

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ - ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΠΜΣ «Πληροφορική»



Εργασία Μαθήματος
«Τεχνολογία Λογισμικού»

Θέμα: Εφαρμογή καταχώρησης ραντεβού ασθενών-ιατρών - AnyDoctorStyle	
Επιβλέπουσα Καθηγήτρια	Κα. Μαρία Βίρβου
Όνομα φοιτητών (Αρ. Μητρώου)	Φώτιος Τσιούμας (ΜΠΠΛ21079)
	Διονύσης Κατσιγιάννης (ΜΠΠΛ21028)
	Ηρακλής Δραγούνης (ΜΠΠΛ21015)
Εξάμηνο - Ακαδημαϊκό Έτος	Χειμερινό Εξάμηνο (3ο) – 2022/23

Περιεχόμενα

1.Εισαγωγή	3
1.1.Στόχοι της εργασίας	3
1.2.Ορισμός του προβλήματος προς επίλυση	3
2.Σύντομη παρουσίαση της RUP	5
3.Φάση: Έναρξη (Inception)	7
3.1.Σύλληψη απαιτήσεων	7
3.2.Ανάλυση – Σχεδιασμός	9
3.2.1.Διάγραμμα Περιπτώσεων Χρήσης (1 ^η έκδοση)	9
3.2.2.Διάγραμμα Τάξεων (1 ^η έκδοση).....	10
4.Φάση: Εκπόνηση Μελέτης (Elaboration)	11
4.1.Ανάλυση – Σχεδιασμός	11
4.1.1.Διάγραμμα Περιπτώσεων Χρήσης (2 ^η έκδοση)	11
4.1.2.Διάγραμμα Τάξεων (2 ^η έκδοση).....	12
4.1.3.Διάγραμμα Αντικειμένων (1 ^η έκδοση)	13
4.1.4.Διάγραμμα Συνεργασίας (1 ^η έκδοση).....	13
4.1.5.Διάγραμμα Σειράς (1 ^η έκδοση)	14
4.1.6.Διάγραμμα Δραστηριοτήτων (1 ^η έκδοση).....	15
4.1.7.Διάγραμμα Καταστάσεων (1 ^η έκδοση)	16
4.1.8.Διάγραμμα Εξαρτημάτων (1 ^η έκδοση).....	17
4.1.9.Διάγραμμα Διανομής (1 ^η έκδοση).....	18
5.Φάση: Κατασκευή (Construction)	19
5.1.Ανάλυση – Σχεδιασμός	19
5.1.1.Διάγραμμα Περιπτώσεων Χρήσης (3 ^η έκδοση)	19
5.1.2.Διάγραμμα Τάξεων (3 ^η έκδοση).....	20
5.1.3.Διάγραμμα Αντικειμένων (2 ^η έκδοση)	21
5.1.4.Διάγραμμα Συνεργασίας (2 ^η έκδοση).....	21
5.1.5.Διάγραμμα Σειράς (2 ^η έκδοση)	22
5.1.6.Διάγραμμα Δραστηριοτήτων (2 ^η έκδοση).....	23
5.1.7.Διάγραμμα Καταστάσεων (2 ^η έκδοση)	24
5.1.8.Διάγραμμα Εξαρτημάτων (2 ^η έκδοση).....	25
5.1.9.Διάγραμμα Διανομής (2 ^η έκδοση).....	26
6.Εγχειρίδιο Χρήστη	27

1. Εισαγωγή

1.1. Στόχοι της εργασίας

Στόχος της συγκεκριμένης εργασίας είναι η ανάπτυξη μίας εφαρμογής ραντεβού ασθενών-γιατρών. Οι χρήστες της εφαρμογής χωρίζονται σε δύο κατηγορίες, τους ιατρούς και τους ασθενείς. Μέσα από την εφαρμογή οι ιατροί θα μπορούν να καταχωρούν το προφίλ τους και οι ασθενείς θα μπορούν να κλείσουν τα ιατρικά ραντεβού τους. Κατά την ανάπτυξη της εφαρμογής θα χρησιμοποιήσουμε το μοντέλο RUP (Rational Unified Process) που αποτελεί ένα μοντέλο κύκλου ζωής λογισμικού, για να μας βοηθήσει την καλύτερη ανάπτυξη της εφαρμογής μας.

1.2. Ορισμός του προβλήματος προς επίλυση

Η συγκεκριμένη εφαρμογή έρχεται να λύσει ένα πλήθος προβλημάτων, αξιοποιώντας τεχνολογίες αιχμής. Τα πιο σημαντικά προβλήματα που καλείται να καλύψει είναι τα παρακάτω:

- Για τους ασθενείς:
 - Ένα από τα πιο συχνά προβλήματα που καλείται να αντιμετωπίσει ένας ασθενής είναι ότι πρέπει να κάνει τεράστια έρευνα, μέσω διαφόρων πηγών για να βρει την ειδικότητα γιατρού που θέλει και σε σχετικά κοντινή τοποθεσία.
 - Ένα άλλο πρόβλημα που αντιμετωπίζει ένας ασθενής είναι η χρονοβόρα διαδικασία πληροφόρησης σχετικά με το ωράριο των γιατρών και τα διαθέσιμα ραντεβού που έχουν, η οποία απαιτεί να βρει πρώτα το σχετικό ωράριο και μετά να πάρει τηλέφωνο σε ώρες εργασίας του γιατρού, ώστε να κλείσει το ραντεβού.
 - Τέλος, πολλές φορές μπορεί ένας ασθενής αφού κλείσει ένα ραντεβού μπορεί να ξεχαστεί και να μην σημειώσει την ημερομηνία με ώρα ή να τα σημειώσει λάθος, με αποτέλεσμα είτε να πρέπει να καλέσει πάλι πίσω για να ενημερωθεί για το ραντεβού του είτε να χάσει το ραντεβού είτε να πάει σε λάθος ραντεβού που φυσικά δεν θα γίνει δεκτός.
- Για τους γιατρούς:
 - Ένα πρόβλημα που αντιμετωπίζουν οι γιατροί, ιδίως όταν είναι στο ξεκίνημα, είναι αυτό την διαφήμισης – γνωστοποίησης της παρουσίας τους σε μία περιοχή και κατ' επέκταση των διαθέσιμων ραντεβού του.
 - Ένα άλλο πρόβλημα είναι η σωστή διαχείριση των ραντεβού όταν ιδίως αυτά έρχονται από πολλαπλές πηγές όπως τηλεφωνικά ραντεβού, επαναληπτικών ραντεβού που καθορίζονται στο τέλος της επίσκεψης ενός ασθενή κ.α.
 - Τέλος, ένα άλλο πρόβλημα που αντιμετωπίζουν οι γιατροί είναι, η ενημέρωση των ασθενών σε περίπτωση αλλαγής ωραρίου.

Οι λύσεις που θα προσφέρει η εφαρμογή μας έχει πολλαπλά οφέλη για διάφορες κατηγορίες χρηστών που παρουσιάζονται παρακάτω μαζί με τα οφέλη ανά κατηγορία:

Κατηγορίες χρηστών:

- Ιδιώτες ασθενείς
- Γιατροί ιδιωτικών ιατρείων
- Γιατροί ιδιωτικών νοσοκομείων
- Γιατροί δημόσιων νοσοκομείων

Για την πρώτη κατηγορία, η εφαρμογή μας έρχεται να προσφέρει τις παρακάτω λύσεις:

- Αποδοτική και γρήγορη εύρεση γιατρών ανά ειδικότητα και ανά περιοχή.
- Άμεση ενημέρωση για τα διαθέσιμα ραντεβού των γιατρών αυτών.
- Γρήγορη και εύκολη καταχώρηση ραντεβού.
- Καλύτερη οργάνωση των ραντεβού του, ώστε να μπορεί να ελέγξει και να ενημερωθεί ανά πάσα στιγμή για τα ραντεβού που έχει κλείσει.

Για τις τρεις τελευταίες κατηγορίες, η εφαρμογή μας έρχεται να προσφέρει τις παρακάτω λύσεις:

- Εύκολη και αποτελεσματική προβολή των γιατρών προς μεγαλύτερο πλήθος ασθενών.
- Ενημέρωση προς τους ασθενείς για την ειδικότητα, την τοποθεσία και των διαθέσιμων - ελεύθερων ραντεβού.
- Καλύτερη οργάνωση των ραντεβού τους και εύκολη δημιουργία ή διαγραφή ραντεβού από μια πηγή.

2. Σύνοψη παρουσίαση της RUP

Η Rational Unified Process (RUP) είναι μια διαδικασία ανάπτυξης λογισμικού για αντικειμενοστραφή μοντέλα. Η RUP αποτελείται από ένα σύνολο οδηγιών σχετικά με τις τεχνικές και οργανωτικές απόψεις της ανάπτυξης λογισμικού και αφορά κυρίως στην Ανάλυση Απαιτήσεων και στο Σχεδιασμό. Δημιουργήθηκε από την Rational Software Corporation, τμήμα της IBM από το 2003. Ο στόχος της είναι να διασφαλίσει την παραγωγή λογισμικού υψηλής ποιότητας που ικανοποιεί τις ανάγκες των τελικών χρηστών μέσα σε ένα συγκεκριμένο χρονοδιάγραμμα και κόστος.

Η RUP προτείνεται από τους Ivar Jacobson, Grady Bootch, and James Rumbaugh για την ανάπτυξη λογισμικού. Ο κύκλος ζωής λογισμικού προτείνεται να είναι επαναληπτικός, η ανάπτυξη δηλαδή να προχωράει σε μια σειρά επαναλήψεων μέχρι να εξελιχθεί το τελικό προϊόν.

Η RUP είναι δομημένη σε δύο διαστάσεις:

1. Χρόνο-χωρισμός του κύκλου ζωής σε φάσεις και επαναλήψεις.
2. Τμήματα διαδικασίας - Καλά ορισμένες εργασίες.

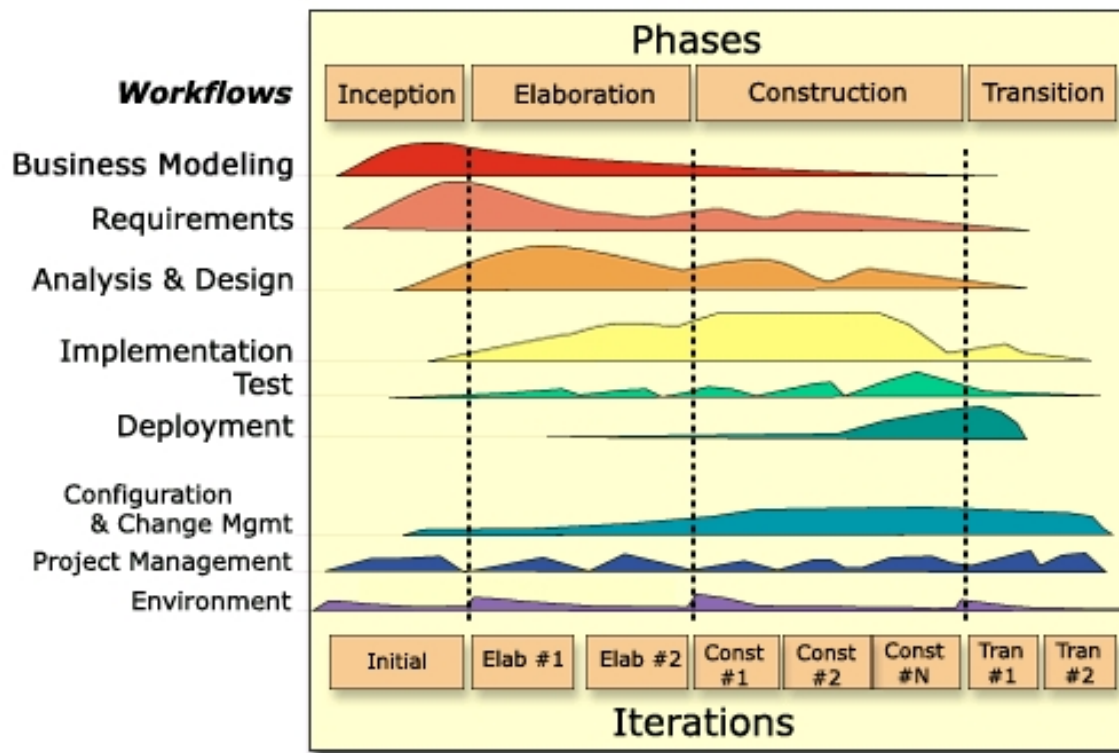
Η δόμηση ενός έργου σε σχέση με το χρόνο ακολουθεί τις εξής φάσεις που έχουν σχέση με το χρόνο:

1. Έναρξη (Inception): Καθορίζει την προοπτική του έργου.
2. Εκπόνηση μελέτης (Elaboration): Σχεδιασμός των απαιτούμενων δραστηριοτήτων και πόρων. Καθορισμός των χαρακτηριστικών και σχεδιασμός της αρχιτεκτονικής.
3. Κατασκευή (Construction): Ανάπτυξη του προϊόντος σε μια σειρά βηματικών επαναλήψεων.
4. Μετάβαση (Transition): Προμήθεια του προϊόντος στην κοινότητα χρηστών (παραγωγή, διανομή, εκπαίδευση).

Η δόμηση έργου σύμφωνα με τη διάσταση των τμημάτων διαδικασίας περιλαμβάνει τις ακόλουθες δραστηριότητες:

1. Σύλληψη απαιτήσεων (Requirements capture): Μια αφήγηση του τι πρέπει να κάνει το σύστημα.
2. Ανάλυση και σχεδιασμός (Analysis and design): Μια περιγραφή του πως θα υλοποιηθεί το σύστημα.
3. Υλοποίηση (Implementation): Η παραγωγή του κώδικα.
4. Έλεγχος (Test): Η επαλήθευση του συστήματος.

Κύκλος Ζωής Ανάπτυξης Λογισμικού:



3. Φάση: Έναρξη (Inception)

3.1. Σύλληψη απαιτήσεων

Η εφαρμογή που θα αναπτύξουμε θα περιλαμβάνει 3 ήδη χρηστών. Τους επισκέπτες που θα είναι όλοι οι users πριν εγγραφούν, τους γιατρούς και τους ασθενείς. Για τον καθένα από τους παραπάνω users έχουμε τις παρακάτω απαιτήσεις:

Επισκέπτης

Για τους επισκέπτες δεν θα κρατείται κάποιο στοιχείο στη βάση, αλλά ένα user ως επισκέπτης, θα έχει πρόσβαση στην φόρμα σύνδεσης στην πλατφόρμας, μέσω της οποίας θα μπορεί είτε να εγγραφεί ως γιατρός ή ως ασθενής, να συνδεθεί ως γιατρός ή ως ασθενής ή να συνεχίσει ως επισκέπτης. Αν επιλέξει να συνεχίσει ως επισκέπτης θα έχει πρόσβαση μόνο σε μια σελίδα που θα παρουσιάζει πληροφορίες για τους γιατρούς του συστήματος.

Γιατρός

Για τους γιατρούς που είναι εγγεγραμμένοι στην εφαρμογή θα κρατούνται τα παρακάτω στοιχεία:

- AMA
- Κωδικός
- Ονοματεπώνυμο
- Ειδικότητα
- Τηλέφωνο
- Email
- Διεύθυνση ιατρείου
- Ταχυδρομικός κώδικας
- Περιοχή

Ο χρήστης που θα είναι γιατρός θα μπορεί:

- Να εγγραφεί στο σύστημα εισάγοντας τιμές στα παραπάνω πεδία.
- Να συνδεθεί στο σύστημα με AMA και Κωδικό.
- Να επεξεργάζεται το προφίλ του.
- Να προσθέτει ραντεβού.
- Να βλέπει τα ραντεβού του ανά εβδομάδα και ανά μήνα.
- Να ακυρώνει ραντεβού.

