Πανεπιστήμιο Πειραιώς – Τμήμα Πληροφορικής

Πρόγραμμα Μεταπτυχιακών Σπουδών

«Προηγμένα Συστήματα Πληροφορικής»

Μεταπτυχιακή Διατριβή

Τίτλος Διατριβής	**Ανάπτυξη ενός BYOD συστήματος απόκρισης κοινού με λειτουργίες εξουσιοδότησης και ταυτοποίησης** **Development of an Audience response system with authorization and authentication features**
Ονοματεπώνυμο Φοιτητή	**Ιωάννης Κουρμπέλης**
Πατρώνυμο	**Γεώργιος**
Αριθμός Μητρώου	**ΜΠΣΠ/ 17032**
Επιβλέπων	**Ευάγγελος Σακκόπουλος, Επίκουρος Καθηγητής**

Ημερομηνία Παράδοσης **Ιούλιος 2020**

Τριμελής Εξεταστική Επιτροπή

(υπογραφή)	(υπογραφή)	(υπογραφή)
Μαρία Βίρβου Καθηγήτρια	Ευθύμιος Αλέπης Αναπληρωτής Καθηγητής	Ευάγγελος Σακκόπουλος Επίκουρος Καθηγητής

Περιεχόμενα

Εικόνες

Παραδείγματα κώδικα

Περίληψη

Στην παρούσα εργασία παρουσιάζεται ένα web-based σύστημα απόκρισης κοινού (ARS) που παρέχει σύγχρονες υπηρεσίες επικοινωνίας. Για την ανάπτυξη του προτεινόμενου εργαλείου ARS χρησιμοποιήθηκαν αποκλειστικά τεχνολογίες ανοιχτού κώδικα, έτσι ώστε να μπορούν να υιοθετηθούν από οποιοδήποτε εκπαιδευτικό ίδρυμα χωρίς οικονομική προσπάθεια. Στην εφαρμογή οι φοιτητές χρησιμοποιούν τις δικές τους συσκευές smartphone. Το προτεινόμενο σύστημα στοχεύει να βοηθήσει την εκπαιδευτική διαδικασία ενισχύοντας τη συμμετοχή των μαθητών μέσω του μοντέλου Active Learning. Ο δάσκαλος μπορεί να θέσει ερωτήσεις στους μαθητές και να πάρει την ανταπόκρισή τους σε πραγματικό χρόνο. Το σύστημα εξασφαλίζει την ανωνυμία και δια-δραστικότητα των συμμετεχόντων και άμεσα και ακριβή δεδομένα, κατάλληλα για περαιτέρω στατιστική ανάλυση. Επιπλέον προσπαθεί να εμβαθύνει στην κάλυψη ζητημάτων ασφάλειας που εγείρονται με την χρήση Bring your own device (BYOD) πολιτικών προτείνοντας ένα μοντέλο ταυτοποίησης και εξουσιοδότησης τριών παραγόντων (3FA) .

Abstract

This postgraduate dissertation presents a web-based audience response system (ARS) that provides modern communication services. Only open source technologies were used to develop the proposed ARS tool, so that they could be adopted by any educational institution without financial effort. In the application, students use their own smartphone devices. The proposed system aims to help the educational process by enhancing student participation through the Active Learning model. The teacher can ask questions to the students and get their response in real time. The system ensures the anonymity and interactivity of the participants and direct and accurate data, suitable for further statistical analysis. In addition, it seeks to deepen the security issues raised by using Bring your own device (BYOD) policies by proposing a three-factor identification and authorization model (3FA)

Εισαγωγή – Σύντομη Περιγραφή

Αντικείμενο της παρούσας διατριβής είναι η δημιουργία ενός BYOD συστήματος απόκρισης κοινού το οποίο αρχικά να καλύπτει της ανάγκες του χρήστη και να επιτυγχάνει τους εκπαιδευτικούς και διδακτικούς σκοπούς για τους οποίους δημιουργήθηκε. Σε δεύτερη ανάλυση η εργασία προτείνει ένα μοντέλο ταυτοποίησης τριών παραγόντων για την απόκτηση πρόσβασης σε BYOD συστήματα με εφαρμογή τόσο σε εκπαιδευτικά συστήματα όπως το παρόν αλλά και σε enterprise mobility εφαρμογές .

