

Δυναμική Παραγωγή Ερωτηματολογίων

Dynamic generation of PROMS

Ξυλούρης Ιωάννης CSD4225



Τμήμα Επιστήμης Υπολογιστών Πανεπιστήμιο Κρήτης

Χρονικό διάστημα: 12/2022 - 11/2023

Επιβλέπων : Χαρίδimos Κονδυλάκης

Επόπτης:

Μέλος Επιτροπής Παρακολούθησης/Αξιολόγησης:

Περιεχόμενα

1. Εισαγωγή	3
1.1 Σκοπός Συστήματος	3
1.2 Γενική περιγραφή	4
2. Γενικό πλαίσιο- PROMs (Patient-reported outcome measures).....	4
3. Σενάρια Χρήσης	5
4. Απαιτήσεις	5
5. Αρχιτεκτονική Συστήματος και Εργαλεία	6
5.1 Αρχιτεκτονική Βάσης Δεδομένων.....	6
5.1.1 Μοντέλο Οντοτήτων - Συναρτήσεων.....	7
5.2 Αρχιτεκτονική Back-End.....	10
5.3 Αρχιτεκτονική Front-End	18
6. Οθόνες της εφαρμογής -Οδηγίες Χρήσεως.....	18
6.1 Οθόνη και οδηγίες χρήσης του Backend.....	18
6.2 Οθόνες και οδηγίες χρήσης εφαρμογής	19
7.Οδηγίες εγκατάστασης του συστήματος	22
8. Επίλογος.....	25
8.1 Σύνοψη.....	25
8.2 Αδυναμίες και Μελλοντική εργασία	25
9. Πηγές.....	27

1. Εισαγωγή

Σκοπός του εγγράφου είναι να παρουσιάσει την ανάπτυξη ενός Web συστήματος που προσομοιώνει την ικανότητα απάντησης ερωτηματολογίων με στόχο την πρόωπη εξέταση και εκτίμηση της υγείας του χρήστη. Το σύστημα απαρτίζεται από ερωτηματολόγια τύπου Proms (Patient-Reported Outcome Measures) και στόχος του είναι κατά την λήξη των δυναμικών αυτών ερωτηματολογίων ο ασθενής να ανανεώσει την εικόνα της κατάστασης της υγείας του. Τα ερωτηματολόγια αυτά αποκτούν διάφορα αποτελέσματα που αξιολογούν την τρέχουσα κατάσταση του ασθενούς. Φυσικά, κρατάται ιστορικό για το κάθε ερωτηματολόγιο που απήντησε ο ίδιος και έτσι μπορεί να εκτιμηθεί και η πορεία της υγείας του.

Κατά την ολοκλήρωση του εγγράφου θα υπάρξει μια ολοκληρωμένη εικόνα για το τι είναι τα PROMs και πως το αναπτυσσόμενο σύστημα χρησιμοποιεί τα ερωτηματολόγια αυτά, τα οποία βοηθούν στην παρακολούθηση της υγείας του ανθρώπου. Η παρούσα ανάπτυξη γίνεται στα πλαίσια της διπλωματικής εργασίας του Τμήματος Επιστήμης Υπολογιστών του Πανεπιστημίου Κρήτης.

1.1 Σκοπός Συστήματος

Ο σκοπός του παρόντος συστήματος, είναι να μπορεί ο χρήστης να απαντήσει σε ένα ερωτηματολόγιο που του παρέχεται και μέσα από απλές αλλά στοχευμένες ερωτήσεις να υπάρξει μια ενημέρωση ως προς την κατάσταση της υγείας του αλλά και σε πολλές περιπτώσεις μια αρχική διάγνωση κάποιου ενδεχόμενου προβλήματος που πιθανόν να αντιμετωπίζει. Αρχικά, ο χρήστης θα επιλέξει το ερωτηματολόγιο που επιθυμεί και έπειτα θα μεταβεί στην συμπλήρωση του. Κατά την ολοκλήρωση του ερωτηματολογίου ο χρήστης θα μπορεί να δει το ιστορικό των απαντήσεων στο ερωτηματολόγιο, καθώς και τον υπολογισμό αυτοματοποιημένων βαθμολογιών που υποστηρίζονται σε κάθε ένα από αυτά. Η βαθμολογία αυτή προσφέρει στο χρήστη μία πρώτη εικόνα για την πορεία του, ενώ ταυτόχρονα αποτυπώνει την αντίληψη του ασθενούς για την κατάσταση του στον γιατρό του.

1.2 Γενική περιγραφή

Στην παρούσα πτυχιακή εργασία υλοποιήθηκαν τα παρακάτω:

- Η υποστήριξη ποικιλίας γλωσσών τόσο στις ερωτήσεις που απαντά ο χρήστης όσο και στις απαντήσεις που λαμβάνει
- Ο υπολογισμός της βαθμολογίας (score) του χρήστη ανάλογα τις απαντήσεις του
- Η υποστήριξη διαφορετικής ακολουθίας ερωτήσεων ανάλογα τις απαντήσεις σε προηγούμενες ερωτήσεις
- Η υποστήριξη τεσσάρων ειδών ερωτήσεων που απαντά ο χρήστης
 - Ερωτήσεις πολλαπλής επιλογής
 - Free text απαντήσεων όπου ο χρήστης εισάγει κείμενο σαν απάντηση
 - Ανέβασμα κάποιας εικόνας (πχ αν η ερώτηση είναι ανεβάστε φωτογραφία της πληγής)
 - Checkbox

Για την ανάπτυξη της βάσης αυτής και για την δημιουργία των ερωτηματολογίων χρησιμοποιήθηκε η PostgreSQL της οποίας ο ρόλος θα αναλυθεί εκτενέστερα σε επόμενη παράγραφο.

2. Γενικό πλαίσιο- PROMs (Patient-reported outcome measures)

Τα ερωτηματολόγια τα οποία δημιουργήθηκαν είχαν την μορφή PROMs (Patient-reported outcome measures) δηλαδή, έκβαση που προκύπτει με βάση τις αναφορές του ασθενή. Χρησιμοποιούνται κατά κύριο λόγο για την αξιολόγηση της υγείας του ασθενούς, κατά την διάρκεια της ασθένειας ή κατά την περίοδο της θεραπείας. Επιπλέον, αυτού του τύπου τα ερωτηματολόγια αναγνωρίζονται λόγω των πολύτιμων και αναγκαίων πληροφοριών για την επίτευξη των στόχων υγείας αλλά και λόγω της προσέγγισης από πλευράς του ασθενούς. Μπορούν να χρησιμοποιηθούν ως επιπρόσθετη πηγή πληροφοριών, συμπληρωματικά με το ιστορικό του ασθενούς που λαμβάνεται από κλινικές καθώς οι ασθενείς δίνουν πληροφορίες για διάφορες πτυχές της υγείας τους. Πιο συγκεκριμένα, δίνουν πληροφορίες, σχετικές με την ποιότητα της ζωής τους, όπως συμπτώματα, θέματα που αφορούν την λειτουργικότητα τους αλλά και θέματα που αφορούν την φυσική, ψυχική ακόμη και την κοινωνική τους υγεία,

δηλαδή την αλληλεπίδραση και την ικανότητα τους να σχηματίζουν σχέσεις με τους γύρω τους. Τέλος, είναι σημαντικό να υπάρχουν τα PROMs καθώς βοηθούν στην κατανόηση του κατά πόσο επιδρούν και βοηθούν τους ασθενείς οι θεραπείες και το σύστημα υγείας στην βελτίωση της κατάστασης της υγείας τους και στην ποιότητα της ζωής τους.

3. Σενάρια Χρήσης

Τα συγκεκριμένα ερωτηματολόγια, έχουν την δυνατότητα να συμπληρώνονται με τέσσερις διαφορετικούς τρόπους μέσω ερωτήσεων πολλαπλής επιλογής, συμπλήρωσης κειμένου, ανέβασμα αρχείου όπως φωτογραφίας αλλά και με την χρήση checkbox. Συμπληρώνοντας το ερωτηματολόγιο υπάρχει η δυνατότητα:

- Ελέγχου της βαθμολογίας του κάθε ασθενή τόσο από εκείνον όσο και από το γιατρό του ώστε και ο ίδιος να αντιλαμβάνεται την πορεία της υγείας του
- Τον έλεγχο της πορείας της υγείας κάθε ασθενούς μέσω των απαντήσεων που δίνει στις εξατομικευμένες ερωτήσεις του προγράμματος
- Σύγκριση των αποτελεσμάτων με βάση το ιστορικό το οποίο υπάρχει προσβάσιμο για κάθε χρήστη

Ο χρήστης με την είσοδο του, απαντά μία σειρά διαδοχικών ερωτήσεων ενώ στο τέλος του δίνεται η δυνατότητα ελέγχου της βαθμολογίας του και των απαντήσεων που έχει δώσει συνολικά.