Ασθενής

Για τους ασθενείς που είναι εγγεγραμμένοι στην εφαρμογή θα πρέπει να κρατούνται στοιχεία:

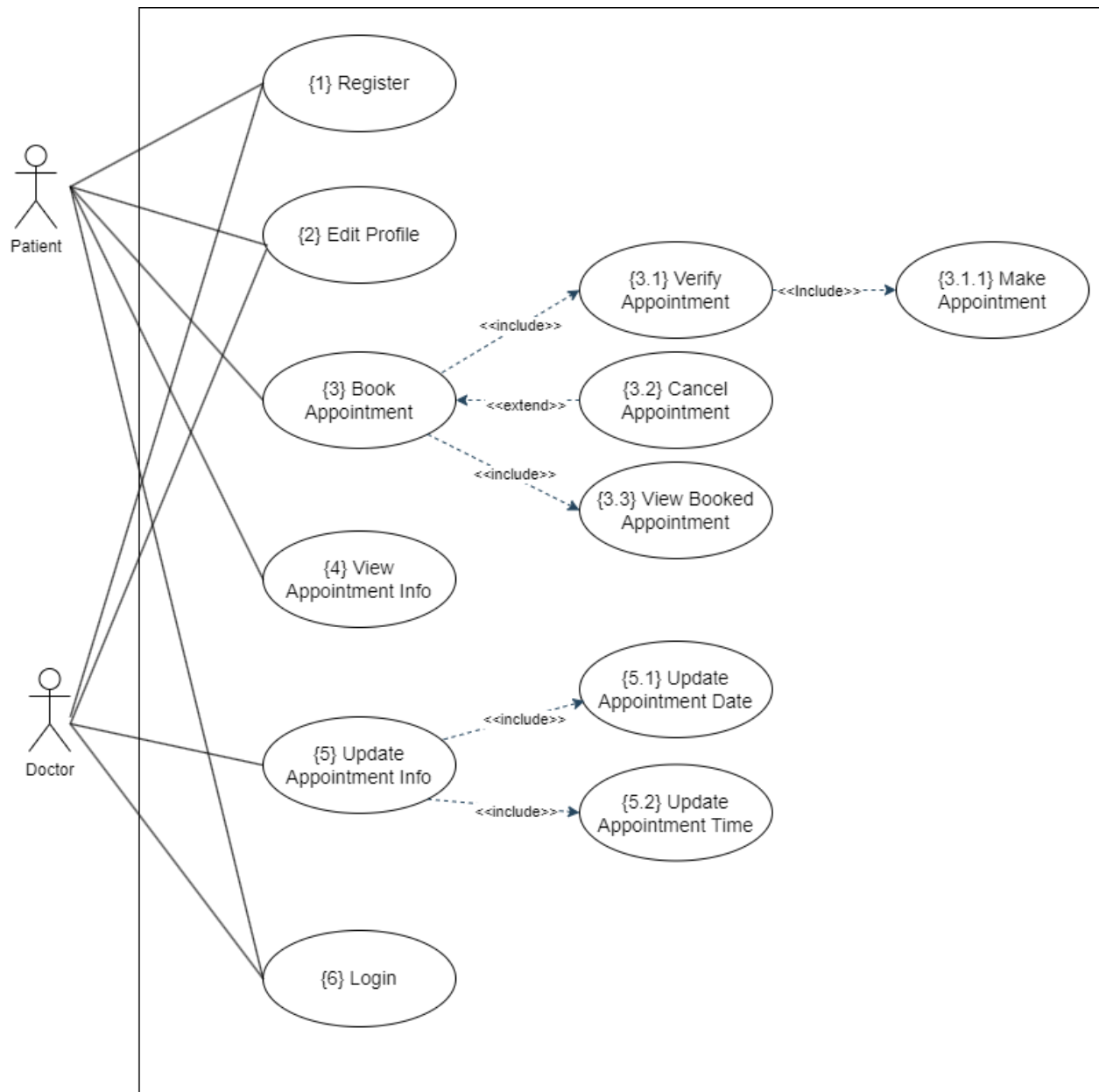
- AMKA
- Κωδικός
- Ονοματεπώνυμο
- Τηλέφωνο
- Email

Ο χρήστης που θα είναι ασθενής θα μπορεί:

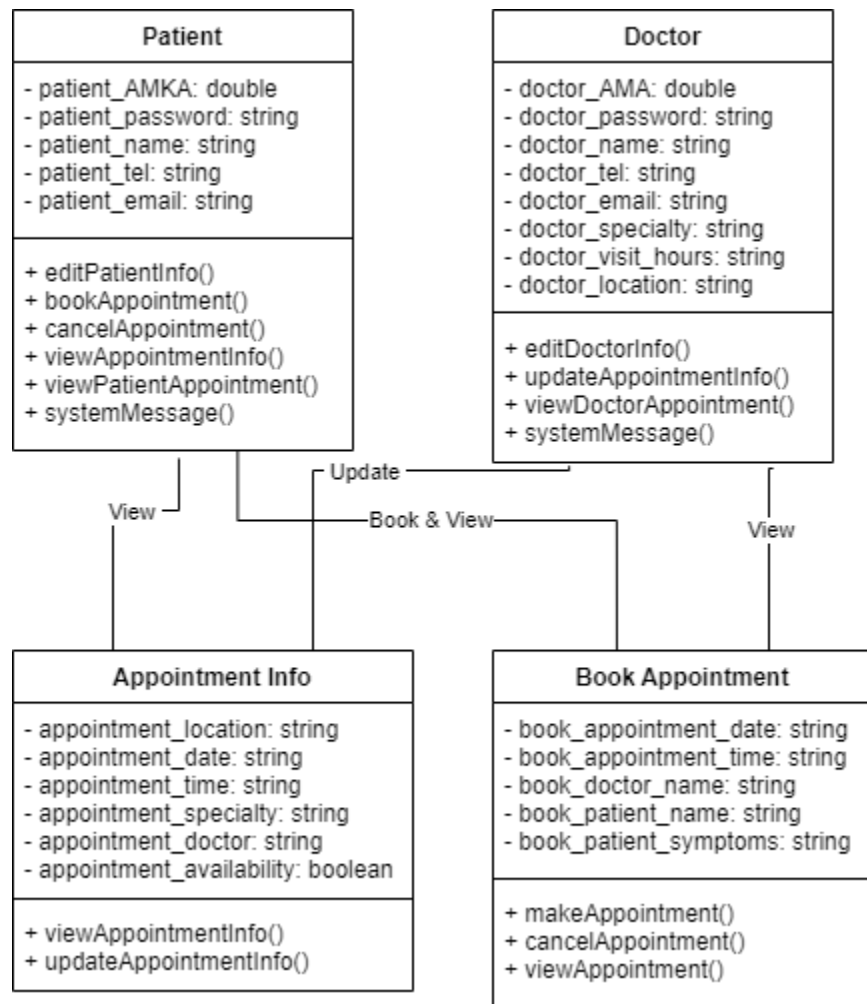
- Να εγγραφεί στο σύστημα εισάγοντας τιμές στα παραπάνω πεδία.
- Να συνδεθεί στο σύστημα με AMA και Κωδικό.
- Να κλείνει ραντεβού πραγματοποιώντας τα παρακάτω βήματα:
 - ο Θα επιλέγει την ειδικότητα που επιθυμεί και την περιοχή που τον εξυπηρετεί.
 - ο Θα επιλέγει τον ιατρό με τον οποίο θέλει να κλείσει ραντεβού.
 - ο Θα επιλέγει ημέρα και ώρα με βάση τις ημερομηνίες και ώρες του γιατρού, που δεν είναι δεσμευμένες από άλλον ασθενή.
 - ο Θα γράφει τον λόγο της επίσκεψής του.
 - ο Θα κλείνει το ραντεβού.
- Να βλέπει τα ραντεβού του ανά εβδομάδα και ανά μήνα.
- Να ακυρώνει ραντεβού.

3.2. Ανάλυση – Σχεδιασμός

3.2.1. Διάγραμμα Περιπτώσεων Χρήσης (1^η έκδοση)



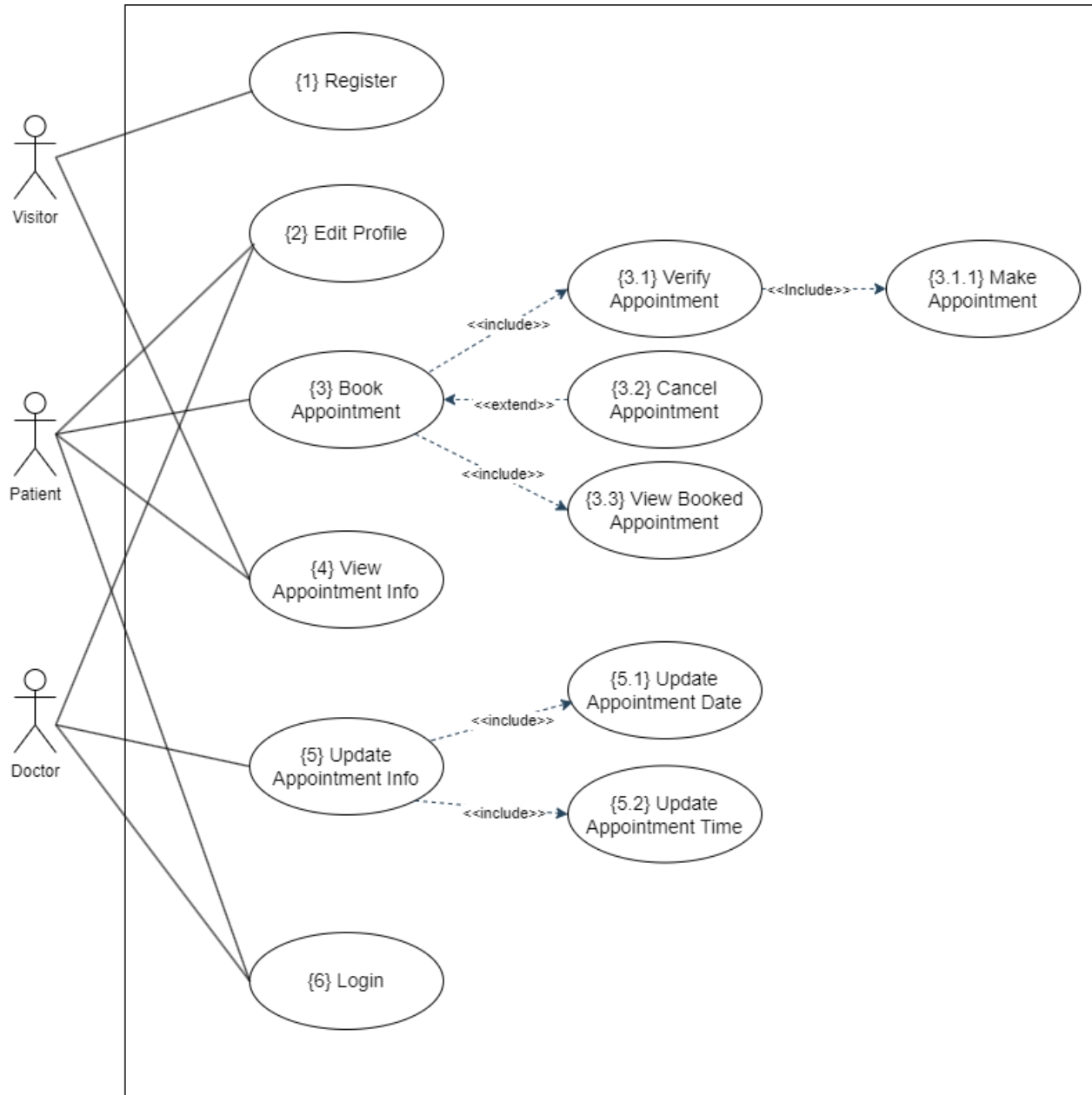
3.2.2. Διάγραμμα Τάξεων (1^η έκδοση)



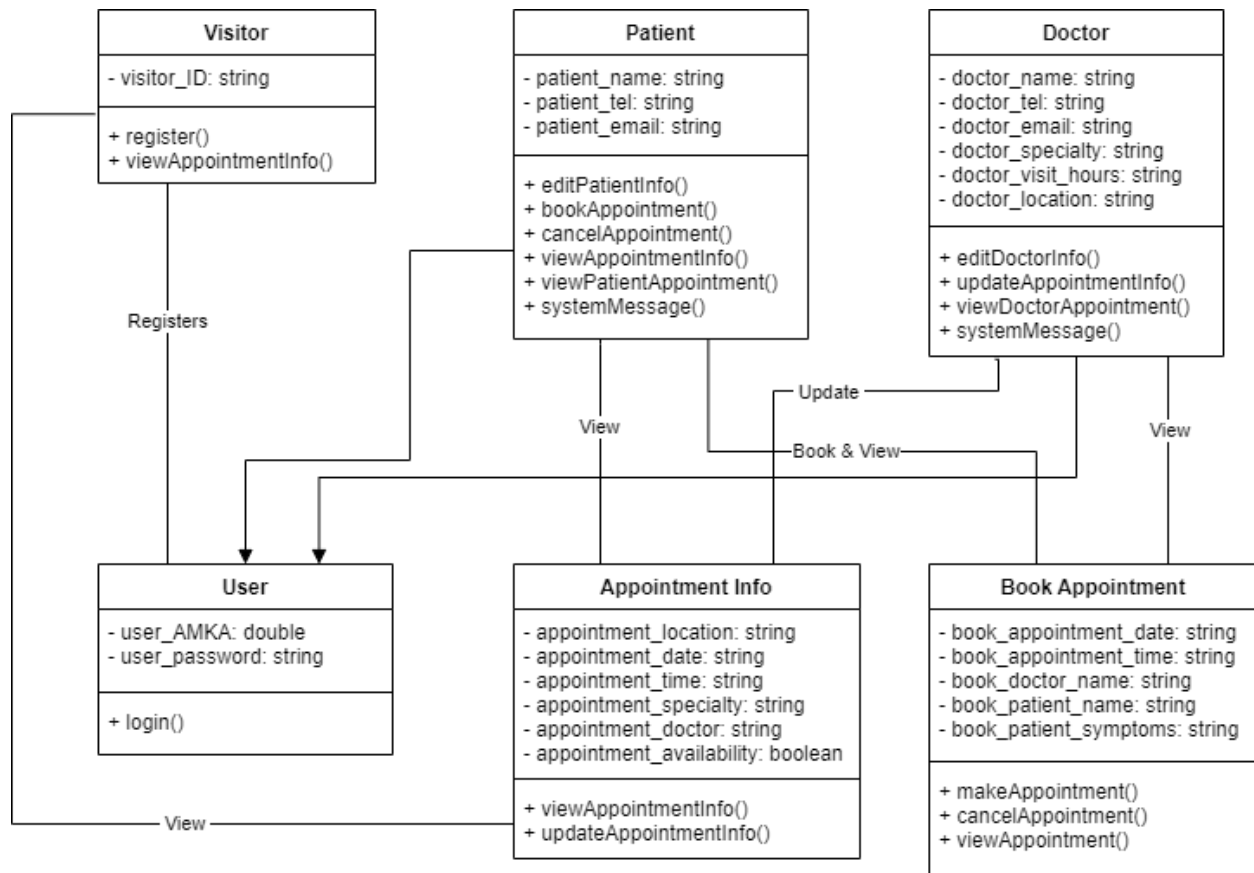
4. Φάση: Εκπόνηση Μελέτης (Elaboration)

4.1. Ανάλυση – Σχεδιασμός

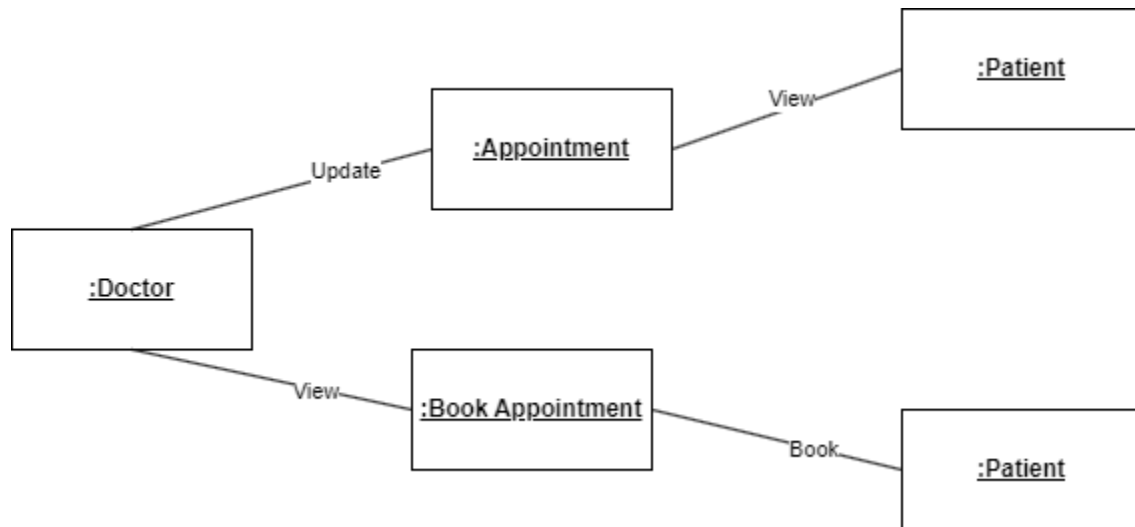
4.1.1. Διάγραμμα Περιπτώσεων Χρήσης (2^η έκδοση)



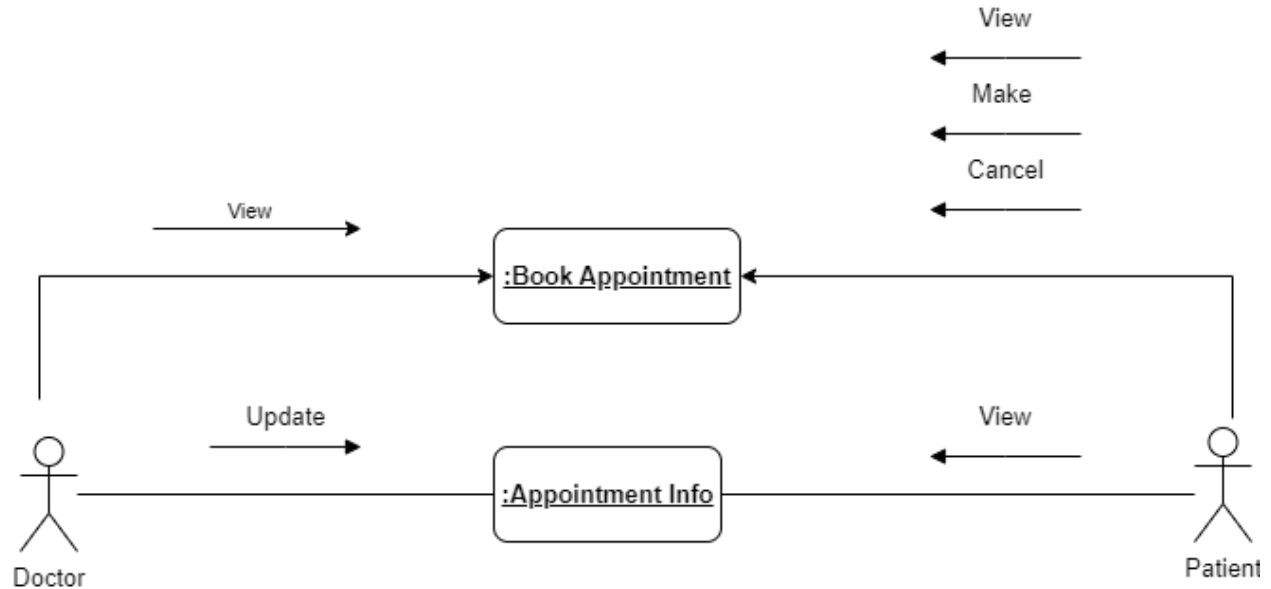
4.1.2. Διάγραμμα Τάξεων (2^η έκδοση)



