity Framework..

```
namespace PatientHistory
{
  using System;
  using System.Data.Entity;
  using System.Data.Entity.Infrastructure;
  using System.Data.Entity.Core.Objects;
  using System.Linq;

  public partial class PatientHistoryEntities : DbContext
  {
    public PatientHistoryEntities()
      : base("name=PatientHistoryEntities")
    {
    }

    protected override void OnModelCreating(DbModelBuilder modelBuilder)
    {
      throw new UnintentionalCodeFirstException();
    }

    public virtual DbSet<Allergy> Allergies { get; set; }
    public virtual DbSet<DiaforikhDiagn> DiaforikhDiagns { get; set; }
    public virtual DbSet<DiseaseAdult> DiseaseAdults { get; set; }
    public virtual DbSet<DiseaseChild> DiseaseChilds { get; set; }
    public virtual DbSet<FamilyHierarchy> FamilyHierarchies { get; set; }
    public virtual DbSet<Habbit> Habbits { get; set; }
    public virtual DbSet<MedicalProblem> MedicalProblems { get; set; }
    public virtual DbSet<MedicalProblemPersonVisit> MedicalProblemPersonVisits { get; set; }
    public virtual DbSet<ParousaNoso> ParousaNosoes { get; set; }
    public virtual DbSet<PastSituation> PastSituations { get; set; }
    public virtual DbSet<PatientsSymptom> PatientsSymptoms { get; set; }
    public virtual DbSet<Person> People { get; set; }
    public virtual DbSet<Pharmaka> Pharmakas { get; set; }
    public virtual DbSet<ReasonVisit> ReasonVisits { get; set; }
    public virtual DbSet<Relationship> Relationships { get; set; }
    public virtual DbSet<Relatype> Relatypes { get; set; }
    public virtual DbSet<SituationsPerson> SituationsPersons { get; set; }
    public virtual DbSet<Surgery> Surgeries { get; set; }
    public virtual DbSet<Symptomata> Symptomatas { get; set; }
    public virtual DbSet<TelikhDiagn> TelikhDiagns { get; set; }
    public virtual DbSet<Anaimia> Anaimias { get; set; }
    public virtual DbSet<Family> Families { get; set; }
```

```
        public virtual DbSet<Osteoporwsh> Osteoporwshes { get; set; }
        public virtual DbSet<Viavhth> Viavhths { get; set; }
        public virtual DbSet<AgePersonNoshmaView> AgePersonNoshmaViews { get; set; }
        public virtual DbSet<NoshmataPersonView> NoshmataPersonViews { get; set; }
        public virtual DbSet<Thireo> Thireos { get; set; }

        public virtual ObjectResult<getLastName_Result> getLastName(string city)
        {
            var cityParameter = city != null ?
                new ObjectParameter("City", city) :
                new ObjectParameter("City", typeof(string));

            return
((IObjectContextAdapter)this).ObjectContext.ExecuteFunction<getLastName_Result>("getLastName",
cityParameter);
        }

        public virtual int parents()
        {
            return ((IObjectContextAdapter)this).ObjectContext.ExecuteFunction("parents");
        }

        public virtual int parentsANAIMIA()
        {
            return ((IObjectContextAdapter)this).ObjectContext.ExecuteFunction("parentsANAIMIA");
        }

        public virtual int parentsDIAVITIS()
        {
            return ((IObjectContextAdapter)this).ObjectContext.ExecuteFunction("parentsDIAVITIS");
        }

        public virtual int parentsOSTEOPORWSH()
        {
            return ((IObjectContextAdapter)this).ObjectContext.ExecuteFunction("parentsOSTEOPORWSH");
        }

        public virtual int parentsTHIREO()
        {
            return ((IObjectContextAdapter)this).ObjectContext.ExecuteFunction("parentsTHIREO");
        }
    }
}
```