4. Απαιτήσεις

Κάθε σύστημα οφείλει να ακολουθεί κάποιους κανόνες ώστε να είναι εύχρηστο και κατανοητό σε καινούργιους developers αλλά και για να διατηρήσει την επεκτασιμότητα του. Πρωταρχικός στόχος στην διατήρησης αυτής της κατάστασης, είναι η ασφάλεια από κάποιον εξωγενή παράγοντα. Σημαντική βοήθεια στην εξάλειψη μια τέτοιας κατάστασης παρέχεται απο την .NET με το “ValidateAntiForgeryToken” δημιουργώντας ένα cookie που περνάει μαζί με τα δεδομένα της φόρμας που συμπληρώνεται και ελέγχεται στο Backend. Αυτό αποτρέπει ένα ενεργό cookie που ορίζει την σύνδεση του χρήστη να χρησιμοποιηθεί από κάποια άλλη ιστοσελίδα σαν “authenticated user” και να περάσει κακόβουλο κώδικα στην βάση δεδομένων. Εξίσου σημαντική είναι η διατήρηση της διαθεσιμότητας. Για να επιτευχθεί κάτι τέτοιο, είναι απαραίτητο ο developer να κατανοεί πλήρως την δομή του συστήματος και να το εγκαταστήσει στο δικό του υπολογιστή ή διακοσμητή. Έτσι, πέρα απο την διαθεσιμότητα μπορεί να ξεκινήσει και η επέκταση του. Για τον λόγο αυτό, απαιτείται όλα τα δεδομένα που ζητούνται από τους χρήστες να είναι σωστά.

Το πρόγραμμα είναι υλοποιημένο στην γλώσσα C# με το ASP .NET CORE WebApp Framework, το ίδιο Framework είναι υπεύθυνο για την ανανέωση των δεδομένων στο Frontend μέσω Razor Pages, ενώ για την βάση δεδομένων έχει χρησιμοποιηθεί η PostgreSQL. Η κώδικας βρίσκεται στο Github ([link](#)).

5. Αρχιτεκτονική Συστήματος και Εργαλεία

Πρωτού αναλύσουμε την αρχιτεκτονική του συστήματος που υλοποιήθηκε αξίζει να σημειωθούν, οι τεχνολογίες που έχουν χρησιμοποιηθεί καθώς και ο λόγος για τον οποίο χρησιμοποιήθηκαν.

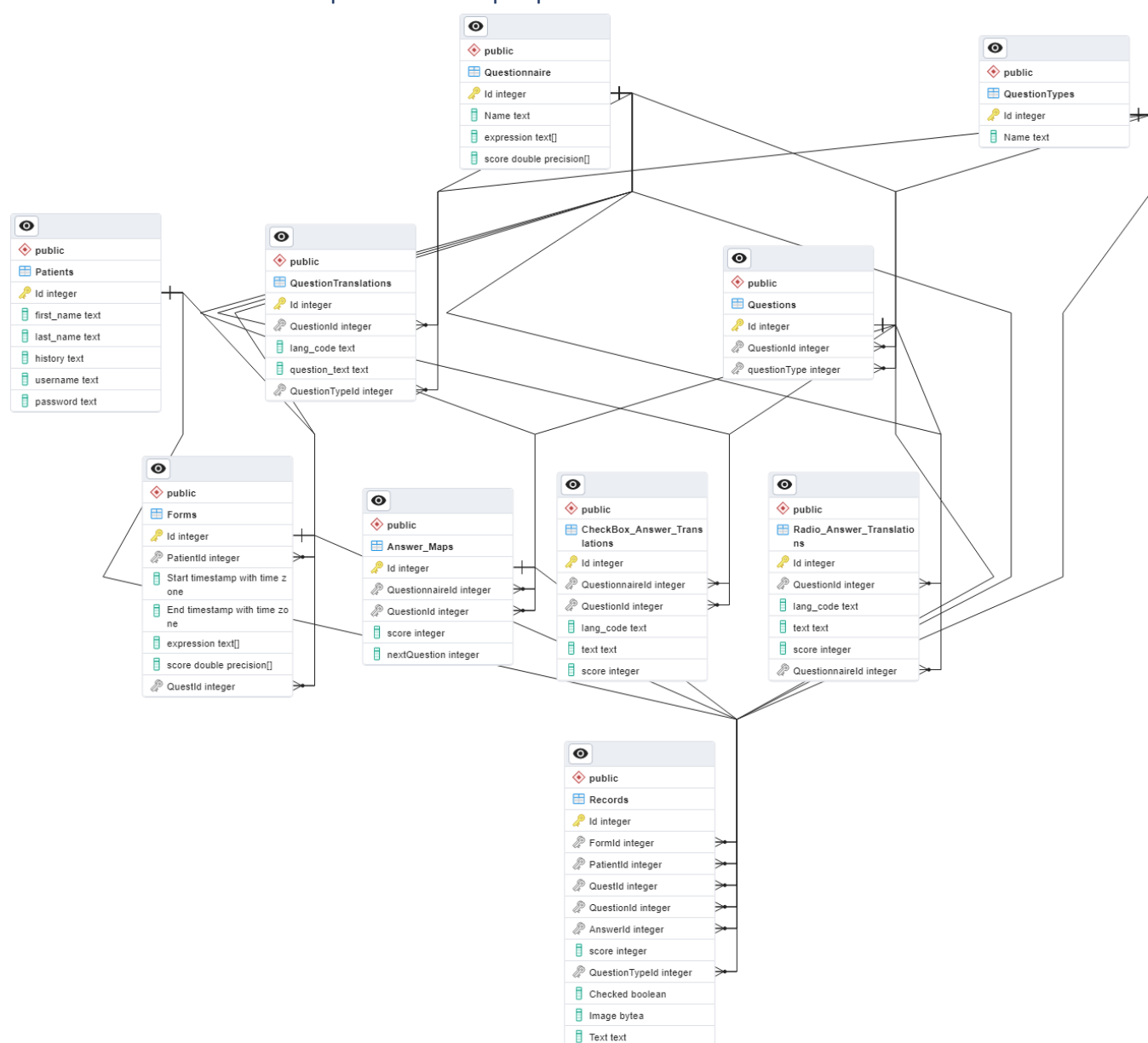
Όσον αφορά το κομμάτι του Backend, που αποτελούσε προτεύον ρόλο στην σχεδίαση του Web ιστότοπου, για τη δημιουργία του έγινε η χρήση Framework ASP .NET CORE 6.0 και η βάση δεδομένων βρίσκεται στην PostgreSQL.

5.1 Αρχιτεκτονική Βάσης Δεδομένων

Η βάση δεδομένων που χρησιμοποιήθηκε ήταν η Postgres. Επιλέχθηκε καθώς, η δομή της βάσης έπρεπε να είναι σχεσιακή, δηλαδή υπάρχουν αυστηρά δομημένοι πίνακες με προκαθορισμένων τύπων με γραμμές και στήλες που σχετίζονται μεταξύ τους μέσω “κλειδιών”. Ταυτόχρονα, η Postgres διαθέτει το πρωτόκολλο ασφάλειας ACID (Atomicity Consistency Isolation Durability), το οποίο διασφαλίζει την εμπιστευσιμότητα στις συναλλαγές του συστήματος της βάσης. Επιπλέον, διαθέτει οριζόντια επεκτασιμότητα (horizontal scalability), δηλαδή η αναβάθμιση μπορεί να γίνει με την προσθήκη πολλαπλών

διακοσμητών (servers), που καθιστά την αναβάθμιση αυτή πιο φτηνή και ταυτόχρονα δεν περιορίζεται από τεχνολογικούς παράγοντες. Παραδείγματος χάριν, δεν υπάρχει κάποιο όριο στη μνήμη σε αντίθεση με έναν απλό υπολογιστή ή διακοσμητή. Παρακάτω εμφανίζεται το Μοντέλο Οντοτήτων - Συναρτήσεων το οποίο θα αναλύσουμε.

5.1.1 Μοντέλο Οντοτήτων - Συναρτήσεων



Εικόνα 1: ER Mode

Η βάση αποτελείται από 11 tables τα οποία χαρακτηρίζουν τις εξής οντότητες:

1. Για τις ερωτήσεις:

1.1. Questionnaire: Id (PK) , Name,

Το Id χαρακτηρίζει τον μοναδικό αριθμό του Ερωτηματολογίου,

Το Name είναι το όνομα που έχει δοθεί για το ερωτηματολόγιο αυτό

1.2. QuestionTypes: Id , Name

Ένα table που λειτουργεί ως “key -> value”. Η κάθε ερώτηση δέχεται ενός τύπου απάντηση. Όπως προαναφέρθηκε, το σύστημα είναι υλοποιημένο ώστε να δέχεται 4 τύπων απαντήσεις :

1 -> Πολλαπλής Επιλογής (αυστηρά μια επιλογή)

2-> Text

3 -> Photo

4 -> Checkbox

Αυτές είναι και οι γραμμές του συγκεκριμένου table.