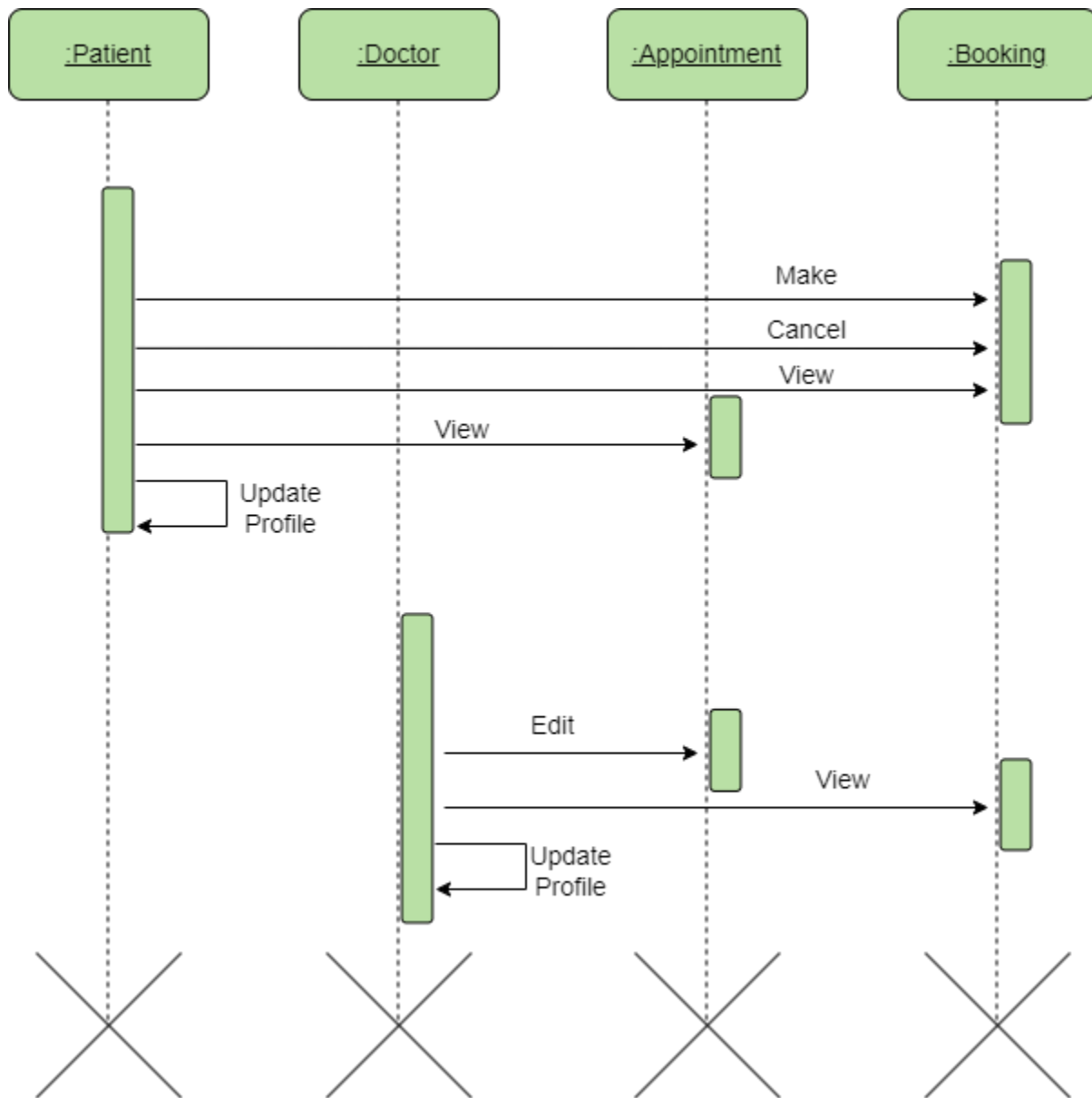
4.1.3. Διάγραμμα Αντικειμένων (1^η έκδοση)



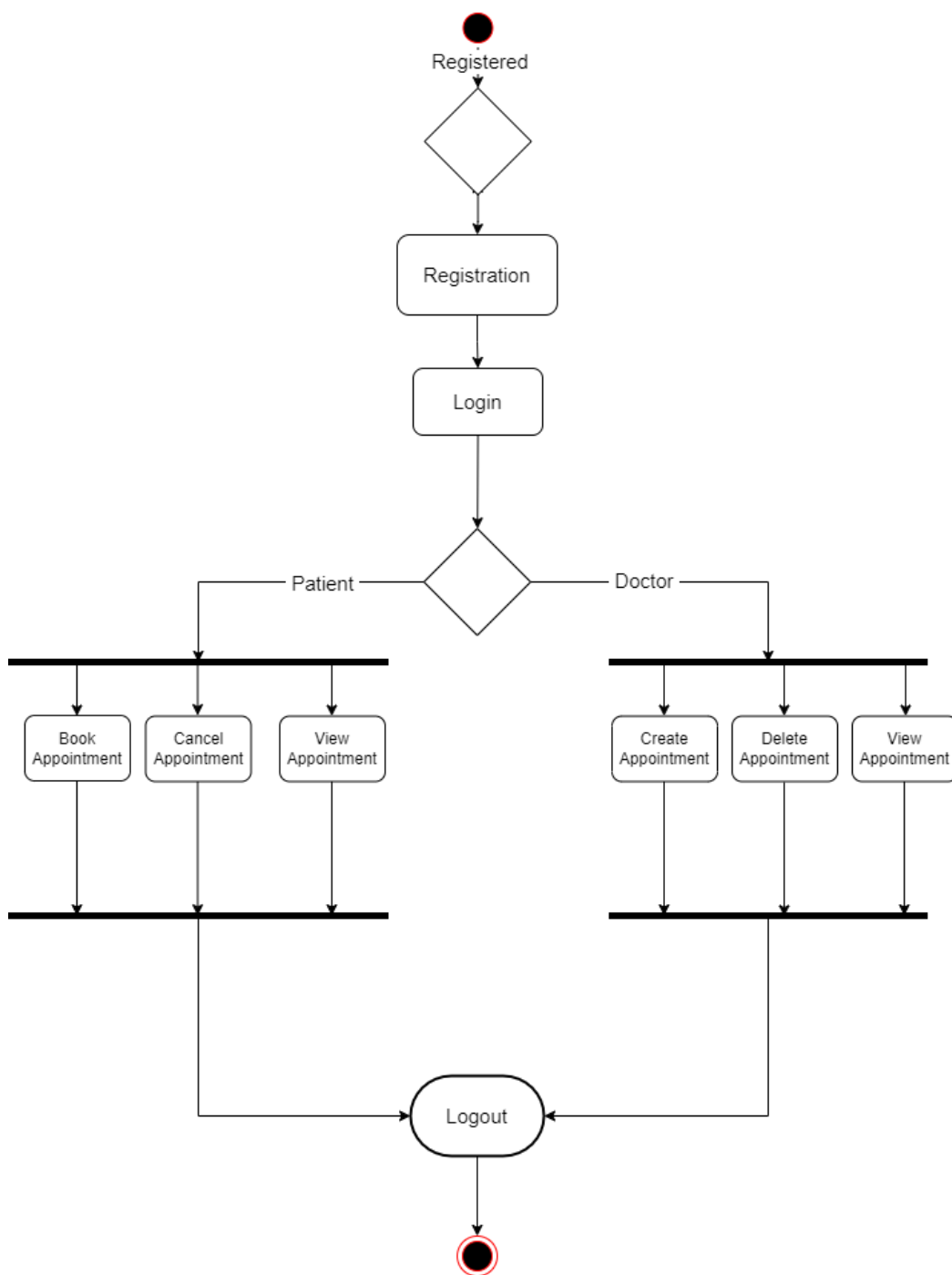
4.1.4. Διάγραμμα Συνεργασίας (1^η έκδοση)



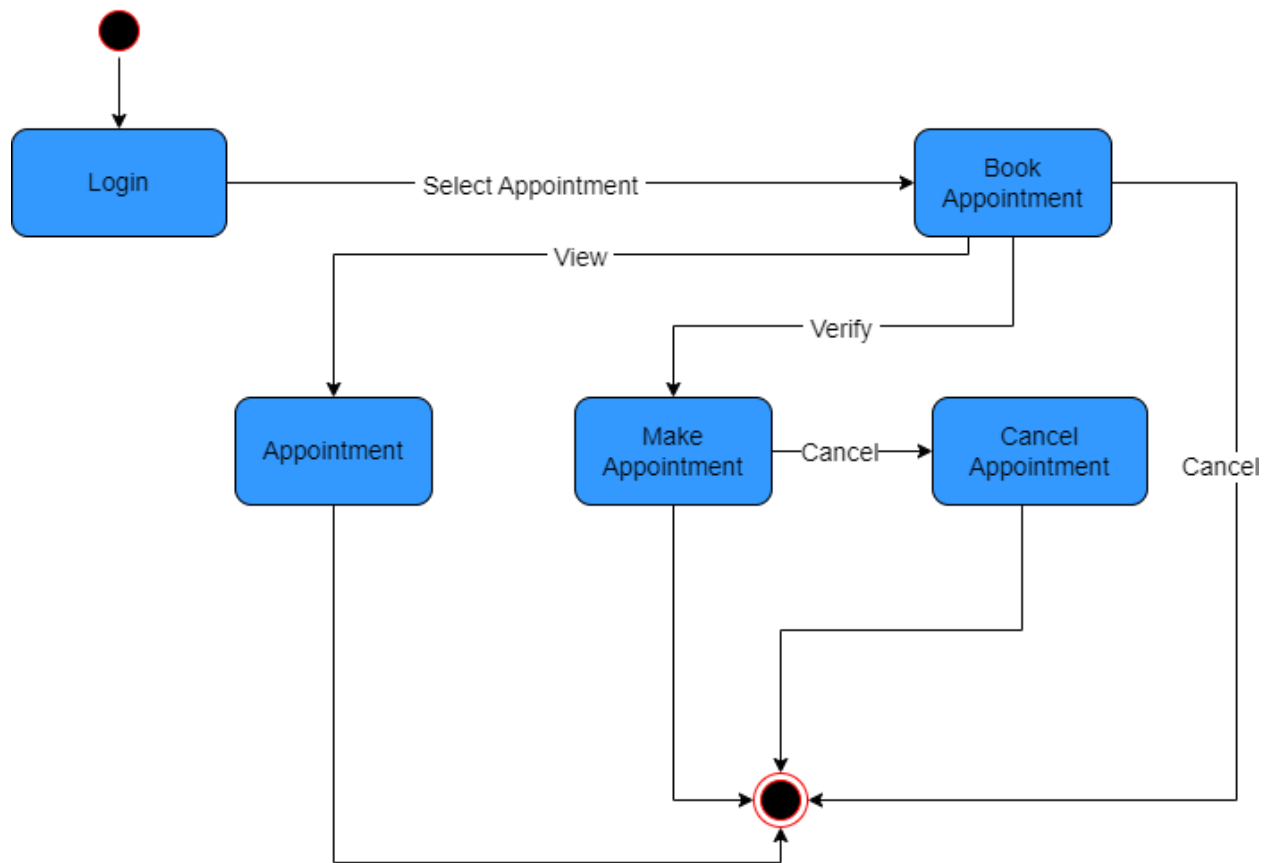
4.1.5. Διάγραμμα Σειράς (1^η έκδοση)



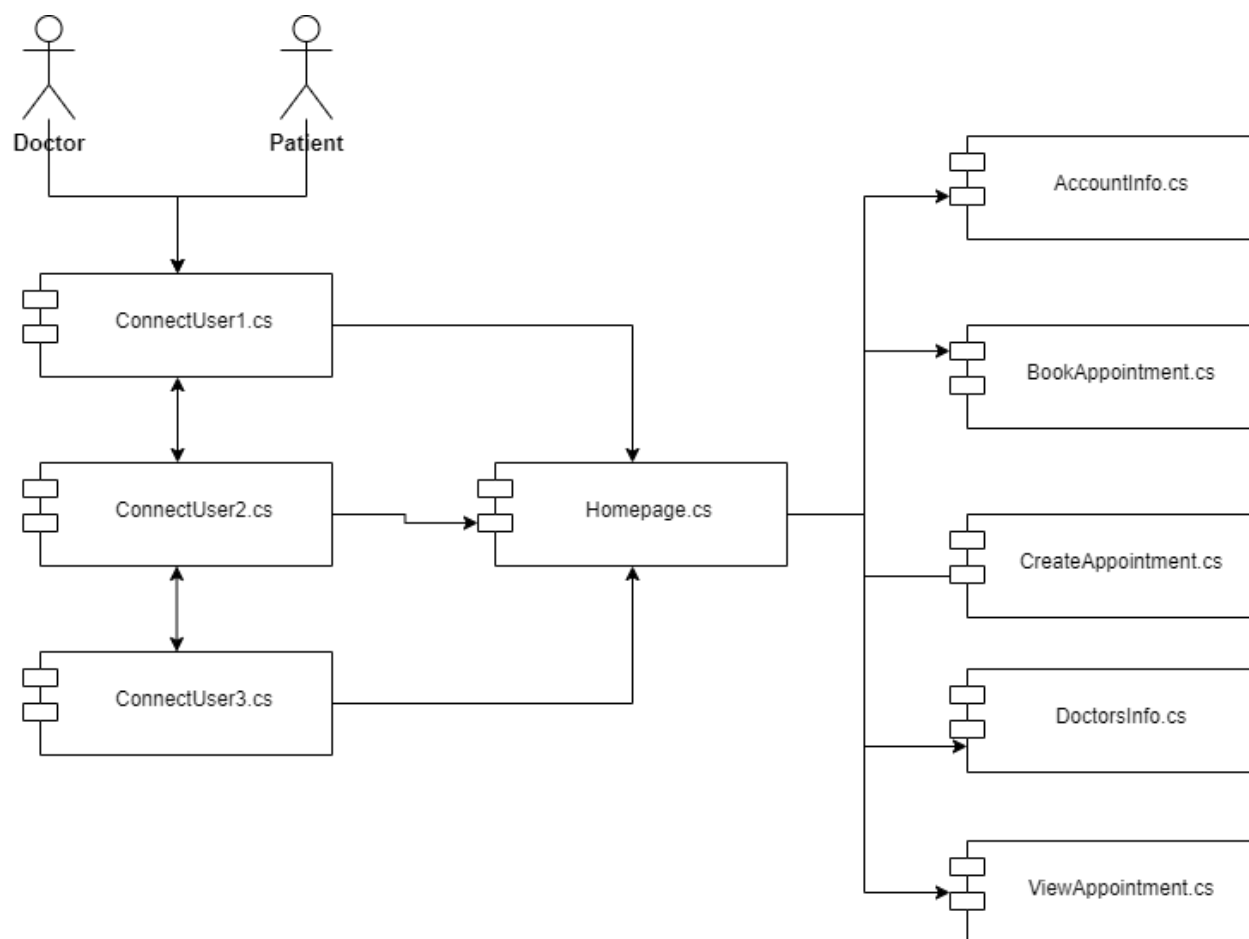
4.1.6. Διάγραμμα Δραστηριοτήτων (1^η έκδοση)



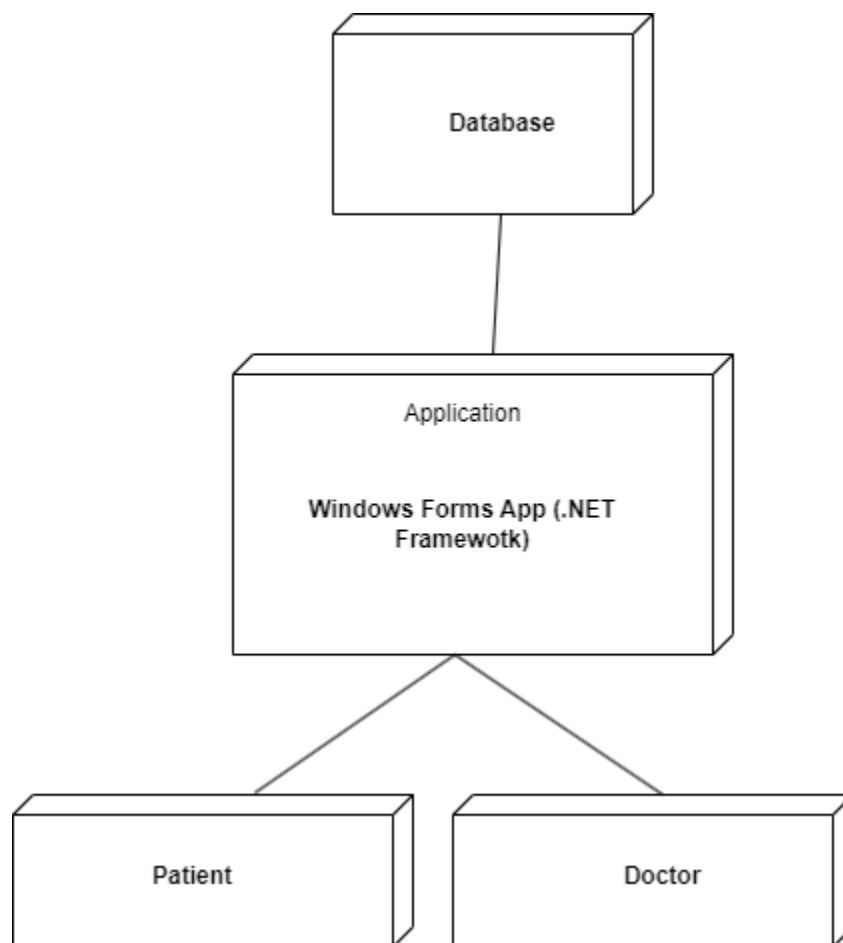
4.1.7. Διάγραμμα Καταστάσεων (1^η έκδοση)