Ποιο συγκεκριμένα το σύστημα εξυπηρετεί δύο τύπους χρηστών , τους καθηγητές και τους μαθητές. Οι καθηγητές μπορούν να δημιουργούν ερωτήσεις και polls και να στέλνουν σε πραγματικό χρόνο τα ερωτήματα που επιθυμούν στα poll που έχουν δημιουργήσει. Όλες οι ερωτήσεις που έχουν δημιουργηθεί αποτελούν μια κοινή βάση ερωτημάτων οι οποία δεν είναι αλληλένδετη με το εκάστοτε poll, με λίγα λόγια ο καθηγητής μπορεί να αναζητήσει και να επιλέξει οποιαδήποτε ερώτηση μέσα από την βάση ερωτημάτων και να την αποστείλει στο poll που δημιούργησε. Οι μαθητευόμενοι με την σειρά του μέσω της κινητής τους συσκευής αφού πρώτα εγγραφούν στο σύστημα ταυτοποίησης μπορούν να δουν τα διαθέσιμα ανοιχτά poll και να συνδεθούν σε αυτά. Εκεί λαμβάνουν σε πραγματικό χρόνο τα ερωτήματα και μπορούν να δώσουν την απάντηση τους. Τέλος ο καθηγητής μπορεί να δει μέσω διαγραμμάτων το ποσοστό επιτυχίας των φοιτητών σε κάθε ερώτημα. Παρακάτω παρουσιάζεται το διαγράμματα περιπτώσεων χρήσης του συστήματος για μαθητές και καθηγητές.

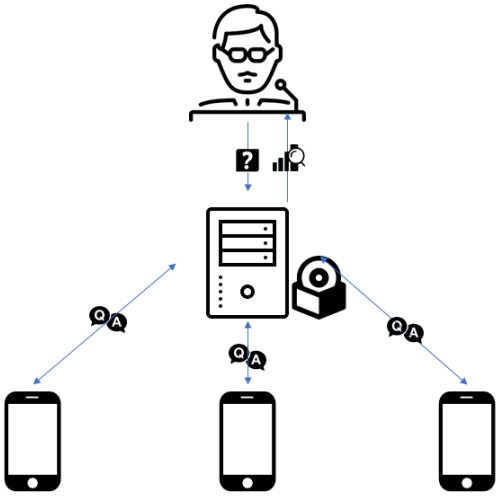
Η εργασία χωρίζεται σε πέντε κεφάλαια. Το πρώτο κεφάλαιο αφορά το θεωρητικό υπόβαθρο πίσω από τα συστήματα απόκρισης κοινού αλλά και τις πολιτικές διαχείρισης και ασφάλειας BYOD υλοποιήσεων. Το δεύτερο κεφάλαιο αναλύει τις τεχνολογίες που χρησιμοποιήθηκαν για την ανάπτυξη του συστήματος με μια πιο εκτενή αναφορά, που περιέχει και παραδείγματα κώδικα, στην .NET Core της οποίας το SDK χρησιμοποιήθηκε κατά κύριο λόγο για την υλοποίηση. Στο τρίτο κεφάλαιο γίνεται εκτενής παρουσίαση του συστήματος μέσω περιγραφής, εικόνων των σελίδων που εμφανίζονται στους χρήστες και παραδειγμάτων κώδικα. Το τέταρτο κεφάλαιο περιέχει αναλυτικές οδηγίες για την εγκατάσταση του συστήματος σε έναν IIS server. Το πέμπτο και τελευταίο κεφάλαιο παρουσιάζει τα τελικά συμπεράσματα αλλά και πιθανές μελλοντικές επεκτάσεις.

Κεφάλαιο 1.

Θεωρητικό υπόβαθρο

1.1. Συστήματα απόκρισης Κοινού

Τα συστήματα απόκρισης κοινού (ARS) ή clickers, όπως συνήθως ονομάζονται, είναι στην ουσία εργαλεία διαχείρισης της συμμετοχής των φοιτητών/μαθητών σε μια μεγάλη τάξη. (Caldwell, 2007). Χρησιμοποιούνται για να αλληλοεπιδρούν με τους μαθητές κατά τη διάρκεια της διάλεξης. Αποτελούνται από συσκευές εισόδου , κεντρικό διακομιστή καθώς και λογισμικό για την επικοινωνία και τον υπολογισμό των αποτελεσμάτων



2

Εικόνα 1.1. Η αρχιτεκτονική ενός ARS

Ο καθηγητής θέτει ερωτήσεις (συχνά ερωτήσεις πολλαπλής επιλογής) με το ARS και όλοι οι μαθητές μπορούν να απαντήσουν με το πάτημα ενός κουμπιού, που αντιπροσωπεύει την απάντησή τους, μέσω των συσκευών εισόδου. Ο κεντρικός εξυπηρετητής συλλέγει όλα τα αποτελέσματα και τα παρουσιάζει αμέσως στον καθηγητή (Caldwell, 2007) (Liu, 2003).