1.3. Questions: Id (PK), QuestionId (FK) , QuestionType (FK)

Το Table αυτό διαθέτει όλα τα Id όλων των ερωτήσεων ανεξαρτήτως τύπου, ο τύπος τους προσδιορίζεται στην 3^η στήλη με το ξένο κλειδί “QuestionType”

1.4. QuestionTranslations : Id , QuestionId , lang_code , Question_Text , QuestionTypeId

Εκεί βρίσκονται οι ερωτήσεις που έχουν προσυμπληρωθεί για κάθε ένα ερωτηματολόγιο. Ο λόγος που υπάρχει αυτό το Table και δεν είναι συγχωνευμένο με το Questions είναι διότι μας ενδιαφέρει η πολυγλωσσικότητα. Αυτή επιτυγχάνεται μέσω της στήλης “Lang_code” έτσι έχουν μικρότερο μέγεθος τα tables και είναι πολύ πιο γρήγορα τα SQL queries. Το Question_Text διαθέτει την ερώτηση.

2. Για τις απαντήσεις :

Το ίδιο μοτίβο έχει χρησιμοποιηθεί και για τις απαντήσεις. Οι απαντήσεις στις οποίες παρέχεται κάποιο κείμενο σαν επιλογή διαθέτουν ξεχωριστό table ώστε να υποστηρίζεται η μετάφραση της δοσμένης απάντησης στην γλώσσα επιλογής. Αυτό ισχύει για τα Tables Radio_Answer_Translations και Checkbox_Answer_Translations και εκτός του δοσμένου κειμένου (text) και του score (θα αναληθεί παρακάτω) είναι ίδια με την QuestionTranslations στην δομή.

Τέλος, υπάρχει το Answer_Map που είναι υπεύθυνο για την δυναμική ακολουθία των ερωτήσεων κάθε ερωτηματολογίου. Πιο αναλυτικά, το Answer_Map διαθέτει τις εξής στήλες: Id, QuestionnaireId, QuestionId, score, nextQuestion

Αποτελεί ένα χάρτη ερωτήσεων στους οποίους ανάλογα την απάντηση που δίνει ο χρήστης σε κάθε ερώτηση η στήλη nextQuestion που είναι ξένο κλειδί του table Question (όπως και το QuestionId) διαθέτει ένα προκαθορισμένο Id που αποτελεί την επόμενη ερώτηση του ερωτηματολογίου. Έτσι, ανάλογα την απάντηση που δίνει ο χρήστης μπορεί να έχει διαφορετική ακολουθία ερωτήσεων και διαφορετικό συνολικό πλήθος ερωτήσεων που είναι η ουσία των δυναμικής του συστήματος. Φυσικά, αυτό δεν περιορίζει τα στατικά ερωτηματολόγια που μπορούν να προστεθούν στην βάση. Σ' αυτή την περίπτωση σε κάθε απάντηση του χρήστη το nextQuestion θα δείχνει στο ίδιο nextQuestion Id .

3. Επιπλέον tables:

Patients: Διαθέτει βασικά στοιχεία του χρήστη που χρειάζονται για να επιτευχθεί η σύνδεση στο σύστημα ενώ παρέχεται και η δυνατότητα εγγραφής.

Records/Forms: Τα tables αυτά είναι υπεύθυνα για την διατήρηση ιστορικού των ερωτηματολογίων.

Records: Id, FormId, PatientId, QuestId, QuestionId, AnswerId, score, QuestionTypeId, Checked, Image, Text

Διαθέτει τα ξένα κλειδιά για το PatientId, QuestId (Questionnaire), QuestionId (Question), AnswerId (Answer_Map), score , QuestionTypeId (QuestionType) και τις επιπλέον στήλες Checked (boolean), Image (byte array), Text (varchar) και score. Όπως αναφέρθηκε, ανάλογα τον τύπο δοσμένων απαντήσεων αποθηκεύονται οι τιμές στην κατάλληλη στήλη ενώ οι υπόλοιπες λαμβάνουν default τιμές null/false. Τέλος, το score που συναντήθηκε και σε προηγούμενο table (Radio_Answer_Translations / Checkbox_Translations) υπάρχει, διότι μια επιπλέον λειτουργία του συστήματος είναι η συγκέντρωση τελικής βαθμολογίας για κάθε ερωτηματολόγιο. Έτσι κάθε επιλογή σε ερώτηση πολλαπλής επιλογής ή επιλογή checkbox μπορεί να προστεθεί στους συνολικούς πόντους του χρήστη στο ερωτηματολόγιο αυτό. Οι πόντοι αυτοί ανάλογα το ερωτηματολόγιο μπορούν να παίξουν σημαντικό ρόλο στην διάγνωση του ασθενούς. Ενώ, πέρα από την υπολογισμό του αθροίσματος των "scores" των ερωτήσεων το σύστημα είναι αναπτυγμένο ώστε να μπορεί να διαχειριστεί πολλαπλούς κανόνες για κάθε ερωτηματολόγιο. Λεπτομερής ανάπτυξη της λειτουργίας αυτής στον τρόπο σχεδιασμού της θα αναπτυχθεί στην παρακάτω ενότητα.

Forms: Id, PatientId, QuestId , StartTime, EndTime, expression, score

Ο πίνακας έχει τα ξένα κλειδιά PatientId, QuestId και τις επιπλέον στήλες StartTime, EndTime, expression, score. Η StartTime σηματοδοτεί ότι ο χρήστης ξεκινάει να απαντάει το ερωτηματολόγιο και η EndTime το τέλος. Οι τιμές αυτές αρχικοποιούνται στον

Questionnaire Controller ενώ οι expression και score στον admin που θα γίνει αναφορά παρακάτω για τον λόγο ύπαρξής τους.

5.2 Αρχιτεκτονική Back-End

Η αρχιτεκτονική που χρησιμοποιεί το σύστημα στο κομμάτι του Back-End είναι συμβατή με το μοντέλο MVC (Model View Controller). Κατ' αυτό τον τρόπο δημιουργείται το βασικό περιβάλλον αλληλεπίδρασης του χρήστη με το σύστημα. Πιο συγκεκριμένα, ο Controller είναι ο διαμεσολαβητής δεδομένων από το Model και View, είναι υπεύθυνος να ενημερώνει τις καταστάσεις τους και ωθεί το Model σε τυχόν αλλαγές. Το Model αποτελεί την δομή των πινάκων της βάσης και μέσω αυτού ανακτώνται (retrieve) και αποθηκεύονται (insert) τα νέα δεδομένα. Τέλος, η View είναι υπεύθυνη για το οπτικό κομμάτι στον χρήστη (User Interface) και μέσω αυτής ο χρήστης αλληλεπιδρά με τον Controller ώστε η πλοήγηση στο σύστημα να φέρνει ανανεωμένο περιεχόμενο. Έχει χρησιμοποιηθεί το Visual Studio 2022 και το framework ASP NET CORE σε γλώσσα προγραμματισμού C#. Πρόκειται για ένα από τα πιο εδραιωμένα frameworks ιδιαίτερα εμπιστεύσιμο, διαθέτει πληθώρα επιλογών και λειτουργιών. Ενώ η Microsoft που έχει την ιδιοκτησία του φροντίζει τόσο για την συντήρηση του όσο και για την εκπαίδευση των προγραμματιστών σε ενεργά forums αλλά και δικά της χρήσιμα tutorials.

Για το κομμάτι της παραμετροποίησης του WebAPP έχουν χρησιμοποιηθεί οι υπηρεσίες Session, Cookies, Routing, Redirection, ApplicationDbContext, Authentication και φυσικά η δομή MVC.

- Session: Για προσωρινή αποθήκευση δεδομένων στην ιστοσελίδα,
- Cookies: Για αποθήκευση των διαπιστευτηρίων του χρήστη,
- Routing: Για την μετάβαση των σελιδών,
- Redirections: Για την εναλλαγή Controller,
- Authentication: Χρησιμοποιείται ως ένα middleware που ελέγχει πριν την φόρτωση της σελίδας αν υπάρχουν Cookies τα οποία επιβεβαιώνουν την σύνδεση του χρήστη στην σελίδα αλλιώς γίνεται Redirect στο Login Page
- ApplicationDbContext: Υπεύθυνη υπηρεσία για την αλληλεπίδραση με την βάση.

Η τελευταία υπηρεσία δημιούργησε όλους τους πίνακες της Βάσης Δεδομένων με βάσης τους κανόνες που έχουν οριστεί στον φάκελο Models.

```

using formsWeb.Models;
using Microsoft.EntityFrameworkCore;

namespace formsWeb.Data
{
    24 references
    public class ApplicationDbContext: DbContext
    {
        0 references
        public ApplicationDbContext(DbContextOptions<ApplicationDbContext> options) : base(options)
        { }

        5 references
        public DbSet<Patient> Patients { get; set; }

        4 references
        public DbSet<Questionnaire> Questionnaire { get; set; }

        2 references
        public DbSet<Question> Questions { get; set; }
        2 references
        public DbSet<QuestionTranslation> QuestionTranslations { get; set; }
        0 references
        public DbSet<QuestionType> QuestionTypes { get; set; }

        2 references
        public DbSet<Radio_Answer_Translation> Radio_Answer_Translations { get; set; }

        1 reference
        public DbSet<Answer_map> Answer_Maps { get; set; }
        4 references
        public DbSet<Record> Records { get; set; }

        4 references
        public DbSet<Form> Forms { get; set; }
        1 reference
        public DbSet<CheckBox_Answer_Translation> CheckBox_Answer_Translations { get; set; }
    }
}

```

Εικόνα 2: Δομή Αρχείου "Class.cs" στον Φάκελο "Data"

Παράλληλα, εκτός από την αποθήκευση και ανάκτηση δεδομένων απο την βάση παραμετροποιεί τα ήδη υπάρχοντα tables και τις στήλες όπως χρησιμοποιήθηκε στο σύστημα με την μορφή Migration. Η διαδικασία αυτή, γίνεται με τον Package Manager controller (που εγκαθιστάται αυτόματα κατά την επιλογή δημιουργίας WebApp στο Visual Studio 2022) με τις εντολές:

- Add-migration "AddYourMigrationName"
- Update-database

Σε αυτό το σημείο θα αναληθούν οι Controllers του συστήματος ενώ τα Models αναλύθηκαν στο 5.1 και τα Views συνοπτικά στο 5.3.