4.1.8. Διάγραμμα Εξαρτημάτων (1^η έκδοση)



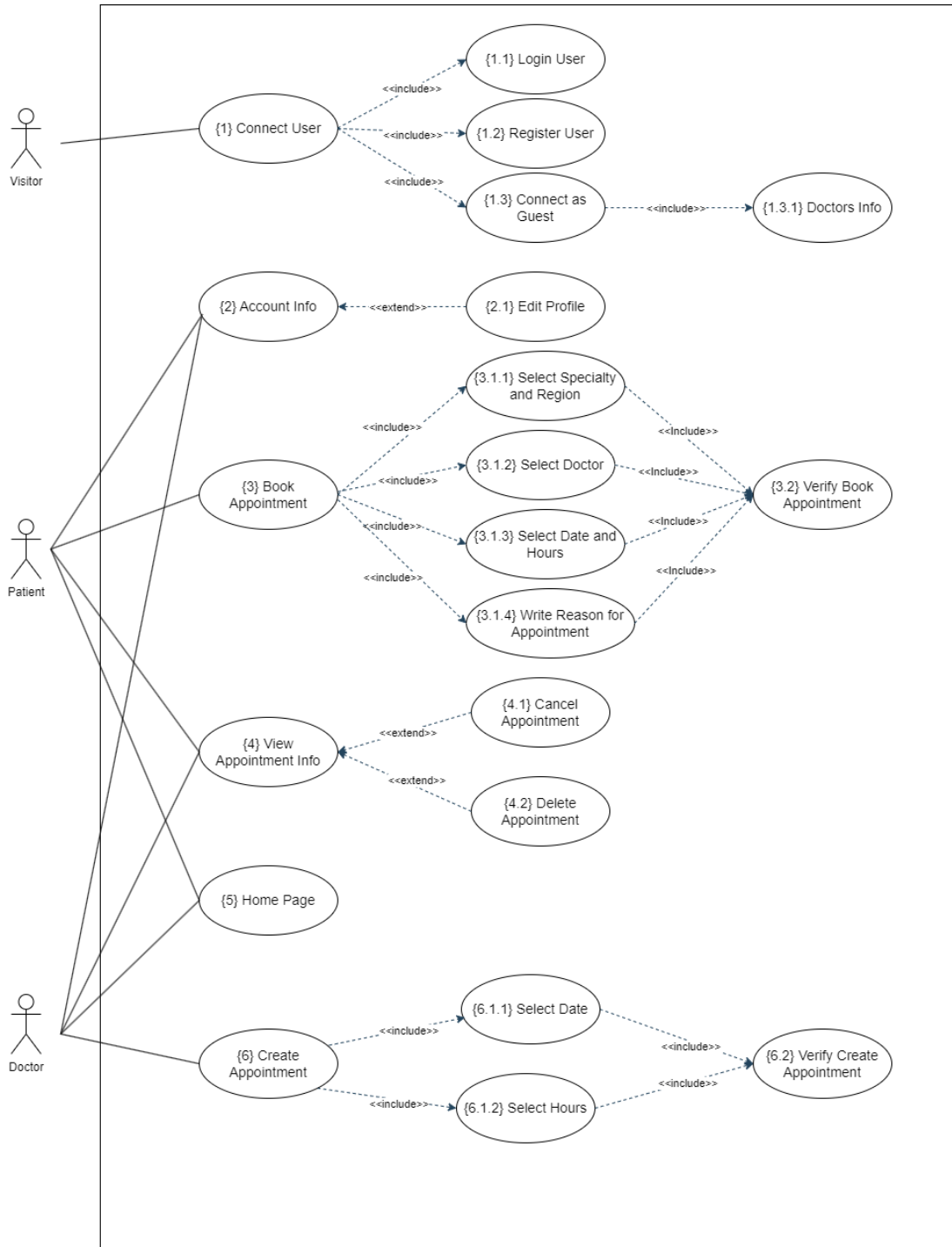
4.1.9. Διάγραμμα Διανομής (1^η έκδοση)



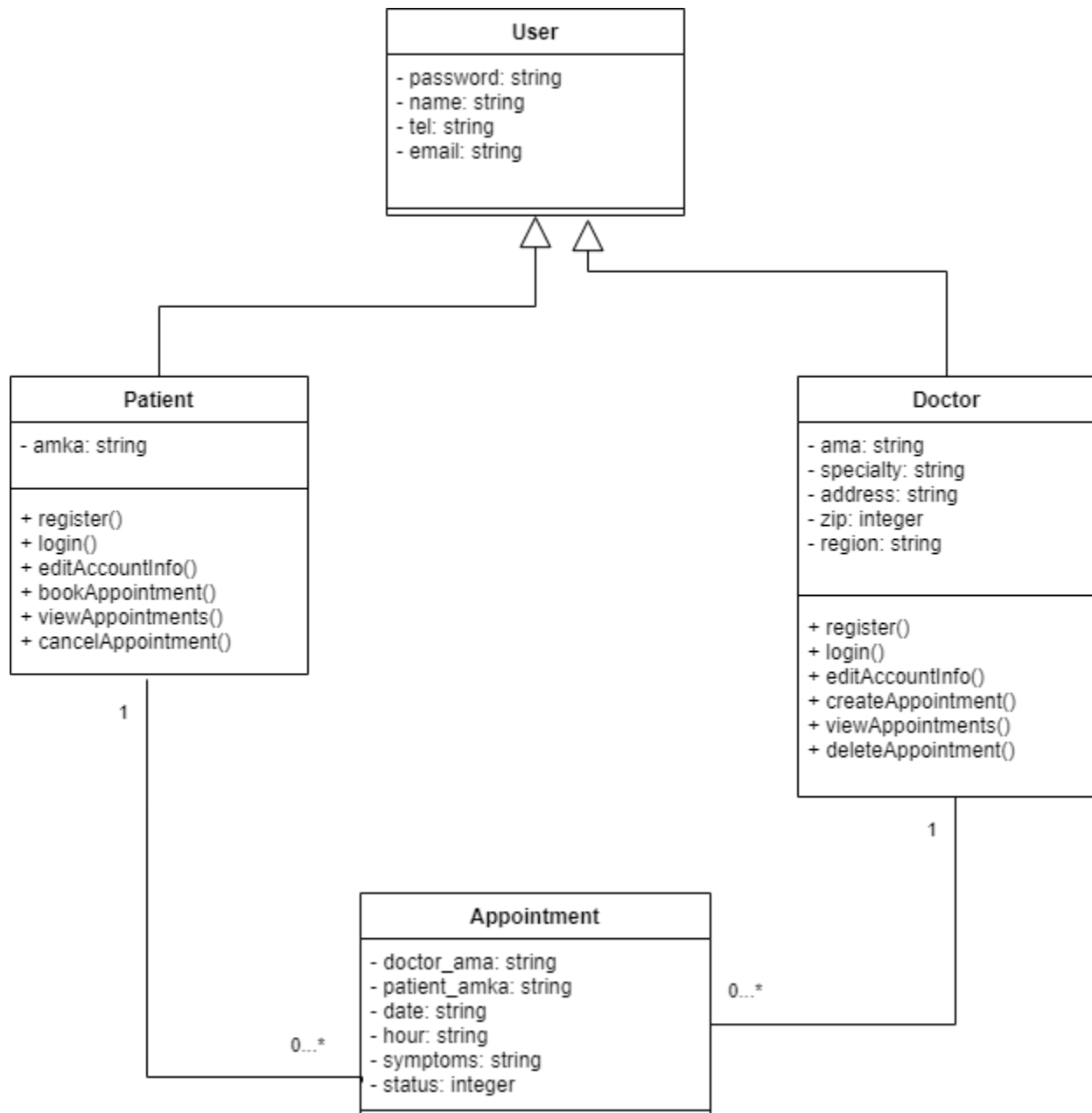
5. Φάση: Κατασκευή (Construction)

5.1. Ανάλυση – Σχεδιασμός

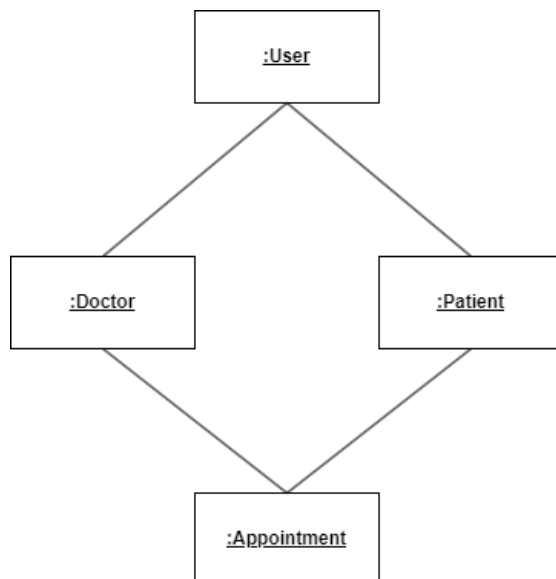
5.1.1. Διάγραμμα Περιπτώσεων Χρήσης (3^η έκδοση)



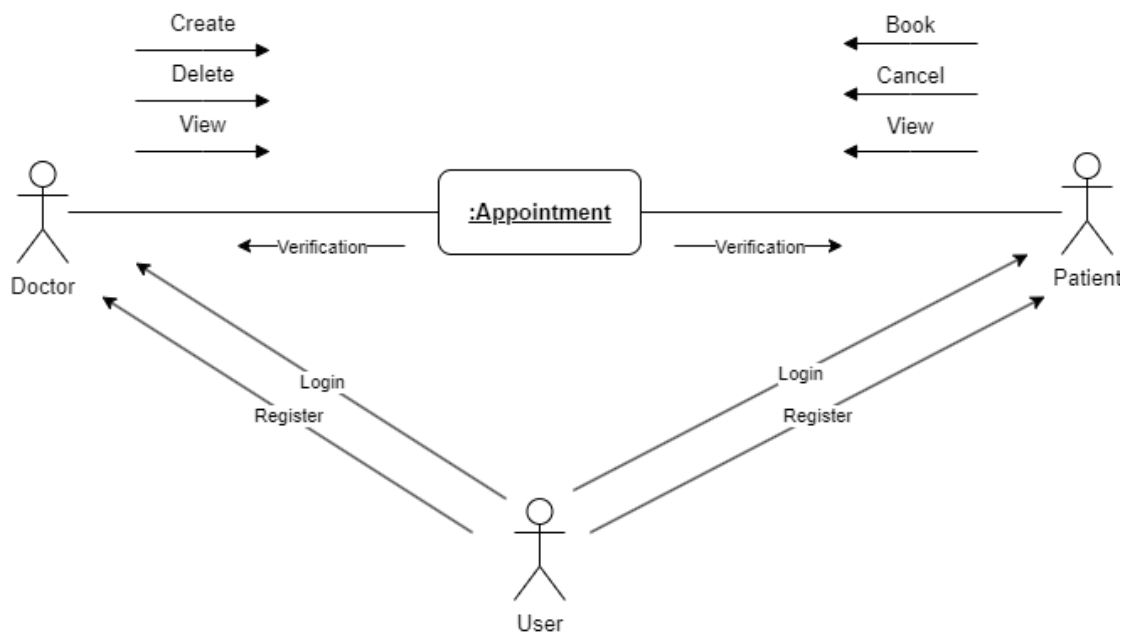
5.1.2. Διάγραμμα Τάξεων (3^η έκδοση)



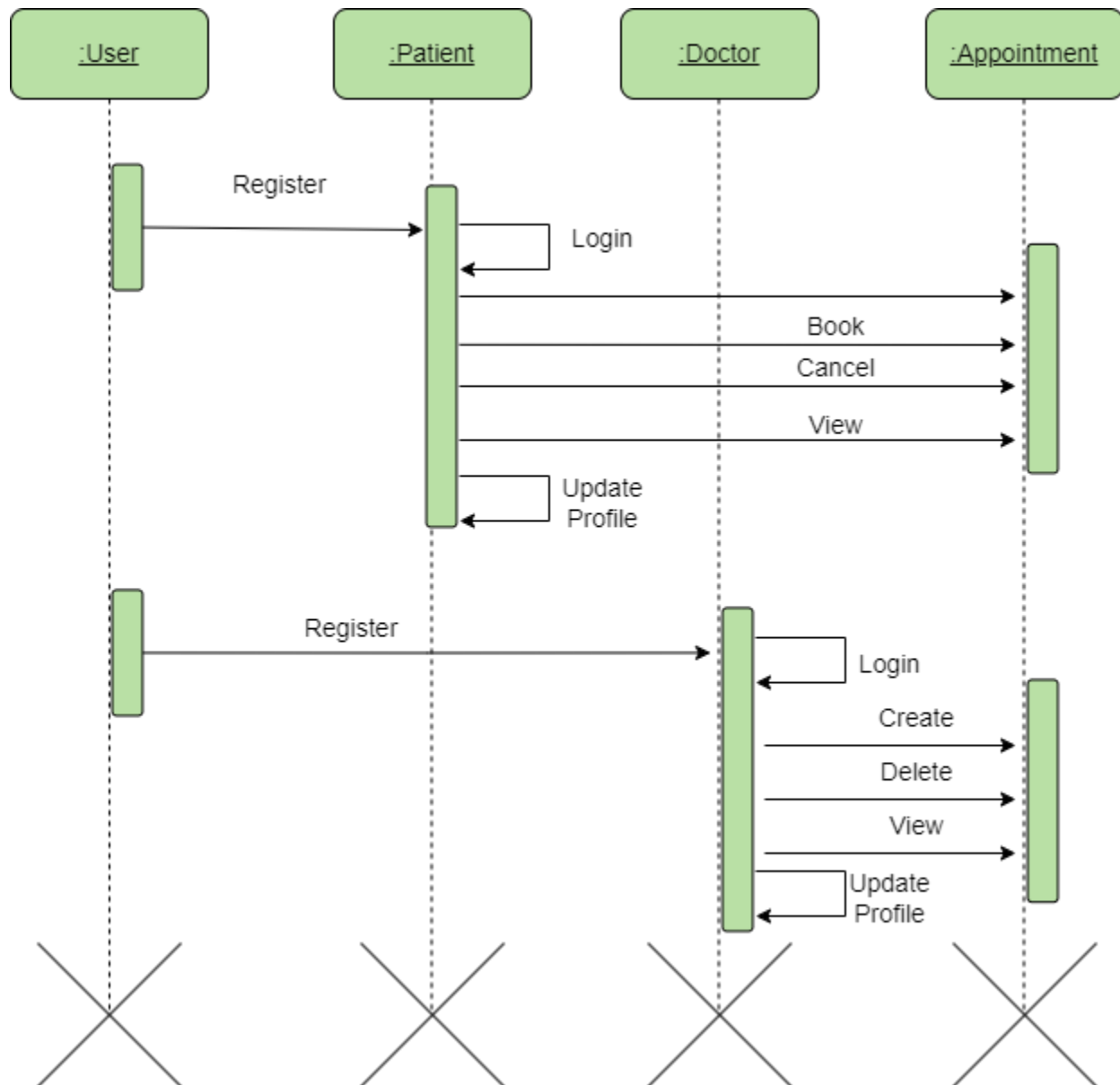
5.1.3. Διάγραμμα Αντικειμένων (2^η έκδοση)



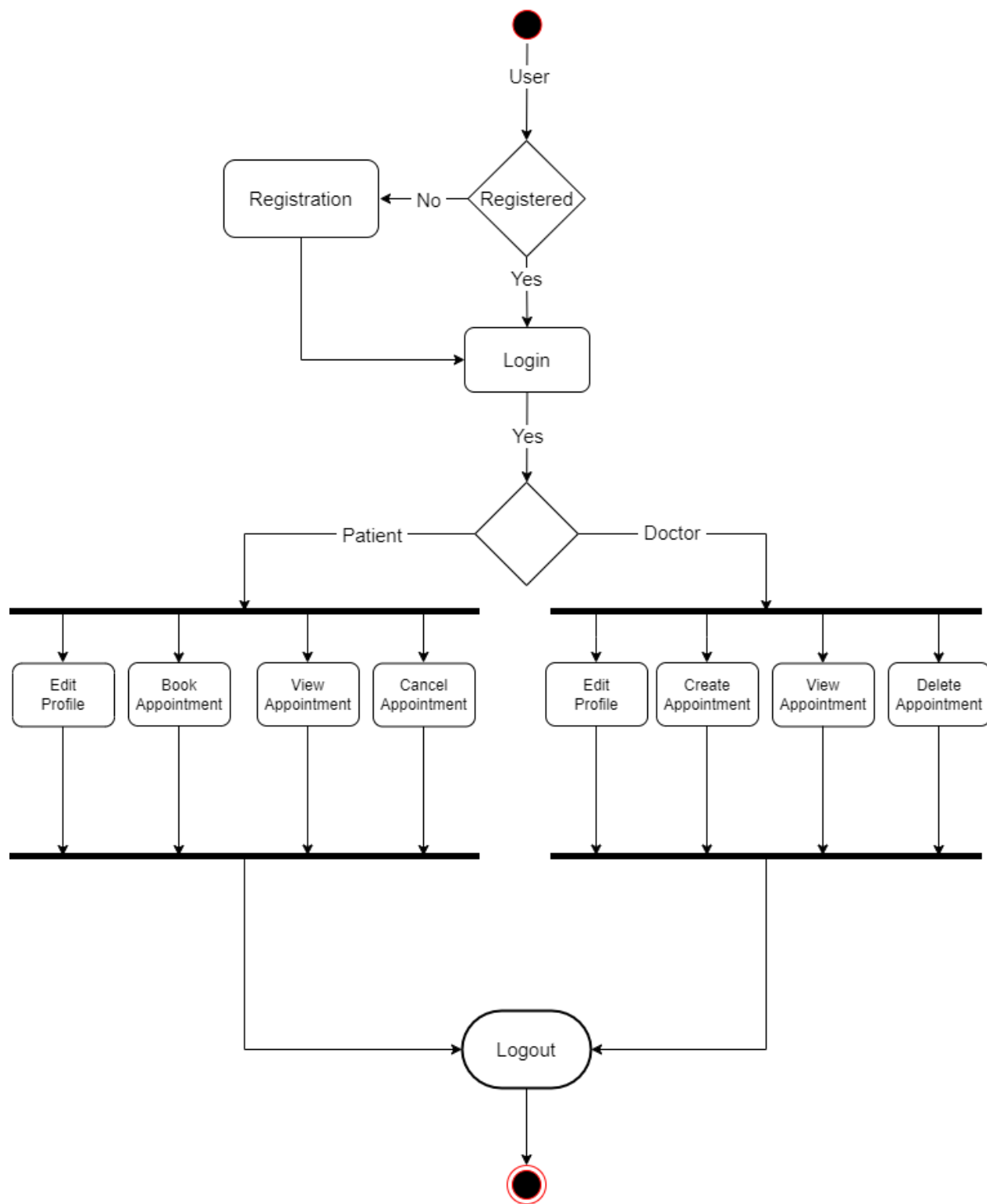
5.1.4. Διάγραμμα Συνεργασίας (2^η έκδοση)