14 ΒΙΒΛΙΟΓΡΑΦΙΑ

- Getting Started with Entity Framework 6 Code First using MVC 5 Tom Dykstra, Rick Anderson, E-book publication date: April, 2014
- ASP.NET MVC 6 Documentation Release Microsoft March 02, 2016
- Pro Entity Framework 4.0 The future of data access in .NET programming Copyright © 2010 by Scott Klein , Apress
- https://developers.google.com/chart/interactive/docs/gallery/orgchart
- https://en.wikipedia.org/wiki/ASP.NET_MVC
- https://docs.microsoft.com/en-us/aspnet/core/tutorials/first-mvc-app/
- https://www.asp.net/mvc/mvc5
- https://www.asp.net/entity-framework
- http://www.entityframeworktutorial.net/what-is-entityframework.aspx
- https://msdn.microsoft.com/en-us/library/aa937723(v=vs.113).aspx
- https://en.wikipedia.org/wiki/Entity_Framework
- https://msdn.microsoft.com/en-us/library/bb399367(v=vs.110).aspx
- http://getbootstrap.com/
- http://www.w3schools.com/bootstrap/
- https://www.asp.net/identity
- http://www.mikesdotnetting.com/article/259/asp-net-mvc-5-with-ef-6-working-with-files
- http://www.aurigma.com/upload-suite/developers/aspnet-mvc/how-to-upload-files-in-aspnet-mvc
- https://www.nuget.org/packages/PagedList.Mvc/
- http://www.dotnetjalps.com/2015/10/create-pdf-asp-net-mvc-rotativa.html
- https://www.codeproject.com/tips/818197/generate-pdf-in-asp-net-mvc-using-rotativa
- https://www.asp.net/web-pages/overview/data/7-displaying-data-in-a-chart
- https://en.wikipedia.org/wiki
- http://www.hl7.org.gr/
- https://sourceforge.net/projects/medilig/
- http://healthit.gr/
- http://profclinic.com/
- http://www.doctors-net.gr/
- http://www.apollo.gr/dev/index.asp
- http://tms.com.gr/
- http://www.vidavo.eu/index.php/el/
- http://www.iatropedia.gr/encyclopedia/sakcharodis-diavitis
- http://hashimoto.gr/
- http://www.iatronet.gr/ygeia/endokrinologia
- http://www.vita.gr/
- http://ygeia.tanea.gr/
- http://health.in.gr/news/scienceprogress
- http://www.onmed.gr/ygeia

- http://thalassemia.com/genetics-inheritance.aspx
- http://www.healthline.com/health/thalassemia
- http://www.diabetes.org/
- http://www.endocrineweb.com/conditions/type-2-diabetes/type-2-diabetes-causes
- http://www.spine-health.com/conditions/osteoporosis/osteoporosis-prevention
- https://ghr.nlm.nih.gov/condition/hashimoto-thyroiditis
- http://hypothyroidmom.com/hashimotos-thyroiditis-its-a-genetics-thing
- Δ. Καραμήτσος, Μ. Σιών, Γ. Γιαννόγλου «Κλινική Εξέταση και Διάγνωση» ΑΠΘ, University Studio Press 2007
- Διπλωματική Εργασία «Πληροφοριακά συστήματα στον τομέα της υγείας» Πανεπιστήμιο Πειραιώς 2014, Κυρίμης Γ. Παναγιώτης
- Διπλωματική Εργασία «Ανάπτυξη εφαρμογής προσωπικού ηλεκτρονικού ιατρικού φακέλου ασθενούς για ατομική του χρήση» Πανεπιστήμιο Πατρών 2009, Παναγιώτα Σπυροπούλου
- Διπλωματική Εργασία «Σχεδιασμός και υλοποίηση ολοκληρωμένου ηλεκτρονικού φακέλου υγείας ασθενούς για κατανεμημένα ιατρικά πληροφοριακά συστήματα νοσοκομείων βασισμένου στις τεχνολογίες web» Εθνικό Μετσόβιο Πολυτεχνείο 2013, Αλέξανδρος Χριστοδουλάκης –Εμμανουήλ Χριστοδουλάκης
- Διπλωματική Εργασία «Δημιουργία συστήματος ηλεκτρονικών ιατρικών φακέλων με χρήση έξυπνων καρτών» Πανεπιστήμιο Αιγαίου 2009, Γεώργιος Χατζηπαναγιώτου
- Διπλωματική Εργασία «Κατανεμημένο Ιατρικό Πληροφοριακό Σύστημα» ΤΕΙ Κρήτης 2010, Δήμητρα Λούσα
- «Ο Ηλεκτρονικός ιατρικός φάκελος στα ελληνικά δημόσια νοσοκομεία», Ευστρατία Χ. Μούρτου1 Επιθεώρηση Υγείας, Τόμος 17, Τεύχος 101, Ιούλιος – Αύγουστος 2006
- «Ηλεκτρονικός Φάκελος Υγείας και Ηλεκτρονική Συνταγογράφηση», Κουμπούρος, Ιωάννης, 2015, (https://repository.kallipos.gr/bitstream/11419/284/1/02_chapter_07.pdf)