HomeController: Διαθέτει την αρχική Σελίδα αλλά το σύστημα δεν είναι χρήσιμο χωρίς σύνδεση οπότε δίνει τις επιλογές για Login, Register (View) που υπόκεινται και τα 2 actions στον Controller του Patient

PatientController: Διαθέτει τα Actions Login, Register, Index. Το πρώτο action που παρουσιάζεται στην παρακάτω εικόνα.

```
[HttpPost]
0 references
public async Task<IActionResult> Login(Patient pt, string returnUrl)
{
    Console.WriteLine(pt);
    var first:Patient? = _db.Patients // DbSet<Patient>
        .Where(c:Patient => c.username == pt.username && c.password == pt.password) // IQueryable<Patient>
        .FirstOrDefault();
    if (first != null) {
        Console.WriteLine(returnUrl);

        //return Redirect("./Questionnaire/Question");
        var claims = new List<Claim>
        {
            new Claim(type: ClaimTypes.Name, value: pt.username),
            new Claim(type: "Id", value: first.Id.ToString())
        };

        var claimsIdentity = new ClaimsIdentity(claims, CookieAuthenticationDefaults.AuthenticationScheme);
        var authProperties = new AuthenticationProperties { };

        await HttpContext.SignInAsync(CookieAuthenticationDefaults.AuthenticationScheme,
            new ClaimsPrincipal(claimsIdentity),
            authProperties); // Task
    }
    return View();
}
```

Εικόνα 3: Δομή Login Action

Πρόκειται για έναν έλεγχο τις ύπαρξης των δοσμένων διαπιστευτηρίων, δημιουργίας νέου Cookie που δεχεται τα στοιχεία του χρήστη και τέλος την ανατροφοδότηση στην σελίδα της επιλογής του χρήστη όταν τα δεδομένα είναι σωστά. Ενώ με όμοιο τρόπο έχει υλοποιηθεί το Register στο action Register.

AdminController: Ο controller αυτός είναι υπεύθυνος για να διαχειρίζεται τα συνολικά δεδομένα του συστήματος και χρειάζεται authentication.

Κατά την μετάβαση στην σελίδα: localhost:5001/admin/index

Ο controller τρέχει την ασύγχρονη συνάρτηση ανανέωσης δεδομένων στον πίνακα Forms. Με άλλα λόγια, εκεί αρχικοποιούνται οι τιμές των στηλών expression/score. Όπως έχει ήδη αναφερθεί, τα ερωτηματολόγια διαθέτουν κάποιους κανόνες βαθμολόγησης. Η expression είναι μία λίστα με ονομασίες των κανόνων αυτών και η scores διαθέτει τις τιμές τους. Αυτά εκχωρούνται μόνο σε αυτή την συνάρτηση και μόνο κατά την πρώτη μετάβαση στο Admin

Page. Η ίδια διατρέπει όλα τα ερωτηματολόγια της Forms και για κάθε ένα που έχει μη αρχικοποιημένη τιμή (null) στην expression εφαρμόζει τους κανόνες του εκάστως ερωτηματολογίου. Τελικά, οι κανόνες αυτοί αποθηκεύονται ταυτόχρονα σε διαφορετικές λίστες με την προϋπόθεση ότι το Ι-οστό όνομα στην λίστα "expression" έχει στην Ι-οστή θέση της λίστας "score" την τιμή που το αντιπροσπεύει.

```
foreach ( Form form in forms)
{
    if (form.QuestId == 1)
    {
        //QuestId is The Barthel Index
        //Sum = sum of all scores
        //Average = sum of all scores / all records
        //Custom1 = sum of 1-5 records / 2 + sum of 6-10 / 5
        double custom1 = 0;
        double custom2 = 0;
        double average = 0;
        var question_n = 0;
        double sum = 0;
        if (form.expression != null) continue;

        foreach ( Record record in records.Where(c => c.FormId == form.Id)){
            question_n++;
            sum = sum + (double) record.score;
            average = sum;
            //Its a static Questionnaire that's why divide by 5
            if(question_n < 6)
            {
                custom1 = custom1 + (double)record.score;
            }
            else
            {
                custom2 = custom2 + (double)record.score;
            }
        }
        custom1 = (double)custom1/5;
        average = (double)average/question_n;
        custom2 = (double)custom2/5;
        custom2 = (double)custom2+custom1;

        form.expression = new List<string>();
        form.expression.Add("Sum");
        form.expression.Add("Average");
        form.expression.Add("Custom");

        form.score = new List<double>();
        form.score.Add(sum);
        form.score.Add(average);
        form.score.Add(custom2);
        _db.Entry(form).State = EntityState.Modified;
    }
    await _db.SaveChangesAsync();
}
```

Εικόνα 4: Υπολογισμός scores στο ερωτηματολόγιο "The Barthel Index"

Ένα action επιπλέον είναι το Details που ανατρέπει την βάση για ένα συγκεκριμένο Id και προσφέρει το ιστορικό του χρήστη σε ένα συγκεκριμένο δοσμένο Id (το Id είναι του πίνακα Forms). Αν υπάρχει το δοσμένο αυτό Id ανατρέπει όλα τα αποθηκευμένα Records της βάσης και τα παρέχει στον χρήστη στη διεύθυνση "/admin/Details/+Id", ενώ παρέχεται και ανακατεύθυνση από το "Admin/Index" action με την επιλογή "More" σε κάποιο ερωτηματολόγιο.

QuestionnaireController: Ο controller είναι υπεύθυνος για την επιλογή του επιθυμητού Questionnaire του χρήστη, την δυναμική επιλογή των ερωτήσεων και των αντιστοιχων απαντήσεων καθώς και την τελική προβολή του συνοπτικού “Summary” του.

```
[Authorize]
public IActionResult Index()
{
    HttpContext.Session.Clear();
    var model = new QuestionnaireViewModel();
    var question_model = new QuestionViewModel();
    var combinedViewModel = new Combined()
    {
        Questionnaire = model,
        Question = question_model
    };
    var obj = _db.Questionnaire
        .OrderBy(c => c.Id)
        .ToList();

    model.QuestSelectList = new List<SelectListItem>();
    foreach (var element in obj)
    {
        model.QuestSelectList.Add(new SelectListItem
        {
            Text = element.Name, Value = element.Id.ToString()
        });
    }

    string user = JsonConvert.SerializeObject(combinedViewModel);
    _ctx.HttpContext?.Session.SetString("User", user);
    return View(combinedViewModel);
}
```

Εικόνα 5: Προβολή των διαθέσιμων ερωτηματολογίων προς απάντηση

Στο σημείο αυτό χρησιμοποιείται για πρώτη φορά η υπηρεσία του Session όπου αποθηκεύεται το CombinedViewModel που διαθέτει τα QuestionViewModel και QuestionnaireViewModel. Τα συγκεκριμένα ViewModel παρέχουν στον χρήστη τα δεδομένα απο την βάση και επιστρέφουν τις επιλογές του χρήστη. Αναλυτικότερα, για το QuestionnaireViewModel η λίστα περιέχει όλα τα διαθέσιμα ερωτηματολόγια και η QuestSelected επιστρέφει το Id του Questionnaire της επιλογής ώστε να ξεκινήσει η διαδικασία των απαντήσεων. Το QuestionViewModel έχει το Id της συγκεκριμένης ερώτησης που θα απαντηθεί, το QuestId που είναι το Id του Questionnaire που επέλεξε προηγουμένως. Ακόμα, όλους τους αποδεκτούς τύπους δεδομένων που είναι αποδεκτές ως απάντηση καθώς και τις προτεινόμενες απαντήσεις όπου παρέχονται.