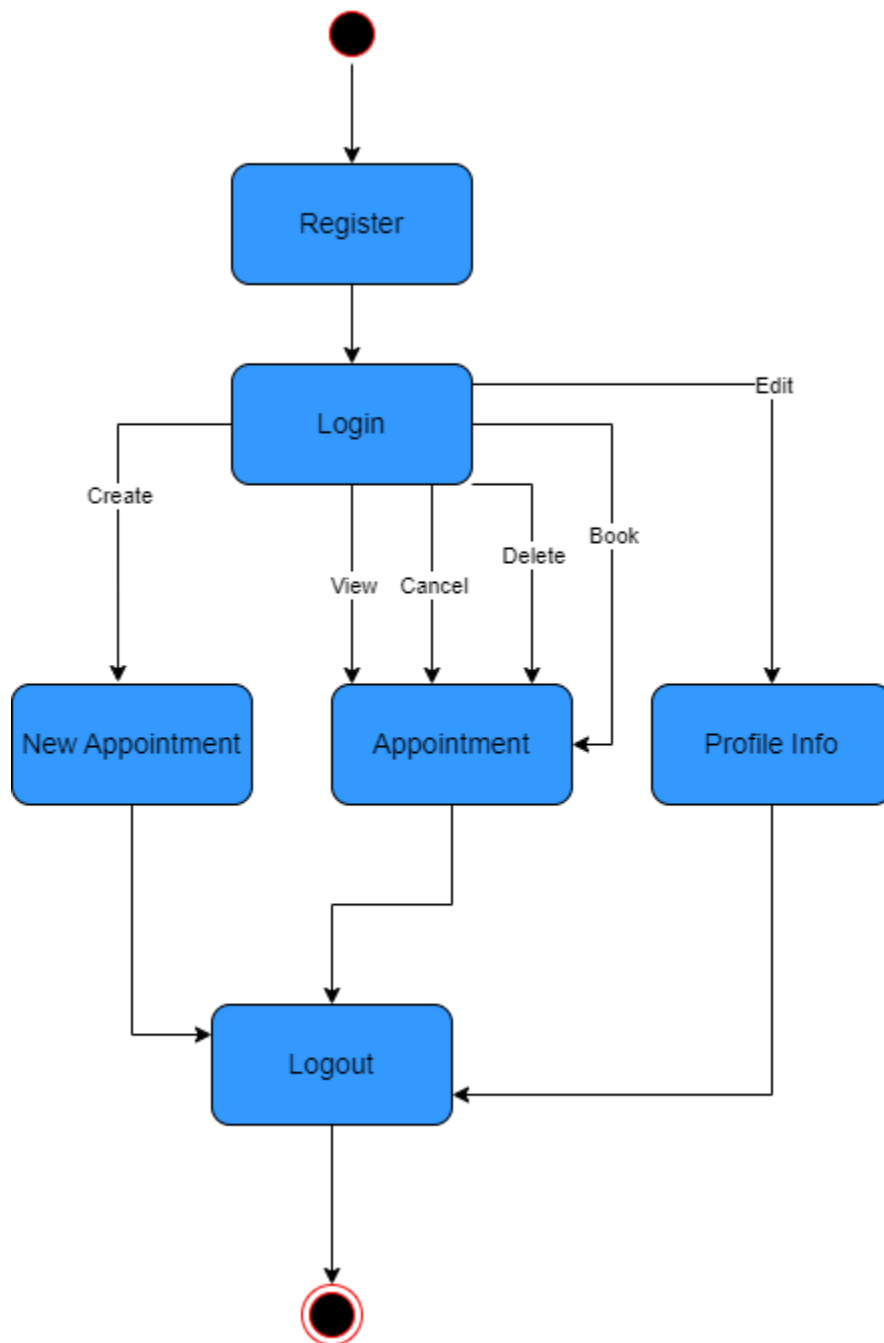
5.1.5. Διάγραμμα Σειράς (2^η έκδοση)



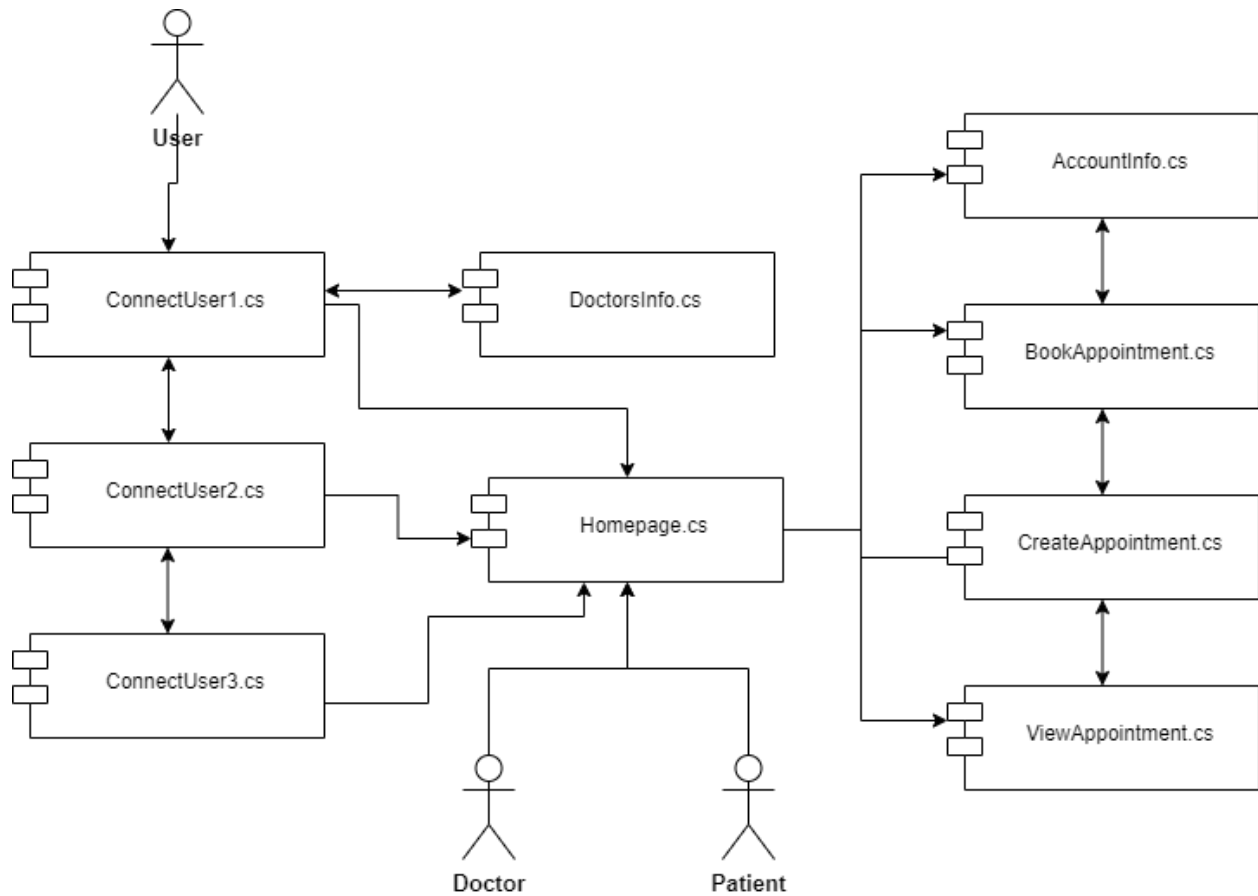
5.1.6. Διάγραμμα Δραστηριοτήτων (2^η έκδοση)



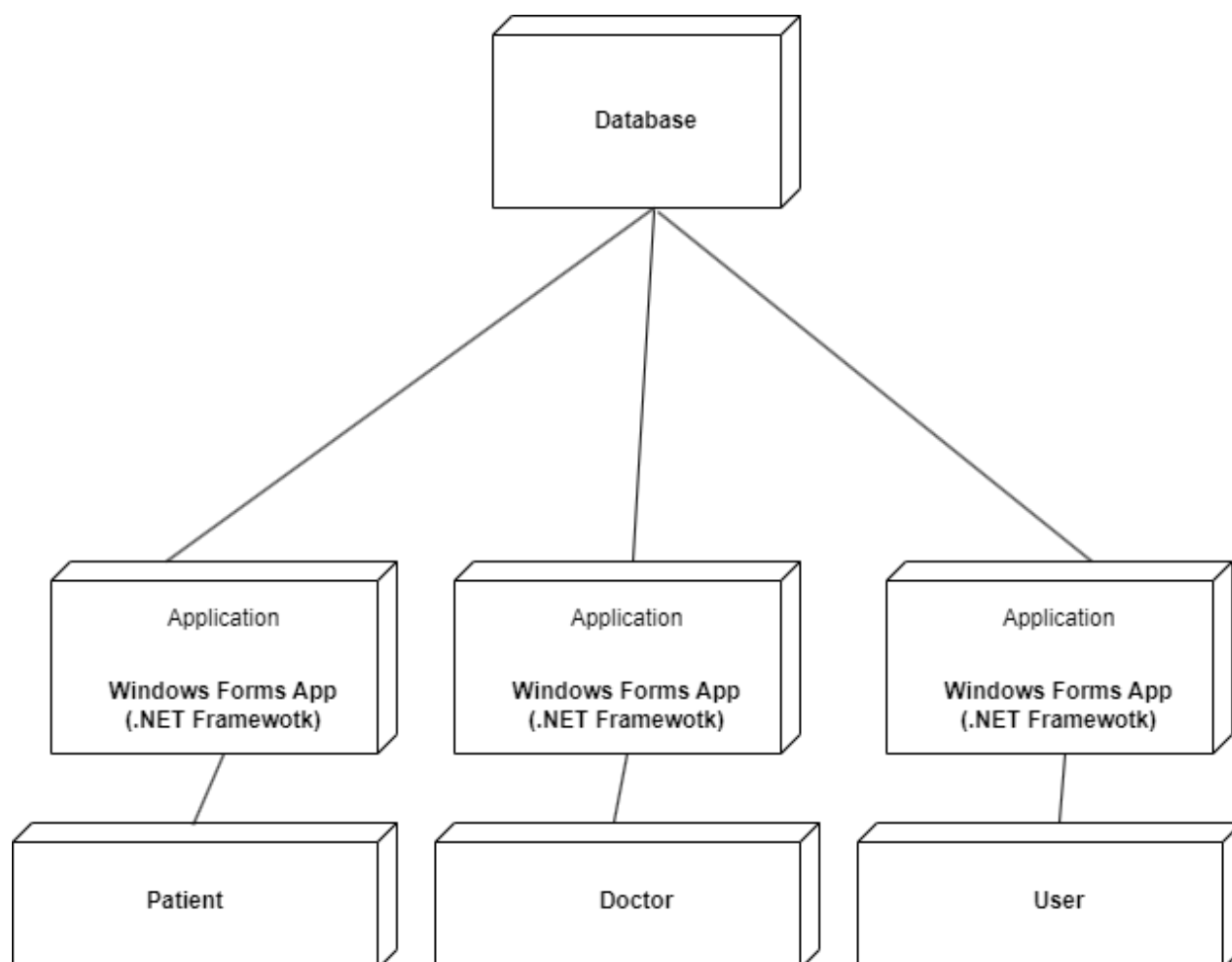
5.1.7. Διάγραμμα Καταστάσεων (2^η έκδοση)



5.1.8. Διάγραμμα Εξαρτημάτων (2^η έκδοση)

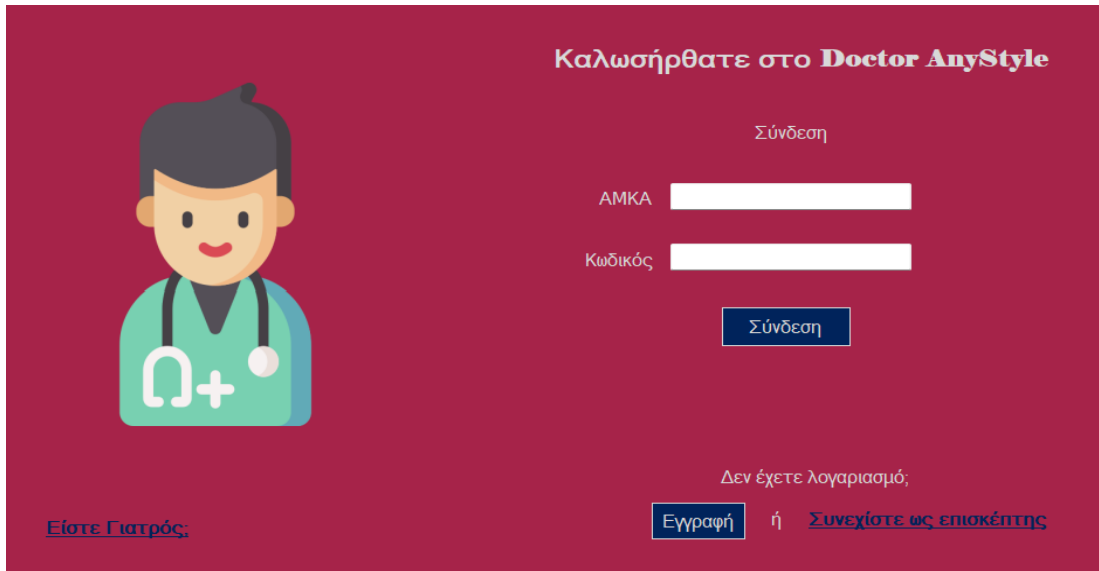


5.1.9. Διάγραμμα Διανομής (2^η έκδοση)



6. Εγχειρίδιο Χρήστη

Η αρχική οθόνη που βλέπει ο χρήστης ανοίγονταν την εφαρμογή είναι η οθόνη διασύνδεσης του με την εφαρμογή όπου του ζητάει να εισάγει στα αντίστοιχα πεδία τα απαραίτητα στοιχεία για την είσοδο του.



Καλωσήρθατε στο **Doctor AnyStyle**

Σύνδεση

ΑΜΚΑ

Κωδικός

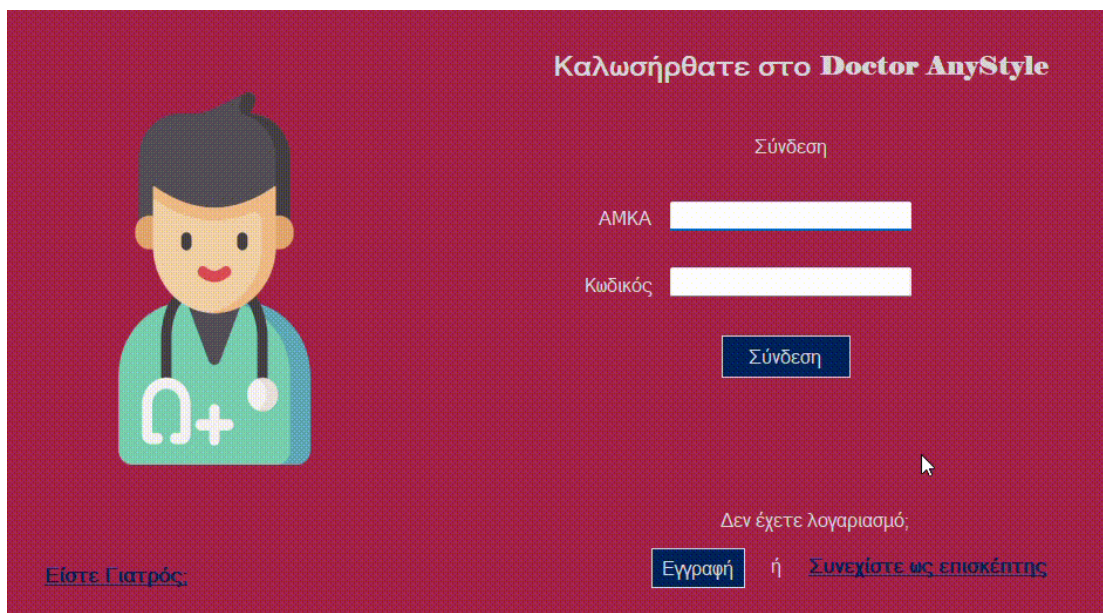
Σύνδεση

Δεν έχετε λογαριασμό;

[Εγγραφή](#) ή [Συνεχίστε ως επισκέπτης](#)

[Είστε Γιατρός;](#)

Στη συνέχεια αφού ο χρήστης/ασθενής εισάγει τα σωστά στοιχεία ΑΜΚΑ και Κωδικό Πρόσβασης και πατώντας στο κουμπί **Σύνδεση** θα οδηγηθεί στην κεντρική οθόνη της εφαρμογής όπου μπορεί να δει και να τροποποιήσει τα στοιχεία του λογαριασμού του, να δει τα ραντεβού του και να κλείσει ένα νέο ραντεβού.