- Options = πολλαπλές επιλογές
- OptionsCheck = επιλογες του Checkbox
- ImageBytes = τα bytes της εικόνας που ανέβασε ο χρήστης
- Text = το κείμενο που έδωσε ο χρήστης ως απάντηση

- Checked (boolean) = Αν πάτησε το checkbox ή όχι
- Selected (int) = Η επιλογή που έκανε στο πολλαπλής

```
namespace formsWeb.Models.ViewModel
{
    public class QuestionnaireViewModel
    {
        public List<SelectListItem>? QuestSelectList { get; set; }
        public int QuestSelected { get; set; }
    }
}
```

Εικόνα 6: Δομή QuestionnaireViewModel

```
namespace formsWeb.Models.ViewModel
{
    public class QuestionViewModel
    {
        //Question Id
        public int Id { get; set; }
        [ForeignKey("Questionnaire")]
        public int QuestId { get; set; }
        //Question Text from question translation
        public string? Question { get; set; }
        //All given radio selections from checkbox_translations
        public IList<Radio_Answer_Translation>? Options { get; set; }
        //All given checkbox selections from checkbox_translations
        public IList<CheckBox_Answer_Translation>? OptionsCheck { get; set; }
        public byte[]? ImageBytes { get; set; }
        public string? Text { get; set; }
        public int QuestionType { get; set; }
        public bool Checked { get; set; }
        //Then one that the user selected
        public int Selected { get; set; }
        //The score from the user's selection
        public double Score { get; set; }
        public int total { get; set; } = 0;
        public int next { get; set; }
    }
}
```

Εικόνα 7: Δομή QuestionViewModel

Η προηγούμενη μέθοδος είναι μια GET μέθοδος που παρέχονται όλα τα ερωτηματολόγια. Συνεχίζοντας, η επιλογή του ερωτηματολογίου, η απάντηση κάθε ερώτησης και “λήξη” του γίνεται απο μία POST μέθοδο που ανάλογα τα δεδομένα ανανεώνει τα δεδομένα στην View. Αυτό επιτυγχάνεται από την συνάρτηση Question. Η ίδια, ελέγχει τρεις καταστάσεις των δεδομένων που παρέχονται.

- Questionnaire = null

- Question = null
- Question.id = 0

Κατά την επιλογή του Questionnaire δηλαδή στην πρώτη κατάσταση της συνάρτησης γνωρίζουμε ότι το μοντέλο που θα επιστραφεί στο Backend θα διαθέτει τιμή (QuestSelected) και θα ξεκινήσει η διαδικασία απάντησης του επιλεγμένου. Εκεί δημιουργείται νέο entry στον πίνακα Forms και καταχωρείται η ημερομηνία και ώρα που ξεκίνησε μαζί με τα ξένα κλειδιά: Id του ασθενή (PatientId) και φυσικά το Id του ερωτηματολογίου (QuestId). Επειδή το μοντέλο αυτό επικοινωνεί με τη Backend με την μορφή sessions τα δεδομένα αυτά καταστρέφονται στην ανανέωση της View πράγμα που φυσικά θέλαμε, ώστε το σύστημα να είναι πολύ πιο γρήγορο και χωρίς πολλά άχρηστα δεδομένα. Στην συνέχεια, η κατάσταση θα είναι απάντηση κάποιας ερώτησης οπότε δημιουργείται νέο entry στον πίνακα Records με FormId το Id που δημιουργήθηκε πριν και αποθηκεύονται τα δεδομένα που επιστράφηκαν ενώ εκτελείται ένα "Query" στον Answer_Map ώστε να βρεθεί η επόμενη ερώτηση που ακολουθείται από την select απο:

1. Questions
2. Question_Translations
3. Radio_Answer_Translations/Checkbox_Answer_Translations.

Αν ο τύπος ερώτησης δέχεται Image ή Text δεν θα γίνει κάποιο Query γι'αυτά οπότε δεν θα εκτελεστεί το 3.


```

var questions = _db.Questions
    .Where(c => c.Id == mode.Question.Id)
    .Select(c => new Question
    {
        Id = c.Id,
        QuestionId = c.QuestionId,
        questionType = c.questionType
    }).FirstOrDefault();

if(questions != null)
{
    //model.Question.Id = questions.Id;
    //model.Question.QuestionId = questions.QuestionId;
    //model.Question.QuestionType = questions.questionType;
    mode.Question.Id = questions.Id;
    mode.Question.QuestionId = questions.QuestionId;
    mode.Question.QuestionType = questions.questionType;
    var quest_transl = _db.QuestionTranslations
        .Where(c => c.QuestionId == mode.Question.QuestionId && c.Id == mode.Question.Id )
        .Select(c => new QuestionTranslation
        {
            Id = c.Id,
            lang_code = c.lang_code,
            question_text = c.question_text,
        }).FirstOrDefault();
    mode.Question.Question = quest_transl.question_text;
    if(mode.Question.QuestionType == 1)
    {
        var questionnaire = await Radio_Answer(mode.Question.QuestionId , mode.Question.Id );

        mode.Question.Options = questionnaire;
    }
    else if (mode.Question.QuestionType == 4)
    {
        var questionnaire = await Checkbox_Answer(mode.Question.QuestionId, mode.Question.Id);
        mode.Question.OptionsCheck = questionnaire;
    }
}

```

Εικόνα 8: Διαδικασία προετοιμασίας της View για την επόμενη ερώτηση

Τέλος, κατα σύμβαση, η διαδικασία τελειώνει όταν η τελευταία ερώτηση ζητάει από την Answer_Maps την επόμενη ερώτηση και η επιστρεφόμενη τιμή του nextQuestion έχει μηδενική τιμή. Εκεί, καταγράφεται η τελική ώρα στην οποία απαντήθηκε η τελευταία ερώτηση και γίνεται ανακατεύθυνση στην “Summary”.

```

//END OF THE QUESTIONNAIRE
if(mode.Question.Id == 0)
{
    mode.Forms.End = DateTime.Now.ToUniversalTime();

    var entity = _db.Forms.FirstOrDefault(e => e.Id == mode.Forms.Id);

    if (entity != null)
    {
        entity.End = DateTime.Now.ToUniversalTime();
        _db.SaveChanges();
    }

    return View("Summary", mode);
}

```

Εικόνα 9: Τέλος Ερωτηματολογίου

5.3 Αρχιτεκτονική Front-End

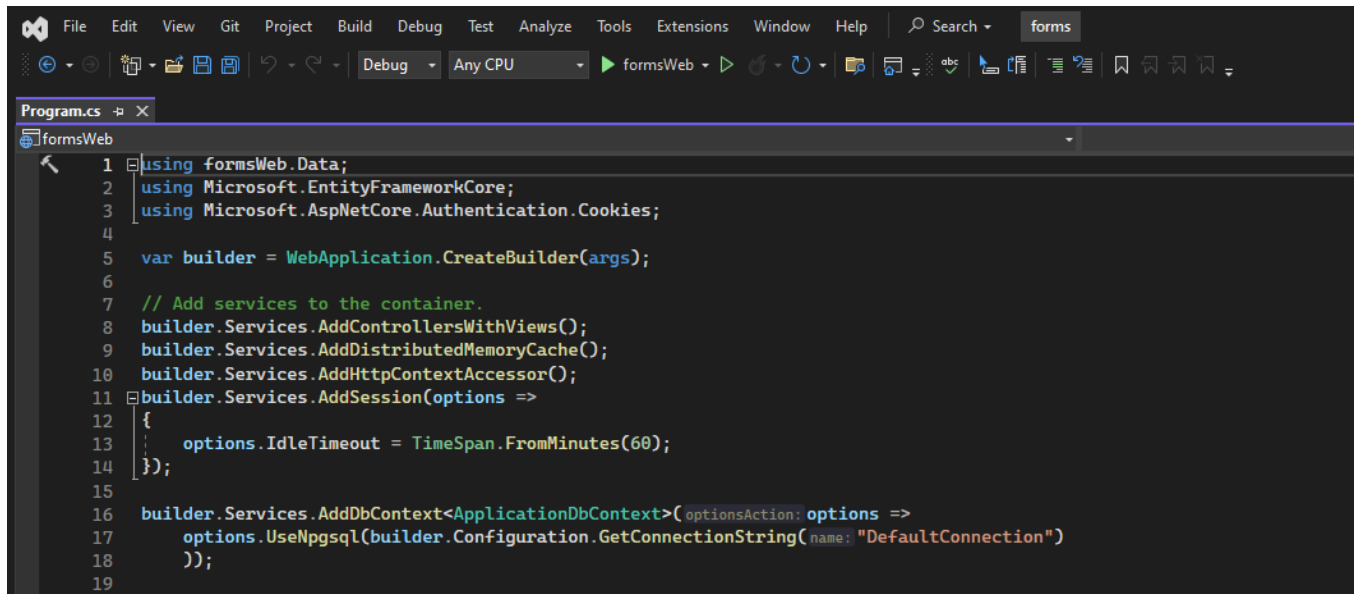
Το FrontEnd το διαχειρίζεται το σύστημα σε server side, δηλαδή δεν υπάρχει κάποιο εξωτερικό Framework. Χρησιμοποιούνται Razor Pages που είναι ένα χαρακτηριστικό του Backend Framework ASP NET CORE και είναι ικανό να δημιουργήσει δυναμικές σελίδες HTML. Σε κάθε περίπτωση το σύστημα χρησιμοποιεί HTML CSS και Javascript ενώ ενσωματώνεται η χρήση της C#. Με αυτό τον τρόπο δημιουργείται μια βασική διεπαφή με τον χρήστη. Παρα την απλότητα τους μπορούν να προσφέρουν ποικίλα πλεονεκτήματα στο σύστημα. Για παράδειγμα, ενσωματώνουν πλήρως και αυτόματα τα ενδιάμεσα λογισμικά (middlewares) που έχουν χρησιμοποιηθεί στο WebApp. Είναι ιδιαίτερα βοηθητικές για την γρήγορη ανάπτυξη εφαρμογών ιστού. Ωστόσο, είναι μία πλήρως επεκτάσιμη επιλογή για Frontend γιατί μπορούν να υποστηρίξουν σύγχρονα client-side frameworks όπως Vue, Angular για να βελτιώσουν την εμπειρία του χρήστη.

6. Οθόνες της εφαρμογής -Οδηγίες Χρήσεως

Στην συγκεκριμένη υποενότητα θα αναλυθούν οι οθόνες εφαρμογής της πλατφόρμας. Πιο συγκεκριμένα, θα δούμε αναλυτικά τον τρόπο σχεδιασμού της βάσης, την δομή του Backend και σε κάθε περίπτωση την οθόνη που προβάλλεται στον χρήστη.

6.1 Οθόνη και οδηγίες χρήσης του Backend

Όπως αναφέρθηκε και προηγουμένως, για το Backend χρησιμοποιήθηκε το Microsoft Visual Studio 2022. Αφού εκκινήσουμε το Web APP με την προαναφερθείσα εφαρμογή βλέπουμε ότι μπορούμε να τρέξουμε το πρόγραμμα πατώντας την επιλογή “FormsWeb” ή οποία θα εκκινήσει το πρόγραμμα μας στην τοπική διεύθυνση: <https://localhost:5001/>



```
1 using formsWeb.Data;
2 using Microsoft.EntityFrameworkCore;
3 using Microsoft.AspNetCore.Authentication.Cookies;
4
5 var builder = WebApplication.CreateBuilder(args);
6
7 // Add services to the container.
8 builder.Services.AddControllersWithViews();
9 builder.Services.AddDistributedMemoryCache();
10 builder.Services.AddHttpContextAccessor();
11 builder.Services.AddSession(options =>
12 {
13     options.IdleTimeout = TimeSpan.FromMinutes(60);
14 });
15
16 builder.Services.AddDbContext<ApplicationDbContext>(optionsAction: options =>
17     options.UseNpgsql(builder.Configuration.GetConnectionString(name: "DefaultConnection")
18 ));
19
```

Εικόνα 10

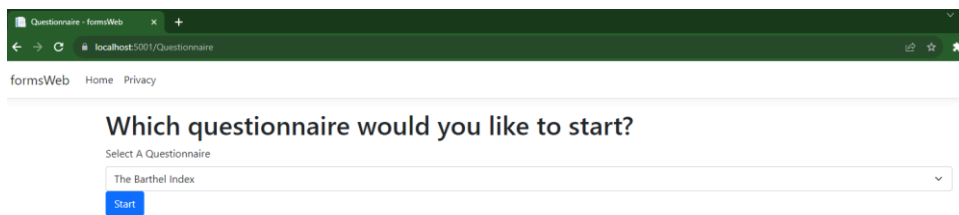
6.2 Θρόνες και οδηγίες χρήσης εφαρμογής

Για την οπτικοποίηση των σεναρίων έχουν χρησιμοποιηθεί όπως προείπαμε Razor Pages.



Εικόνα 11: Σύνδεση χρήστη στην εφαρμογή

Έπειτα από την σύνδεση του χρήστη, ακολουθεί η επιλογή του ερωτηματολογίου που πρόκειται να συμπληρώσει (Εικόνα 12).

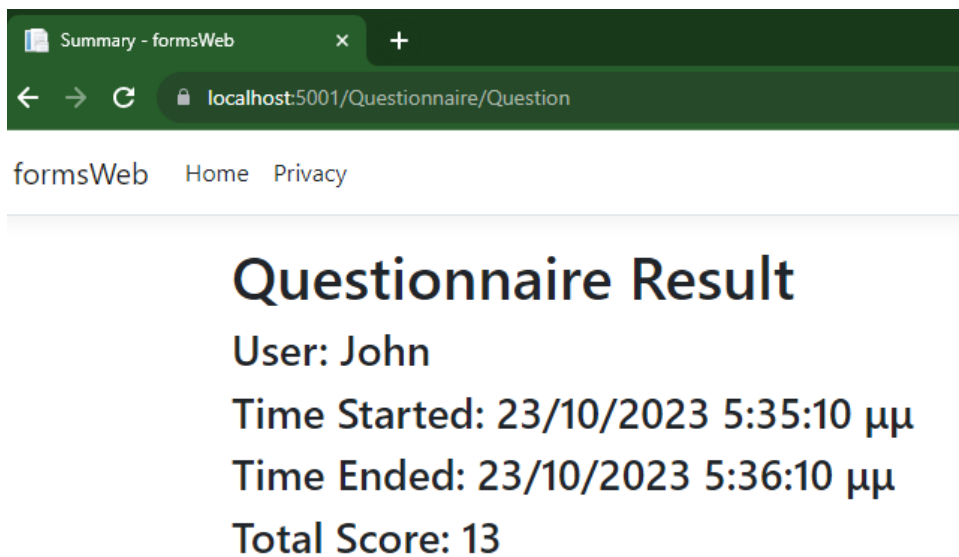


Εικόνα 12: Επιλογή ερωτηματολογίου

Μετά το στάδιο αυτό ακολουθούν ερωτήσεις που καλείται να απαντήσει οι οποίες αφορούν διάφορες καταστάσεις και στάδια της υγείας και της καθημερινής του ζωής ανάλογα την επιλογή του. Οι ερωτήσεις τύπου πολλαπλής επιλογής περιλαμβάνουν από μία έως τέσσερις απαντήσεις από τις οποίες μπορεί να επιλεγεί μόνο μία κάθε φορά (Εικόνα 13).

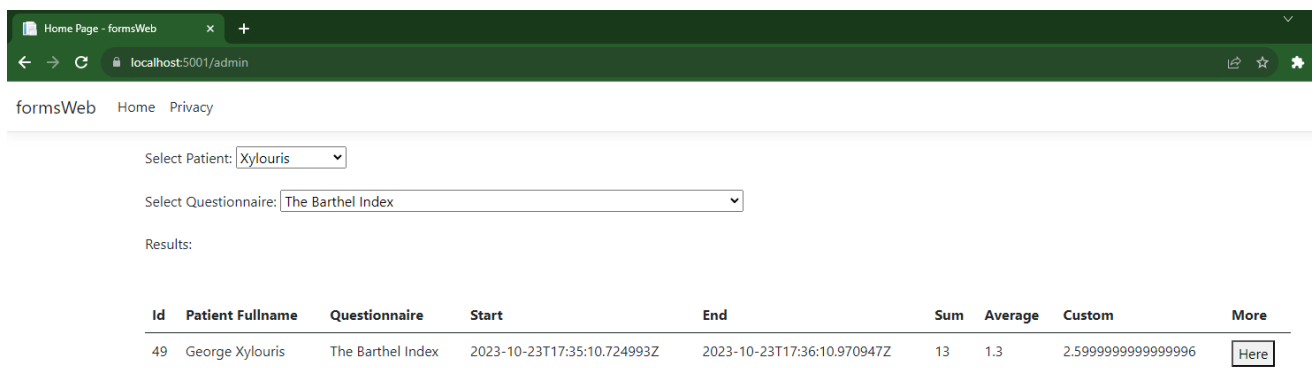
Εικόνα 13: Παράδειγμα ερώτησης

Μετά την επιτυχή ολοκλήρωση ενός ερωτηματολογίου, ο χρήστης βλέπει στην οθόνη το αυτοματοποιημένο μήνυμα με το αποτέλεσμα τα στοιχεία του, την ώρα έναρξης και λήξης του ερωτηματολογίου καθώς και τη συνολική βαθμολογία που συγκέντρωσε (Εικόνα 14).