Καλωσήρθατε στο **Doctor AnyStyle**

Σύνδεση

ΑΜΚΑ

Κωδικός

Σύνδεση

Δεν έχετε λογαριασμό;

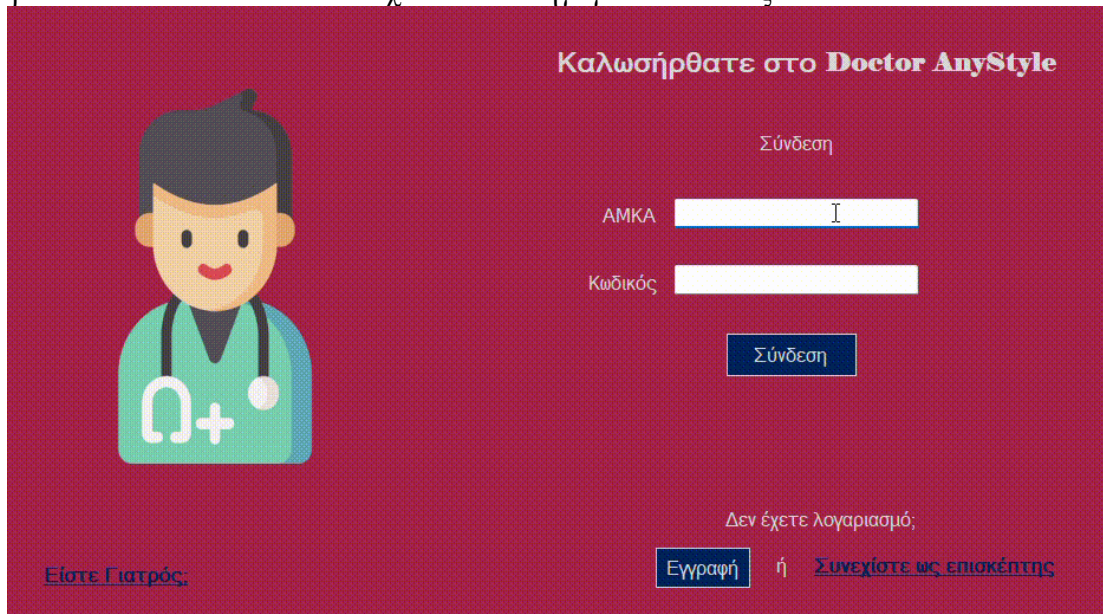
[Εγγραφή](#) ή [Συνεχίστε ως επισκέπτης](#)

[Είστε Γιατρός;](#)

Το οποίο υλοποιήθηκε με τον παρακάτω κώδικα :

```
private void button1_Click(object sender, EventArgs e)
{
    String ama = textBox1.Text.Trim();
    String password = textBox2.Text.Trim();
    if (ama != null && password != null && ama != "" && password != "")
    {
        conn.Open();
        String selectSQL = "Select * from Doctor where " + "ama=@ama and password=@password";
        SQLiteCommand cmd = new SQLiteCommand(selectSQL, conn);
        cmd.Parameters.AddWithValue("@ama", ama);
        cmd.Parameters.AddWithValue("@password", password);
        SQLiteDataReader reader = cmd.ExecuteReader();
        if (reader.Read())
        {
            Program.userId = ama;
            Program.userType = "doctor";
            MessageBox.Show("Έχετε συνδεθεί επιτυχώς!");
            new Homepage().Show();
            this.Hide();
        }
        else
        {
            MessageBox.Show("Λάθος email ή/και κωδικός πρόσβασης.");
        }
        conn.Close();
    }
    else
    {
        MessageBox.Show("Πρέπει να συμπληρώσετε όλα τα πεδία για να συνεχίσετε.");
    }
}
```

Σε περίπτωση που ο χρήστης/ασθενής βάλει λάθος στοιχεία θα λάβει ένα μήνυμα που θα τον ενημερώνει ότι κάποιο από τα στοιχεία που εισήγαγε είναι λάθος.



Καλωσήρθατε στο **Doctor AnyStyle**

Σύνδεση

ΑΜΚΑ

Κωδικός

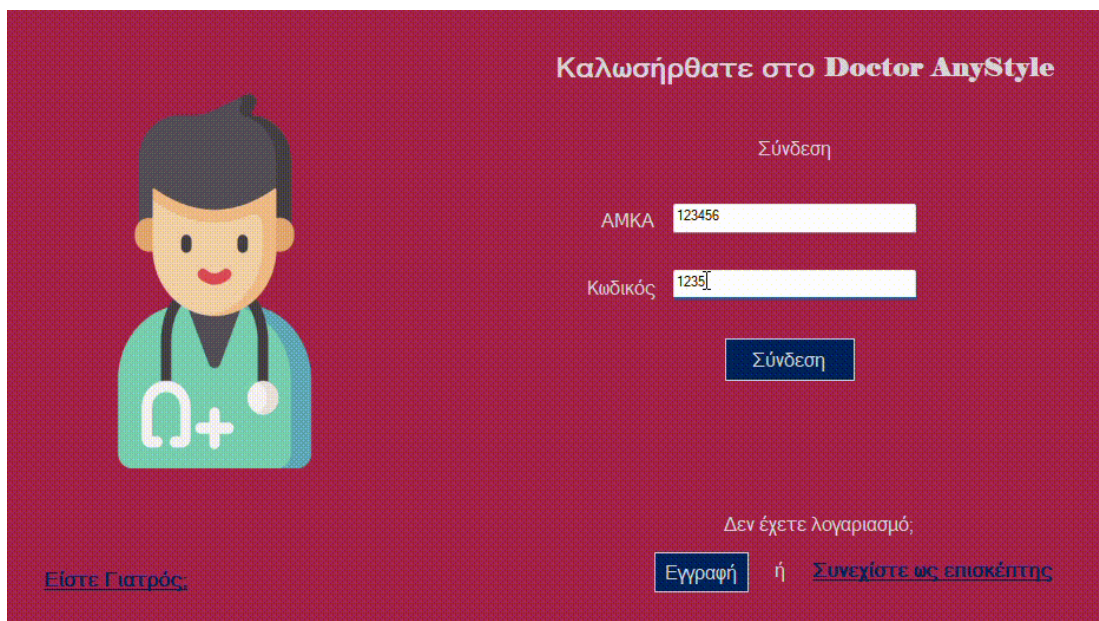
Δεν έχετε λογαριασμό;

[Είστε ΓΙΑΤΡΟΣ:](#) ή [Συνεχίστε ως ΕΠΙΣΚΕΠΤΗΣ](#)

Το οποίο υλοποιήθηκε με τον παρακάτω κώδικα :

```
MessageBox.Show("Λάθος email ή/και κωδικός πρόσβασης.");
```

Σε περίπτωση που δεν έχει εισάγει στοιχεία σε όλα τα πεδία της φόρμας θα λάβει μήνυμα λάθους που θα τον ενημερώνει ότι θα πρέπει να συμπληρώσει όλα τα πεδία για να μπορέσει να συνεχίσει.



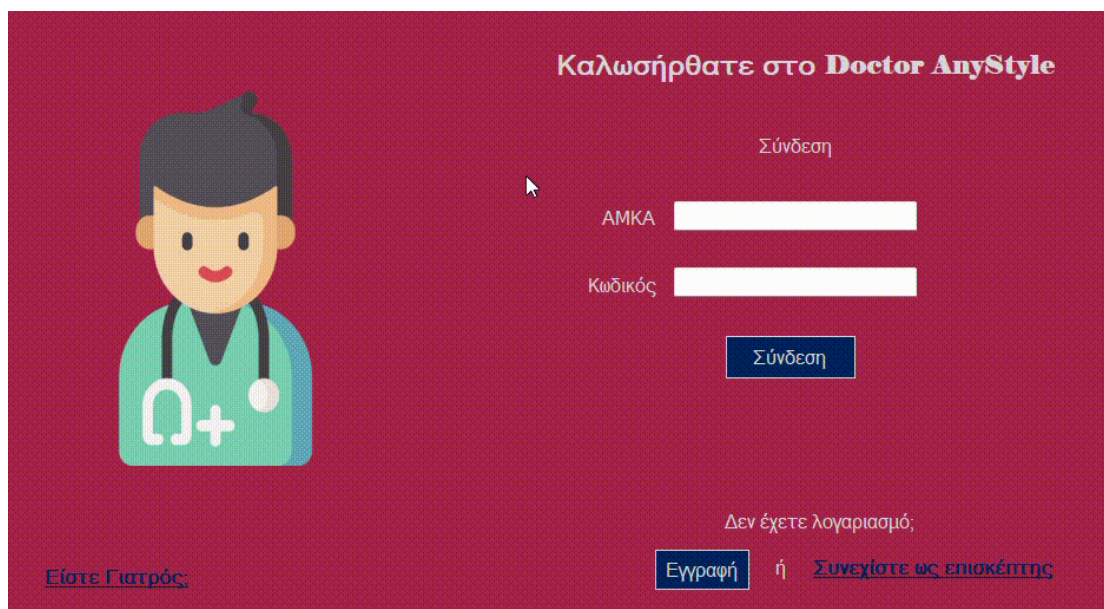
Το οποίο υλοποιήθηκε με τον παρακάτω κώδικα :

```
if (amka != null && password != null && amka != "" && password != "")  
{  
    MessageBox.Show("Πρέπει να συμπληρώσετε όλα τα πεδία για να συνεχίσετε.");  
}
```

Σε περίπτωση που ο χρήστης δεν είναι εγγεγραμμένος έχει την επιλογή να συνεχίσει σαν απλός επισκέπτης κάνοντας κλικ πάνω στο label **Συνεχίστε ως Επισκέπτης** όπου θα οδηγηθεί σε οθόνη με πληροφορίες όλων των εγγεγραμμένων ιατρών της εφαρμογής.



Σε αντίθετη περίπτωση αν επιθυμεί μπορεί να κάνει εγγραφή στην εφαρμογή πατώντας πάνω στο κουμπί **Εγγραφή** όπου και τον οδηγεί στην φόρμα συμπληρώνοντας τα απαραίτητα πεδία και στη συνέχεια μπορεί να συνεχίσει πατώντας το κουμπί **Είσοδος** όπου οδηγείται στην αρχική οθόνη που μπορεί πλέον να πραγματοποιήσει είσοδο στην εφαρμογή.



The login screen for Doctor AnyStyle features a dark red background. On the left, there is a cartoon illustration of a male doctor with a stethoscope. To the right of the illustration, the text 'Καλωσήρθατε στο Doctor AnyStyle' is displayed. Below this, the 'Σύνδεση' (Login) section contains two input fields labeled 'ΑΜΚΑ' and 'Κωδικός', followed by a blue 'Σύνδεση' button. At the bottom right, a link 'Δεν έχετε λογαριασμό;' is shown, with 'Εγγραφή' (Registration) and 'ή Συνεχίστε ως επισκέπτης' (or Continue as guest) options. At the bottom left, there is a link 'Είστε Γιατρός;' (Are you a doctor?).

Σε περίπτωση που ο χρήστης είναι γιατρός και θέλει να πραγματοποιήσει είσοδο ή και εγγραφή στην εφαρμογή θα πρέπει να κάνει κλικ στο label **Είστε Γιατρός;** Κάτω αριστερά της αρχικής οθόνης όπου και θα οδηγηθεί στην αντίστοιχη οθόνη



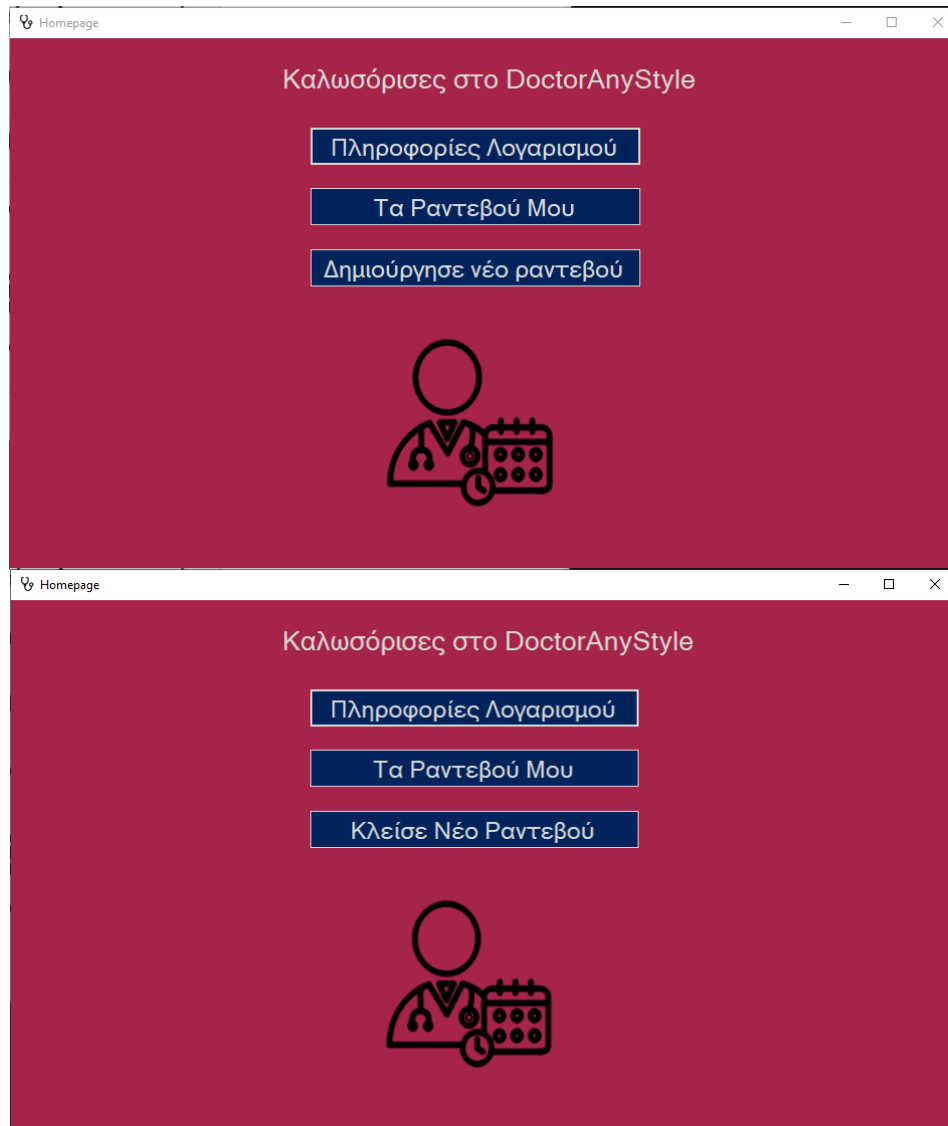
The registration screen for Doctor AnyStyle is divided into two sections by a vertical dashed line. The left section, titled 'Σύνδεση', contains input fields for 'ΑΜΑ' and 'Κωδικός', and a blue 'Σύνδεση' button. The right section, titled 'Εγγραφή', contains input fields for 'ΑΜΑ', 'Email', 'Κωδικός', 'Διεύθυνση', 'Ονοματεπώνυμο', 'Ταχ. Κώδικας', 'Ειδικότητα', 'Περιοχή', and 'Τηλέφωνο Επικοινωνίας', followed by a blue 'Εγγραφή' button. At the bottom left, there is a link 'Επιστροφή' (Return).

Στη συνέχεια μπορεί να πραγματοποιήσει είσοδο την εφαρμογή συμπληρώνοντας τα στοιχεία του στην φόρμα δεξιά Σύνδεση πατώντας το κουμπί **Σύνδεση** για να συνδεθεί στην εφαρμογή και εδώ ο χρήστης/γιατρός λαμβάνει τα ίδια μηνύματα λάθους με τον χρήστη/ασθενή σε περίπτωση λάθους εισαγωγής στοιχείων η μη συμπληρώσεις όλων των πεδίων της φόρμας, αφού συμπληρώσει τα σωστά στοιχεία και συνδεθεί με την εφαρμογή θα οδηγηθεί στην οθόνη όπου θα μπορέσει να επεξεργαστεί τα στοιχεία του λογαριασμού να δει τα ραντεβού καθώς και να δημιουργήσει νέα διαθέσιμα ραντεβού για τους ασθενείς.