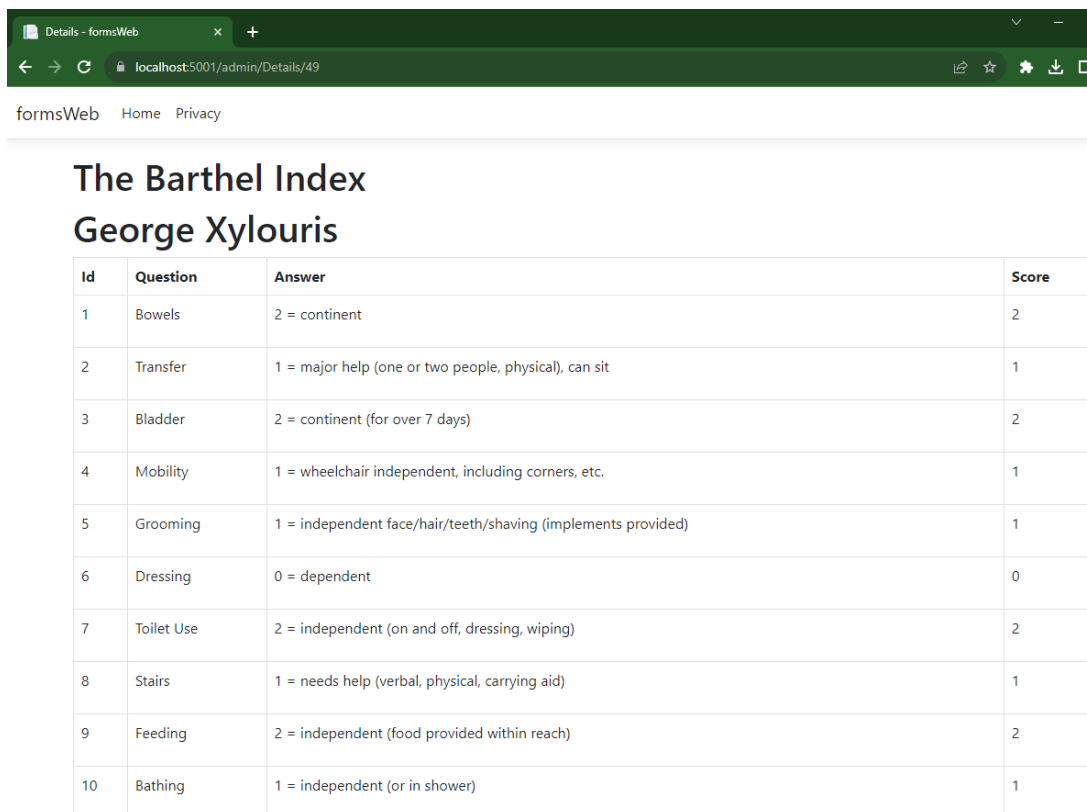


Εικόνα 14: Αυτοματοποιημένο μήνυμα κατά τη λήξη του ερωτηματολογίου

Κάθε χρήστης έχει πρόσβαση στο administration page, μέσω της χρήσης του παρακάτω URL <http://localhost:5001/admin> όπως φαίνεται στην εικόνα 6. Στην συγκεκριμένη οθόνη, ο χρήστης μπορεί να επιλέξει τα ολοκληρωμένα ερωτηματολόγια είτε επιλέγοντας κάποιο χρήστη είτε μέσω της επιλογής κάποιου ερωτηματολογίου. Εκεί έχει την επιλογή “More” μέσω της οποίας ανακατευθύνεται στο ιστορικό απαντήσεων του χρήστη στο επιλεγμένο ερωτηματολόγιο όπου του παρέχονται αναλυτικά όλες οι απαντήσεις που δόθηκαν και τα scores που έλαβε (Εικόνα 15).



Εικόνα 15: Administration page

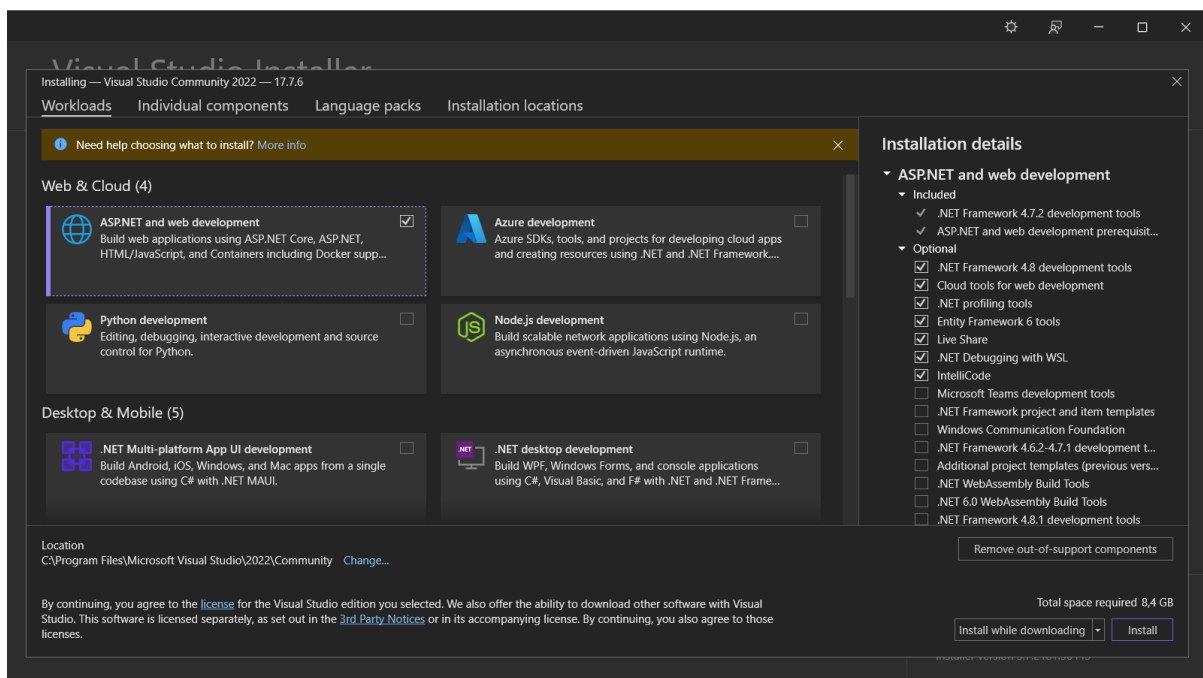


Id	Question	Answer	Score
1	Bowels	2 = continent	2
2	Transfer	1 = major help (one or two people, physical), can sit	1
3	Bladder	2 = continent (for over 7 days)	2
4	Mobility	1 = wheelchair independent, including corners, etc.	1
5	Grooming	1 = independent face/hair/teeth/shaving (implements provided)	1
6	Dressing	0 = dependent	0
7	Toilet Use	2 = independent (on and off, dressing, wiping)	2
8	Stairs	1 = needs help (verbal, physical, carrying aid)	1
9	Feeding	2 = independent (food provided within reach)	2
10	Bathing	1 = independent (or in shower)	1

Εικόνα 16: Αναλυτικά δεδομένα ερωτηματολογίου χρήστη

7.Οδηγίες εγκατάστασης του συστήματος

Η διαδικασία εγκατάστασης και δοκιμής του συστήματος είναι ιδιαίτερα εύκολη και θα αναλυθούν όλα τα στάδια παρακάτω. Αρχικά χρειάζεται να κατεβούν τα εργαλεία που τρέχουν το σύστημα που δεν είναι άλλα από τα Visual Studio 2022 (Community) και Postgresql (pgadmin 4). Μετά την λήψη των εργαλείων χρειάζεται η εγκατάσταση τους ανάλογα τις προτιμήσεις του developer. Σημαντικό είναι κατά την εγκατάσταση του Visual Studio 2022 να επιλεχθεί η επιλογή “ASP NET and web.development” ώστε να εγκατασταθούν αυτόματα τα εργαλεία-υπηρεσίες που διαχειρίζεται το πρόγραμμα.



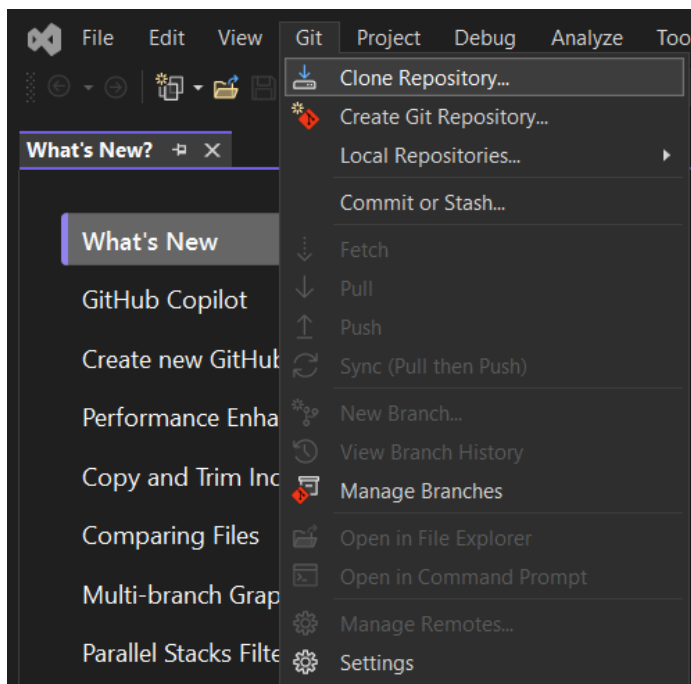
Εικόνα 17: Επιλογή ASP.NET and web Development

Έπειτα χρειάζεται να πάρουμε τον κώδικα από το Github με το link:

<https://github.com/prlijohn/loannisXylourisCSD4225>

Η να τον κάνουμε “clone” όπως φαίνεται παρακάτω μέσω του link:

<https://github.com/prlijohn/loannisXylourisCSD4225.git>



Εικόνα 18

Ουσιαστικά έχει τελειώσει η διαδικασία με το Backend και χρειάζεται να “ενωθεί” με την βάση.

Κατά την εγκατάσταση του pgadmin 4 είναι σημαντικό να δωθεί προσοχή σε 2 εκχωρήσεις όπως φαίνεται στο Connection String του Backend.

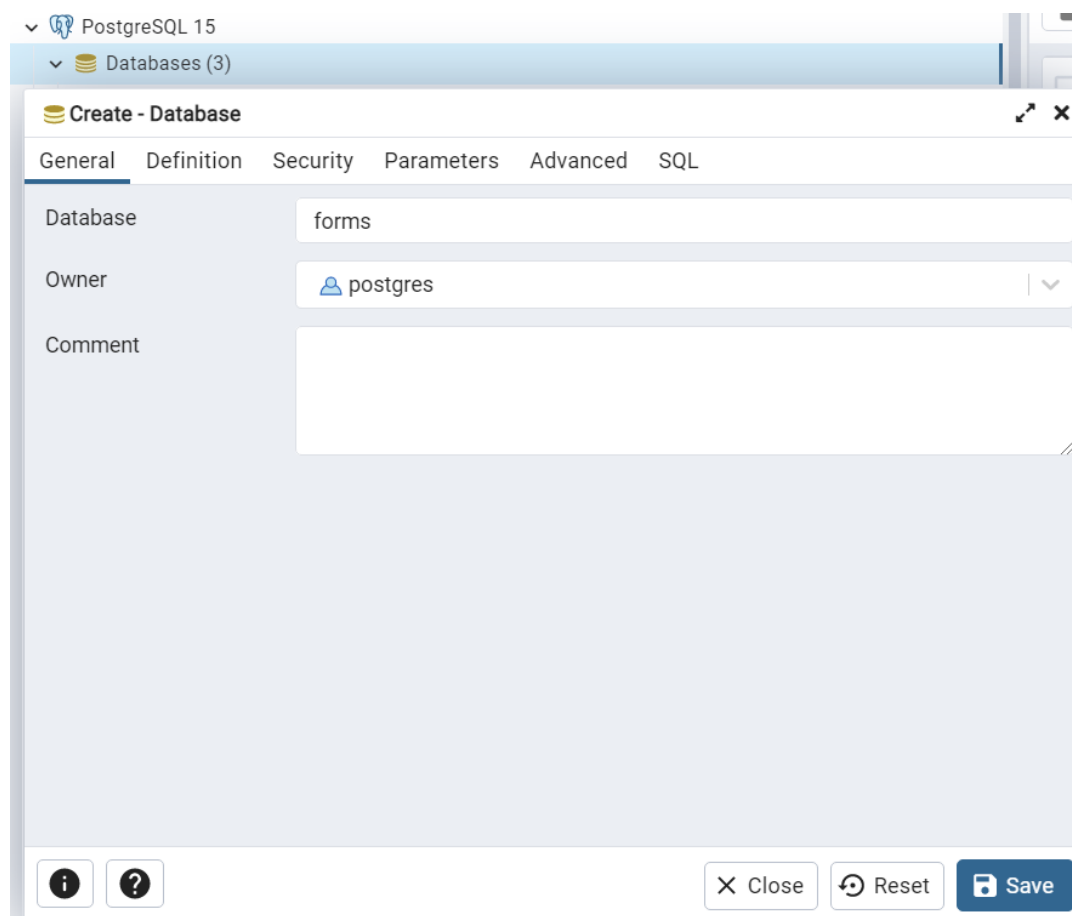
```
"AllowedHosts": "*",
"ConnectionStrings": {
  "DefaultConnection": "Server=localhost;port=5432;Database=forms;Integrated Security=true;Pooling=true;User Id=postgres;Password=12345;"
}
```

Εικόνα 19: appsettings.json

Ο developer οφείλει να ακολουθήσει τα δεδομένα του “Default Connection” που φαίνονται παραπάνω είτε να τα επεξεργαστεί τα ήδη υπάρχοντα ανάλογα με τις ανάγκες του. Σε κάθε περίπτωση τα δεδομένα αυτά πρέπει να είναι πανομοιότυπα τόσο στην Βάση Δεδομένων όσο και στο Backend.

Τέλος, ο developer αν ακολούθησε το μοντέλο του “Connection String” θα πρέπει να δημιουργήσει ένα κενό database με όνομα “forms” πατώντας:

Δεξί click στην επιλογή “Databases” -> Create -> Database... και συμπληρώνουμε μόνο το name και πατάμε Save.



Εικόνα 20

Έπειτα, πατάμε δεξί click στην νέα forms και Restore.

Συμπληρώνουμε μόνο το filename που είναι το dump_file που βρίσκεται στο github και αποτελεί τα δεδομένα της Βάσης Δεδομένων. Πατώντας Restore θα ξεκινήσει μια μικρή διαδικασία δημιουργίας της βάσης. Όταν η διαδικασία αυτή τελειώσει το σύστημα είναι έτοιμο να ξεκινήσει.

8. Επίλογος

Στη συγκεκριμένη ενότητα, θα γίνει τόσο η σύνοψη της εργασίας αυτής αλλά και θα αναλυθούν οι αδυναμίες και οι μελλοντικές εργασίες οι οποίες μπορούν να γίνουν για την περαιτέρω βελτίωση του συστήματος.

8.1 Σύνοψη

Το σύστημα που δημιουργήθηκε καταφέρνει με επιτυχία να οδηγήσει αφενώς στην επιλογή και αφετέρου στην απάντηση ενός από τα παρέχοντα ερωτηματολόγια. Η διαδικασία είναι πολύ απλή όπως έχει αναφερθεί σε αρκετά από τα επιμέρους κεφάλαια της εργασίας αυτής.

Η χρήση της παραπάνω εφαρμογής απευθύνεται σε έναν χρήστη που είτε επιθυμεί να αξιολογήσει την κατάσταση της υγείας του είτε επειδή ο γιατρός του τον έχει παραπέμψει στο παρόν σύστημα. Είναι σημαντικό ότι το σύστημα διαθέτει ιστορικό και δίνει την δυνατότητα σύγκρισης των αποτελεσμάτων που μπορούν να βοηθήσουν στην εύρεση του προβλήματος. Ο ίδιος διαθέτει στο προφίλ του το ιστορικό του όλων των ερωτηματολογίων που έχει απαντήσει και φυσικά όλες οι δοσμένες απαντήσεις σε κάθε τύπου ερώτησης. Στόχος του συστήματος είναι μια πρώιμη γνωμάτευση, ενημέρωση της υγείας, η παροχή αυτής της πληροφορίας μπορεί να βοηθήσει ακόμα και τον οικογενειακό γιατρό χωρίς την ανάγκη επίσκεψης του, ενώ ταυτόχρονα είναι σημαντική η αντίληψη που προβάλλεται μέσω αυτού από τον χρήστη για την ποιότητα της ζωής του.

Το σύστημα έχει υλοποιηθεί με την χρήση C# με το ASP NET Core Web App Framework για το Backend κομμάτι, η βάση δεδομένων έχει δημιουργηθεί στην PostgreSQL ενώ το frontend παρέχεται από το ίδιο το Backend Framework και είναι τα Razor Pages. Η γλώσσα προγραμματισμού είναι η C# σε όλο το WebApp.

Συμπερασματικά, το σύστημα λειτουργεί πλήρως και ικανοποιεί όλες τις ανάγκες και απαιτήσεις που χρειάζονται για να μπορεί να παρέχει την συμπλήρωση δυναμικών ερωτηματολογίων τύπου PROMs από τον χρήστη.

8.2 Αδυναμίες και Μελλοντική εργασία

Το σύστημα διαθέτει κάποιες αδυναμίες και σίγουρα μπορεί να βελτιωθεί σε πολλούς τομείς. Αρχικά, είναι σημαντική η δυνατότητα παροχής στον χρήστη πληθώρα επιλογών ως προς τα ερωτηματολόγια. Διαθέτει 2 προς το παρόν χωρίς να σημαίνει αυτό ότι δεν είναι ικανό να υποστηρίξει επιπλέον. Ταυτόχρονα, θα πρέπει ο AdminController να είναι διαχειρίσιμος μόνο από τον admin ή από συγκεκριμένους γιατρούς, σε κάθε περίπτωση θα

πρέπει να υπάρξει κάποιας μορφής “ρόλων” στον τύπο χρηστών και να μην είναι διαθέσιμα όλα τα στοιχεία σε όλους τους χρήστες. Μία επιπλέον λειτουργία που θα μπορούσε να προστεθεί είναι τα “Details” που είναι οι απαντήσεις για κάθε ερωτηματολόγιο να έχουν την επιλογή για Export σαν PDF/XLSX.

9. Πηγές

Έμβλημα Πανεπιστημίου Κρήτης: <https://www.uoc.gr/university/logotipo/emvlima.html>

Visual Studio 2022: <https://visualstudio.microsoft.com/vs/>

C#: <https://learn.microsoft.com/en-us/dotnet/csharp/>

ASP NET CORE 6.0: <https://learn.microsoft.com/en-us/aspnet/core/release-notes/aspnetcore-6.0>

Razor Pages: <https://learn.microsoft.com/en-us/aspnet/core/razor-pages/>

PostgreSQL: <https://www.postgresql.org/about/>

GET/POST requests: https://www.w3schools.com/tags/ref_httpmethods.asp

Canadian Institute for Health Information: <https://www.cihi.ca/en/patient-reported-outcome-measures-proms>

Ερωτηματολόγια που χρησιμοποιούνται απο την βάση:

The Barthel Index: <https://www.albertahealthservices.ca/assets/about/scn/ahs-scn-bjh-hf-barthel-index-of-adls.pdf>

LAWTON – BRODY(I.A.D.L): <https://www.alz.org/careplanning/downloads/lawton-iadl.pdf>