Η οποία υλοποιήθηκε με τον παρακάτω κώδικα :

```
private void button1_Click(object sender, EventArgs e)
{
    String ama = textBox1.Text.Trim();
    String password = textBox2.Text.Trim();
    if (ama != null && password != null && ama != "" && password != "")
    {
        conn.Open();
        String selectSQL = "Select * from Doctor where " + "ama=@ama and password=@password";
        SQLiteCommand cmd = new SQLiteCommand(selectSQL, conn);
        cmd.Parameters.AddWithValue("@ama", ama);
        cmd.Parameters.AddWithValue("@password", password);
        SQLiteDataReader reader = cmd.ExecuteReader();
        if (reader.Read())
        {
            Program.userId = ama;
            Program.userType = "doctor";
            MessageBox.Show("Έχετε συνδεθεί επιτυχώς!");
            new Homepage().Show();
            this.Hide();
        }
        else
        {
            MessageBox.Show("Λάθος email ή/και κωδικός πρόσβασης.");
        }
        conn.Close();
    }
    else
    {
        MessageBox.Show("Πρέπει να συμπληρώσετε όλα τα πεδία για να συνεχίσετε.");
    }
}
```

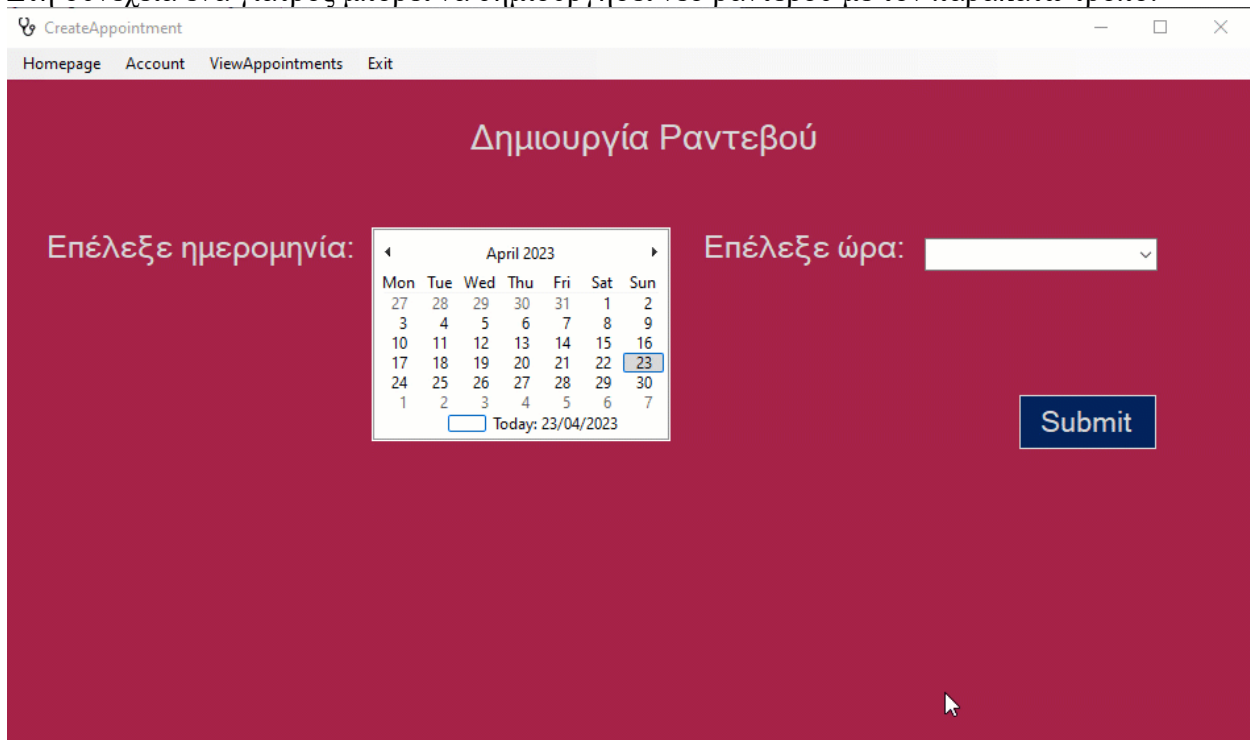
Αφού ένας γιατρός ή ένας ασθενής κάνει με επιτυχία σύνδεση στην εφαρμογή, ανακατευθύνεται στην αρχική σελίδα και αναλόγως με το είδος του user (γιατρός ή ασθενής) εμφανίζεται ως τρίτο κουμπί το «Δημιούργησε νέο ραντεβού» ή το κουμπί «Κλείσε Νέο Ραντεβού» αντίστοιχα, ενώ τα πρώτα δύο κουμπιά είναι κοινά και για τους δύο.



Ο κώδικας που επιτυγχάνει το παραπάνω αποτέλεσμα είναι ο εξής:

```
1 reference
private void Homepage_Load(object sender, EventArgs e)
{
    // Check user type & enable/disable buttons
    if (Program.userType == "doctor")
    {
        button3.Visible = false;
    }
    else if (Program.userType == "patient")
    {
        button4.Visible = false;
    }
}
```


Στη συνέχεια ένα γιατρός μπορεί να δημιουργήσει νέο ραντεβού με τον παρακάτω τρόπο:



Κατά το πάτημα του κουμπιού Submit πραγματοποιείται ένα insert query προς την βάση και συγκεκριμένα τον πίνακα appointments με τον παρακάτω τρόπο:

```
0 references
public partial class CreateAppointment : Form
{
    // Create connection string
    String connStr = "Data source=doctorAnyStyle.db;Version=3";

    // Declare and Initialize variables
    SQLiteConnection conn;
    String date;
    String doctor = Program.userId;
    int status = 1;
    int flag = 0;

    1 reference
    private void CreateAppointment_Load(object sender, EventArgs e)
    {
        // Connect with DB
        conn = new SQLiteConnection(connStr);

        conn.Close();
        if (flag == 0) {
            conn.Open();
            String insertSQL = "Insert into Appointments(doctor,date,hour,status) " +
                "values('"+ doctor + "','"+ date + "','"+ hour + "','"+ status + "')";
            SQLiteCommand cmd = new SQLiteCommand(insertSQL, conn);
            int count = cmd.ExecuteNonQuery();
            if (count > 0)
            {
                MessageBox.Show("Το νέο σου ραντεβού καταχωρήθηκε!");
                new Homepage().Show();
                this.Hide();
            }
            conn.Close();
        } else {
            MessageBox.Show("Το συγκεκριμένο ραντεβού υπάρχει ήδη.");
            flag = 0;
        }
    } else {
        MessageBox.Show("Πρέπει να συμπληρωθούν όλα τα πεδία.");
    }
}
```

Βέβαια σημαντικό εδώ είναι ότι για να γίνει το insert query, γίνονται δύο έλεγχοι:

Έλεγχος αν έχει συμπληρώσει τα δύο πεδία ο γιατρός και αν δεν τα έχει συμπληρώσει τον προτρέπει να τα συμπληρώσει:

```
1 reference
private void monthCalendar1_DateSelected(object sender, DateRangeEventArgs e)
{
    // Initialize variables
    date = monthCalendar1.SelectionRange.Start.ToString("dd/MM/yyyy");
}

1 reference
private void button1_Click(object sender, EventArgs e)
{
    // Initialize variables
    String hour = comboBox1.Text;

    // Check the contents of variables & fill comboBoxes
    if (date != null && hour != null && hour != "") {
        conn.Open();
        String selectSQL1 = "SELECT date, hour " +
            "FROM Appointments " +
            "WHERE date = " + date + " AND hour = " + hour;

        SqlCommand cmd1 = new SqlCommand(selectSQL1, conn);
        SqlDataReader reader1 = cmd1.ExecuteReader();

        while (reader1.Read())
        {
            MessageBox.Show("Υπάρχει ραντεβού για τον " + date + " στις " + hour + " ώρες.");
        }
    }
    else {
        MessageBox.Show("Πρέπει να συμπληρωθούν όλα τα πεδία.");
    }
}
```

CreateAppointment

Homepage Account ViewAppointments Exit

Δημιουργία Ραντεβού

Επέλεξε ημερομηνία:

Mon	Tue	Wed	Thu	Fri	Sat	Sun
27	28	29	30	31	1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30
1	2	3	4	5	6	7

Today: 23/04/2023

Επέλεξε ώρα:

Submit

Έλεγχος αν ο γιατρός έχει δημιουργήσει ξανά το ίδιο ραντεβού και αν το έχει δημιουργήσει ξανά τον ενημερώνει ότι το ραντεβού υπάρχει ήδη. Αυτό επιτυγχάνεται με ένα select query και ένα flag που το select query ψάχνει στον πίνακα Appointments αν υπάρχει ξανά το ίδιο ραντεβού από τον ίδιο γιατρό. Αν υπάρχει, το flag γίνεται 1 και ακολουθεί ενημερωτικό μήνυμα.

```
// Check the contents of variables & fill comboBoxes
if (date != null && hour != null && hour != "") {
    conn.Open();
    String selectSQL1 = "SELECT date, hour " +
        "FROM Appointments " +
        "WHERE doctor='" + doctor + "' " +
        "AND date = '" + date + "' " +
        "AND hour = '" + hour + "';";

    SQLiteCommand cmd1 = new SQLiteCommand(selectSQL1, conn);
    SQLiteDataReader reader1 = cmd1.ExecuteReader();
    while (reader1.Read()) {
        flag = 1;
    }
    conn.Close();
    if (flag == 0) {
        conn.Open();
        String insertSQL = "Insert into Appointments(doctor,date,hour,status) " +
            "values('" + doctor + "','" + date + "','" + hour + "','" + status + "');";
        SQLiteCommand cmd = new SQLiteCommand(insertSQL, conn);
        cmd.ExecuteNonQuery();
        conn.Close();
    }
}
```

```
    } else {
        MessageBox.Show("Το συγκεκριμένο ραντεβού υπάρχει ήδη.");
        flag = 0;
    }
}
```

CreateAppointment

Homepage Account ViewAppointments Exit

Δημιουργία Ραντεβού

Επέλεξε ημερομηνία:

Mon	Tue	Wed	Thu	Fri	Sat	Sun
27	28	29	30	31	1	2
3	4	5	6	7	8	9
10	11					
17	18					
24	25					
1	2					

Επέλεξε ώρα: 09:00 - 10:00

Submit

Το συγκεκριμένο ραντεβού υπάρχει ήδη.

OK

Αντίστοιχα, ένας ασθενής μπορεί να κλείσει ένα ραντεβού. Για να το κάνει αυτό θα πρέπει να εκτελέσει με τη σειρά τα παρακάτω βήματα. Σε κάθε βήμα όπως φαίνεται παρακάτω κλειδώνουν και ξεκλειδώνουν πεδία, ώστε να εκτελέσει ο ασθενής τα βήματα διαδοχικά και να τον οδηγήσουν σε κλείσιμο ραντεβού χωρίς λάθη:

BookAppointment

Homepage Account ViewAppointments Exit

Κλείσε Ραντεβού

Επέλεξε ειδικότητα: Επέλεξε περιοχή:

Επέλεξε γιατρό:

Επέλεξε ημέρα και ώρα:

Αιτία επίσκεψης:

Σε περίπτωση που ο ασθενής δεν συμπληρώσει κάποιο πεδίο και προσπαθήσει να προχωρήσει πατώντας submit στο επόμενο βήμα, τον ενημερώνει το σύστημα ότι πρέπει να συμπληρώσει τα πεδία:

BookAppointment

Homepage Account ViewAppointments Exit

Κλείσε Ραντεβού

Επέλεξε ειδικότητα: Παθολόγος Επέλεξε περιοχή: Καλλιθέα

Επέλεξε γιατρό:

Επέλεξε ημέρα και ώρα:

Αιτία επίσκεψης:

Συμπληρώστε όλα τα πεδία.

OK

Ο κώδικας της συγκεκριμένης φόρμας έχει αρκετά σημαντικά στοιχεία. Αρχικά τα dropdown πεδία γεμίζουν με τους κατάλληλους ελέγχους μόνο με στοιχεία από γιατρούς που έχουν διαθέσιμα ραντεβού και μόνο με τα διαθέσιμα αυτά ραντεβού. Ακολουθεί παράδειγμα:

```
// Connect with DB & run queries
conn = new SQLiteConnection(connStr);
conn.Open();
String selectSQL1 = "SELECT DISTINCT Doctor.specialty " +
    "FROM Doctor " +
    "JOIN Appointments " +
    "ON Doctor.ama = Appointments.doctor " +
    "WHERE Appointments.status = 1 ;";

SQLiteCommand cmd1 = new SQLiteCommand(selectSQL1, conn);
SQLiteDataReader reader1 = cmd1.ExecuteReader();
while (reader1.Read())
{
    comboBox1.Items.Add(reader1.GetString(0));
}

String selectSQL2 = "SELECT DISTINCT Doctor.region " +
    "FROM Doctor " +
    "JOIN Appointments " +
    "ON Doctor.ama = Appointments.doctor " +
    "WHERE Appointments.status = 1 ;";

SQLiteCommand cmd2 = new SQLiteCommand(selectSQL2, conn);
SQLiteDataReader reader2 = cmd2.ExecuteReader();
while (reader2.Read())
{
    comboBox2.Items.Add(reader2.GetString(0));
}

conn.Close();
```

Επίσης, σε κάθε βήμα ενεργοποιεί και απενεργοποιεί panel με πεδία, ώστε ο ασθενής να κλείσει βήμα βήμα το ραντεβού τους σωστά:

```
// Enable/Disable panels
panel2.Enabled = false;
panel3.Enabled = true;

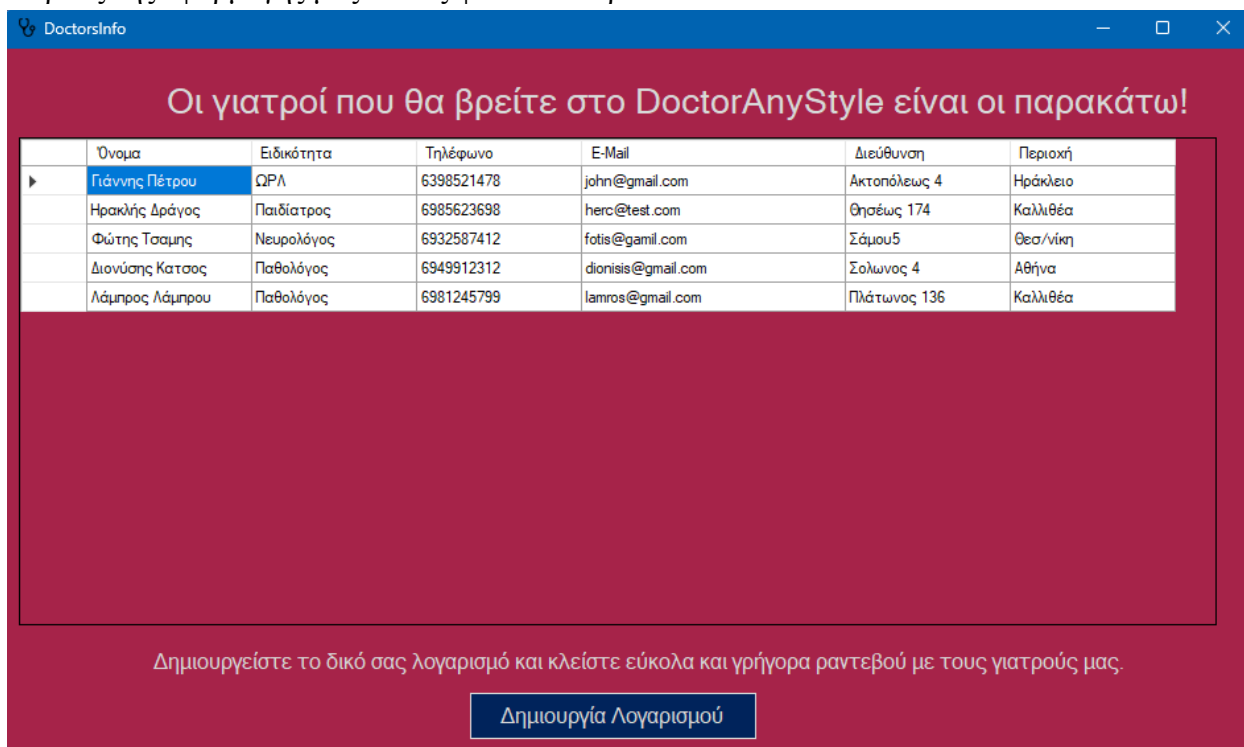
// Enable/Disable panels
panel3.Enabled = false;
panel4.Enabled = true;
```

Τέλος σε κάθε βήμα γίνονται έλεγχοι για να ελέγξουμε αν ο χρήστης έχει συμπληρώσει τα πεδία σωστά και δεν έχει αφήσει και κενά στην αρχή και στο τέλος.

```
private void button3_Click(object sender, EventArgs e)
{
    // Initialize variables
    date = comboBox4.Text.Trim();
    hour = comboBox5.Text.Trim();

    // Check the contents of variables & run query
    if (date != null && date != "" && hour != null && hour != "")
    {
        // Run query
    }
    else
    {
        MessageBox.Show("Συμπληρώστε όλα τα πεδία.");
    }
}
```

Αν στην αρχική οθόνη κατά την έναρξη της εφαρμογής ο χρήστης συνεχίσει σαν **επισκέπτης** τότε θα έχει τη δυνατότητα να δει έναν συγκεντρωτικό κατάλογο με όλους τους συμβεβλημένους ιατρούς της εφαρμογής μας. Όπως φαίνεται παρακάτω:



Το αποτέλεσμα της φόρμας επιτυγχάνεται με το παρακάτω χωρίο κώδικα με ουσιαστικότερο κομμάτι, την κλήση στην βάση μας,

SELECT * FROM Doctor ;

```
1 reference
private void DoctorsInfo_Load(object sender, EventArgs e)
{
    conn = new SQLiteConnection(connStr);
    conn.Open();
    String selectSQL1 = "SELECT *" +
        "FROM Doctor;";

    SQLiteCommand cmd1 = new SQLiteCommand(selectSQL1, conn);
    SQLiteDataReader reader1 = cmd1.ExecuteReader();
    while (reader1.Read())
    {
        dataGridView1.Rows.Add(new object[]
        {
            reader1.GetValue(reader1.GetOrdinal("name")),
            reader1.GetValue(reader1.GetOrdinal("specialty")),
            reader1.GetValue(reader1.GetOrdinal("tel")),
            reader1.GetValue(reader1.GetOrdinal("email")),
            reader1.GetValue(reader1.GetOrdinal("address")),
            reader1.GetValue(reader1.GetOrdinal("region"))
        });
    }
    conn.Close();
}
```

που φέρνει όλον τον πίνακα **Doctor** και έπειτα **επιλεκτικά** πληθαίνουμε το dataGridView1 παίρνοντας τις στήλες που θέλουμε με την εντολή

```
reader1.GetValue(reader1.GetOrdinal("#onoma_stilis_pinaka#"))
```

Σε αυτήν την φόρμα ο **εγγεγραμμένος** χρήστης έχει τη δυνατότητα, να **επεξεργαστεί** τα στοιχεία του λογαριασμού του. Αυτά, που αντιστοίχως ζητήθηκαν από τον καθένα στην εγγραφή του, αναλόγως αν ήταν ιατρός ή ασθενής. Έτσι λοιπόν πολύ απλά, στα εδάφια που φαίνονται σβήνει και ξαναγράφει τα στοιχεία του όπως επιθυμεί και έπειτα πατά το κουμπί **Save** χαμηλά της φόρμας, για να σωθούν οι αλλαγές που έγραψε. Η ανάδραση της διαδικασίας είναι ένα παράθυρο διαλόγου που ενεργοποιείται για την επιτυχή καταχώρηση ή την έλλειψη στοιχείων.

Περίπτωση ασθενούς χρήστη:

The screenshot shows a web application window titled "AccountInfo". The menu bar includes "Homepage", "ViewAppointments", "BookAppointment", and "Exit". The main content area has a dark red background and is titled "Στοιχεία Λογαριασμού" (Account Details). Below the title, there is a instruction in Greek: "Αν θες να αλλάξεις κάποιο στοιχείο, πραγματοποιήσε την αλλαγή στο αντίστοιχο πεδίο και πάτα το κουμπί Save." (If you want to change any element, perform the change in the corresponding field and click the Save button). The form contains five input fields with labels and values: "Ονοματεπώνυμο:" (Name) with "Ανδρέας Ανδρεαδάκης", "ΑΜΚΑ:" (AMKA) with "24109500888", "Password:" with "@#!dsa@#!es", "Τηλέφωνο:" (Phone) with "6985623698", and "E-Mail:" with "andresadread@outlook.com". At the bottom center, there is a blue button with a floppy disk icon and the text "Save".

Περίπτωση ιατρού χρήστη:

The screenshot shows a web application window titled "AccountInfo". The menu bar includes "Homepage", "ViewAppointments", "CreateAppointment", and "Exit". The main content area has a red background and is titled "Στοιχεία Λογαριασμού". Below the title, there is a message in Greek: "Αν θες να αλλάξεις κάποιο στοιχείο, πραγματοποιήσε την αλλαγή στο αντίστοιχο πεδίο και πάτα το κουμπί Save." The form contains several input fields with labels and values: "Ονοματεπώνυμο:" with value "Λάμπρος Λάμπρου", "AMA:" with value "124578963", "Password:" with value "@#!dsa@#!es", "Τηλέφωνο:" with value "6981245789", "E-Mail:" with value "lamros@gmail.com", "Διεύθυνση:" with value "Πλάτωνος 134", "T.K:" with value "17674", "Περιοχή:" with value "Καλλιθέα", and "Εδικότητα:" with value "Παθολόγος". At the bottom right of the form is a blue "Save" button.

Ο κώδικας που εκτελείται κατά το πάτημα του Save είναι ο παρακάτω και ουσιαστικά είναι ένας έλεγχος για κενά εδάφια και ένα **UPDATE** στην βάση μας με τα ανανεωμένα στοιχεία πλέον.

Κώδικας ιατρού :

```
if (Program.userType == "doctor" && d_name != null && d_ama != null && d_password != null && d_tel != null && d_email  
d_address != null && d_zip != null && d_region != null && d_specialty != null && d_name != "" && d_ama != "" && d_pass  
d_address != "" && d_zip != "" && d_region != "" && d_specialty != "")  
{  
    save.CommandText = "UPDATE Doctor " +  
        "SET name = :named " +  
        ", ama = :ama " +  
        ", password = :pass " +  
        ", tel = :tele " +  
        ", email = :mail " +  
        ", address = :address " +  
        ", zip = :zip " +  
        ", region = :region " +  
        ", specialty = :specialty " +  
        "WHERE ama = :userid";  
    save.Parameters.Add("named", DbType.String).Value = d_name;  
    save.Parameters.Add("ama", DbType.String).Value = d_ama;  
    save.Parameters.Add("pass", DbType.String).Value = d_password;  
    save.Parameters.Add("tele", DbType.String).Value = d_tel;  
    save.Parameters.Add("mail", DbType.String).Value = d_email;  
    save.Parameters.Add("address", DbType.String).Value = d_address;  
    save.Parameters.Add("zip", DbType.VarNumeric).Value = d_zip;  
    save.Parameters.Add("region", DbType.String).Value = d_region;  
    save.Parameters.Add("specialty", DbType.String).Value = d_specialty;  
    save.Parameters.Add("userid", DbType.String).Value = Program.userId;  
    save.ExecuteNonQuery();  
    MessageBox.Show("Τα στοιχεία σου ανανεώθηκαν.");  
}
```


Κώδικας ασθενούς :

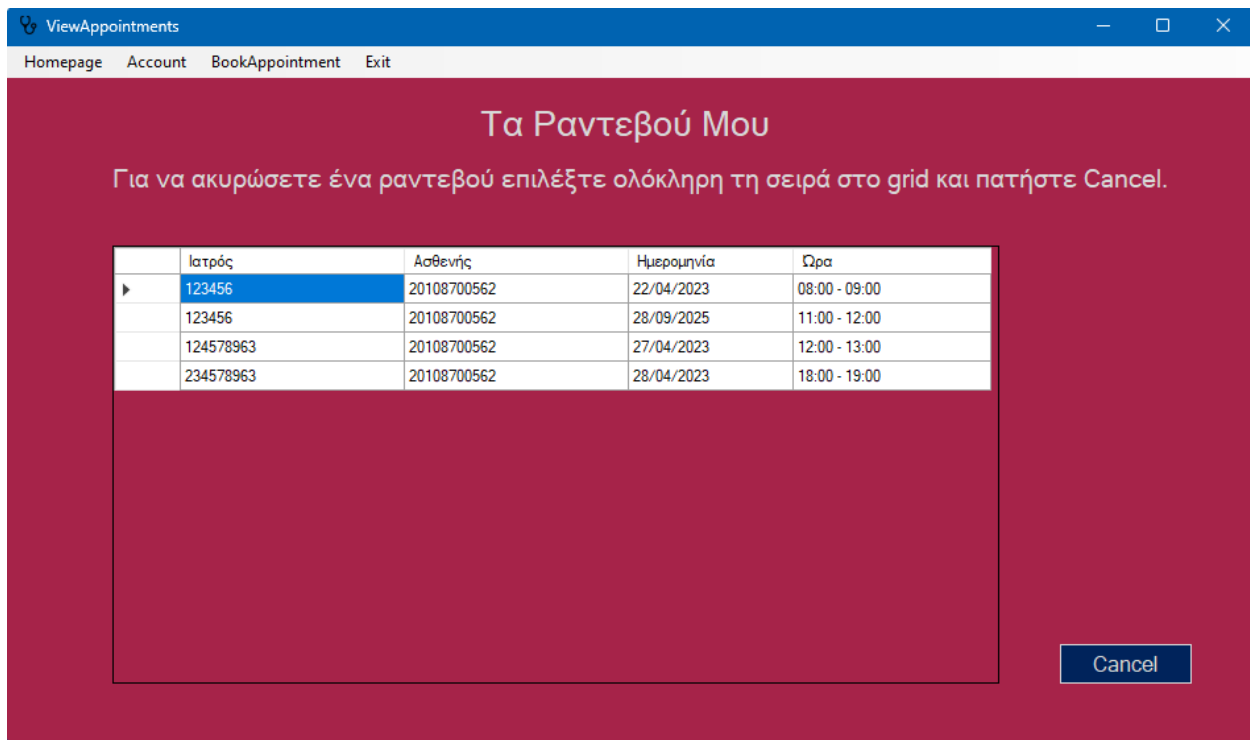
```
else if (Program.userType == "patient" && p_name != null && p_tel != null && p_amka != null && p_pass != null && p_email != null &&
p_name != "" && p_tel != "" && p_amka != "" && p_pass != "" && p_email != "")
{
    save.CommandText = "UPDATE Patient " +
        "SET name = :named " +
        ", amka = :amka " +
        ", password = :pass " +
        ", tel = :tele " +
        ", email = :mail " +
        "WHERE amka = :userid";
    save.Parameters.Add("named", DbType.String).Value = p_name;
    save.Parameters.Add("amka", DbType.String).Value = p_amka;
    save.Parameters.Add("pass", DbType.String).Value = p_pass;
    save.Parameters.Add("tele", DbType.String).Value = p_tel;
    save.Parameters.Add("mail", DbType.String).Value = p_email;
    save.Parameters.Add("userid", DbType.String).Value = Program.userId;
    save.ExecuteNonQuery();
    MessageBox.Show("Τα στοιχεία σου ανανεώθηκαν.");
}
else
{
    MessageBox.Show("Παρακαλώ συμπληρώστε όλα τα πεδία.");
}
conn.Close();
new Homepage().Show();
this.Hide();
```

Στην περίπτωση αποτυχίας ανανέωσης στοιχείων, λόγω κενού εδαφίου, σε οποιαδήποτε περίπτωση χρήστη, η γραμμή που δημιουργεί την ανάδραση του διαλόγου με το μήνυμα είναι η παρακάτω:

`MessageBox.Show("Παρακαλώ συμπληρώστε όλα τα πεδία.");`

Σε αυτή τη φόρμα ο εγγεγραμμένος χρήστης, βλέπει τα ραντεβού που έχει κλείσει. Αν είναι ασθενής τότε, τότε του δίνεται η δυνατότητα ακύρωσης ενός ραντεβού επιλέγοντας το και πατώντας το κουμπί **Cancel** κάτω δεξιά της οθόνης. Αντίστοιχα αν είναι ιατρός, μπορεί να διαγράψει το ραντεβού είτε σαν διαθεσιμότητα γενικά απ' τις διαθέσιμες ώρες του, είτε σαν ακύρωση κανονισμένου με ασθενή ραντεβού πατώντας το κουμπί **Delete**.

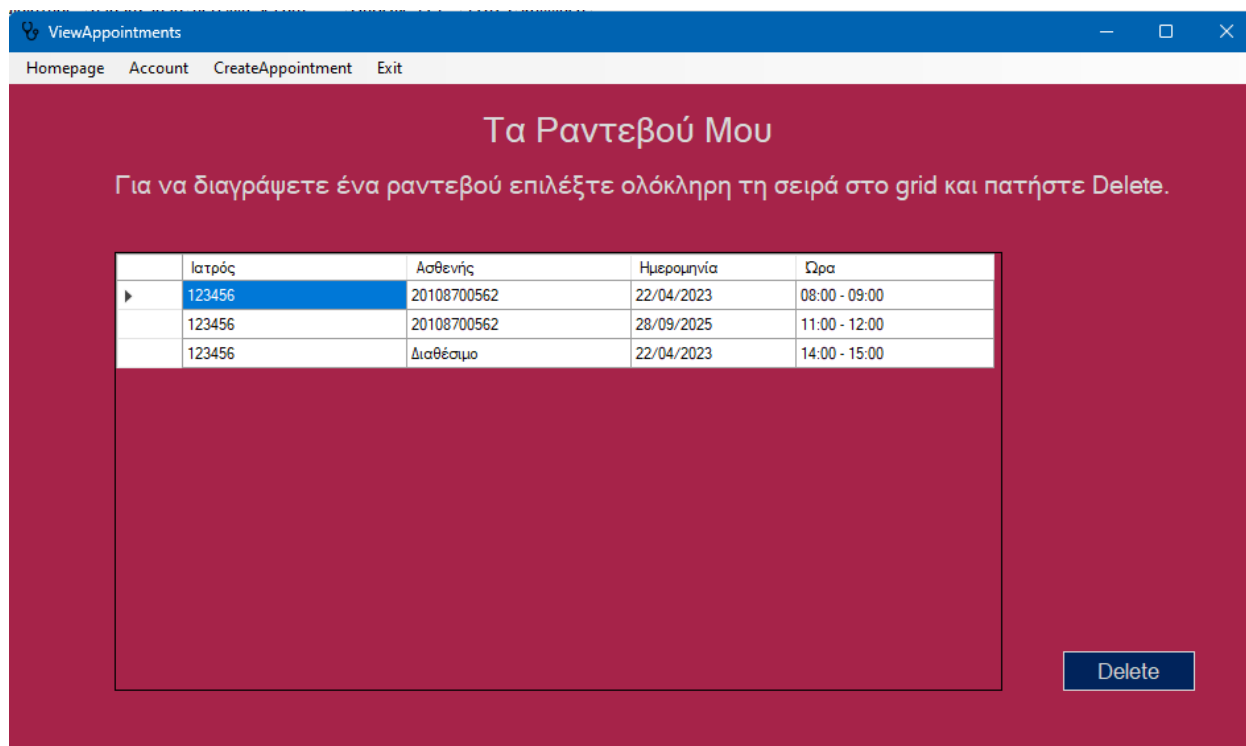
Περίπτωση ασθενούς χρήστη με τον κώδικα της, να φαίνεται παρακάτω:



```
else
{
    String selectSQL1 = "SELECT * " +
        "FROM Appointments " +
        "WHERE patient = '" + Program.userId + "' ";

    SQLiteCommand cmd1 = new SQLiteCommand(selectSQL1, conn);
    SQLiteDataReader reader1 = cmd1.ExecuteReader();
    while (reader1.Read())
    {
        dataGridView1.Rows.Add(new object[]
        {
            reader1.GetValue(reader1.GetOrdinal("ID")),
            reader1.GetValue(reader1.GetOrdinal("doctor")),
            reader1.GetValue(reader1.GetOrdinal("patient")),
            reader1.GetValue(reader1.GetOrdinal("date")),
            reader1.GetValue(reader1.GetOrdinal("hour"))
        });
    }
    button1.Text = "Cancel";
    createAppointmentToolStripMenuItem.Visible = false;
    label3.Visible = false;
}
conn.Close();
```

Περίπτωση ιατρού χρήστη με τον κώδικα της να φαίνεται παρακάτω:



```
conn = new SQLiteConnection(connStr);
conn.Open();
if (Program.userId == "doctor")
{
    String selectSQL1 = "SELECT * " +
        "FROM Appointments " +
        "WHERE doctor = '" + Program.userId + "' ";

    SQLiteCommand cmd1 = new SQLiteCommand(selectSQL1, conn);
    SQLiteDataReader reader1 = cmd1.ExecuteReader();
    while (reader1.Read())
    {
        dataGridView1.Rows.Add(new object[]
        {
            reader1.GetValue(reader1.GetOrdinal("ID")),
            reader1.GetValue(reader1.GetOrdinal("doctor")),
            reader1.GetValue(reader1.GetOrdinal("patient")),
            reader1.GetValue(reader1.GetOrdinal("date")),
            reader1.GetValue(reader1.GetOrdinal("hour"))
        });
    }
    dataGridView1.Columns["Column3"].DefaultCellStyle.NullValue = "Διαθέσιμο";
    bookAppointmentToolStripMenuItem.Visible = false;
    label2.Visible = false;
}
```

Τέλος παρακάτω είναι ο κώδικας που εκτελείται για την διαγραφή των ραντεβού σε κάθε περίπτωση. Αξίζει να σημειωθεί ότι στην περίπτωση του ιατρού, έχουμε κανονική διαγραφή της εγγραφής του ραντεβού στον πίνακα **Appointments** στη βάση μας, βάσει του ID της που το επιλέγουμε από το αντικείμενο dataGridView1 με το εξής:

```
"DELETE FROM Appointments " +  
"WHERE ID = '" + dataGridView1.CurrentRow.Cells[0].Value + "'";
```

Ενώ στην περίπτωση του ασθενούς, κάνουμε απλά μια **εκκαθάριση** των τιμών στον πίνακα **Appointments**, αφήνοντας έτσι το ραντεβού διαθέσιμο ξανά.

```
"UPDATE Appointments " +  
"SET patient = '" + null + "' " +  
", symptoms = '" + null + "' " +  
", status = 1 " +  
"WHERE ID = '" + dataGridView1.CurrentRow.Cells[0].Value + "'";
```

```
private void button1_Click(object sender, EventArgs e)  
{  
    if (Program.userType == "doctor")  
    {  
        conn.Open();  
        String deleteSQL = "DELETE FROM Appointments " +  
                            "WHERE ID = '" + dataGridView1.CurrentRow.Cells[0].Value + "'";  
        SQLiteCommand cmd = new SQLiteCommand(deleteSQL, conn);  
        int count = cmd.ExecuteNonQuery();  
        if (count > 0)  
        {  
            MessageBox.Show("Το ραντεβού σου διαγράφηκε.");  
            new Homepage().Show();  
            this.Hide();  
        }  
        conn.Close();  
    }  
    else  
    {  
        conn.Open();  
        String updateSQL = "UPDATE Appointments " +  
                            "SET patient = '" + null + "' " +  
                            ", symptoms = '" + null + "' " +  
                            ", status = 1 " +  
                            "WHERE ID = '" + dataGridView1.CurrentRow.Cells[0].Value + "'";  
        SQLiteCommand cmd = new SQLiteCommand(updateSQL, conn);  
        int count = cmd.ExecuteNonQuery();  
        if (count > 0)  
        {  
            MessageBox.Show("Το ραντεβού σου ακυρώθηκε.");  
            new Homepage().Show();  
            this.Hide();  
        }  
        conn.Close();  
    }  
}
```