Τα ARS έχουν απασχολήσει ερευνητές από τη δεκαετία του 1960 (Judson, 2002). Οι πρώτοι πρόδρομοι των ARS δοκιμάστηκαν και αναπτύχθηκαν σε στρατιωτικά σενάρια (Froehlich, 1963) και κολέγια (Boardman, 1968) εκείνη την εποχή. Ωστόσο, δεδομένου ότι η τεχνολογία δεν ήταν τόσο αναπτυγμένη όσο σήμερα, αυτά τα ARSs ήταν πολύ περίπλοκα και δαπανηρά συστήματα. Για να χρησιμοποιηθεί ένα ARS, ολόκληρο το δωμάτιο πρέπει να είχε συνδεθεί, καθώς δεν υπήρχε άλλος τρόπος επικοινωνίας των συσκευών εισόδου. Ένα πολύ διάσημο ARS ήταν το Classtalk, το οποίο αναπτύχθηκε το 1996 (Dufresne, 1996). Αυτό το σύστημα διέθετε νέους τρόπους και δυνατότητες αλληλεπίδρασης με τους μαθητές κατά τη διάρκεια των διαλέξεων. Εκείνη την εποχή, εξελίχθηκαν οι διαφορετικές μορφές διδασκαλίας με ARS's. Τα ARS αλλάξαν,

από την άνοδο των ασύρματων τεχνολογιών (Ebner, 2009). Οι συσκευές εισόδου δεν συνδέονταν πλέον μέσω καλωδίου αλλά μέσω ασύρματων συνδέσεων και ονομάζονταν ασύρματες συσκευές Internet Learning (WILD) (Roschelle, 2003). Το σύστημα έγινε φορητό και μπορούσε να χρησιμοποιηθεί σε διαφορετικές αίθουσες διδασκαλίας αντί για μια μόνο. Τα τελευταία χρόνια, η χρήση κινητών συσκευών, όπως τα έξυπνα τηλέφωνα, τα tablet και οι φορητοί υπολογιστές, έχει αυξηθεί σημαντικά (Ebner, 2012). Οι μαθητές φέρνουν τις δικές τους συσκευές σε διαλέξεις , αυτή η πολιτική ονομάζεται Bring your own device - BYOD (Logicalis, 2012). Παρά τους κινδύνους του BYOD (Thomson, 2012) φαίνεται ότι είναι προφανές να χρησιμοποιούμε τις συσκευές των μαθητών ως συσκευές εισόδου για ARSs.

1.2. Η πολιτική BYOD

Η πολιτική BYOD είναι μια πολιτική που επιτρέπει στους υπαλλήλους να αξιοποιήσουν τις προσωπικές τους συσκευές τεχνολογίας, ώστε να συνδέονται ,να έχουν πρόσβαση στα δεδομένα ή να φέρουν εις πέρας τις εργασίες τους στην εταιρεία που εργάζονται. Συνοπτικά, τα προγράμματα BYOD επιτρέπουν στους χρήστες να έχουν πρόσβαση στις παρεχόμενες από τον εργοδότη υπηρεσίες ή / και δεδομένα μέσω των προσωπικών τους tablet smartphone η άλλων συσκευών (whitehouse.gov, 2012).

Η Intel κατάφερε να επιτύχει καλύτερη παραγωγικότητα, βελτιωμένη ασφάλεια και μεγαλύτερο έλεγχο. Ένα από τα επιχειρήματα υπέρ του BYOD είναι ότι προάγει την αποτελεσματική αξιοποίηση των πόρων καθώς οι υπολογιστές γραφείου παραμένουν αχρησιμοποίητοι όταν τελειώσουν οι ώρες γραφείου. Αυτή η ιδέα, που ονομάζεται επίσης 'consumerizarion of IT ',έγινε ευρέως αποδεκτή από τους εργοδότες και τους υπαλλήλους. Οι εργοδότες το θεωρούσαν αποτελεσματικό μέσο για την περικοπή του κόστους των υποδομών πληροφορικής και εξοπλισμού. Από την μεριά τους οι υπάλληλοι βρήκαν αυτή την ιδέα ελκυστική καθώς τους παρείχε την δυνατότητα χρήσης μιας τεχνολογίας της δικής τους επιλογής για την διεκπεραίωση των εργασιών τους. Τον Ιανουάριο του 2012, η Avanade κυκλοφόρησε τα αποτελέσματα μιας έρευνας στην οποία συμμετείχαν περισσότεροι από 600 επιχειρηματίες του χώρου της πληροφορικής σε 17 χώρες που έρχεται σε αντίθεση με τους κοινούς μύθους για τους παράγοντες που οδηγούν στην ιδέα του 'consumerizarion of IT '. Η μελέτη αυτή (Avanade Research &Insights, n.d.) ,δείχνει μια τεράστια μετατόπιση στη χρήση προσωπικών συσκευών στον χώρο εργασίας και ένα εκπληκτικό επίπεδο επένδυσης σε αυτή τη τάση από τους εργοδότες. Ακολουθεί μια σύντομη περίληψη αυτής της έκθεσης.

- Οι επιχειρήσεις δεν αντιστέκονται, αλλά αποδέχονται αυτή την αλλαγή.
- Οι επιχειρήσεις επενδύουν σε προσωπικό και πόρους ώστε να εφαρμόσουν αυτήν την αλλαγή.
- Όταν οι υπάλληλοι φέρνουν τις προσωπικές τους συσκευές στο χώρο εργασίας, τις χρησιμοποιούν για την πρόσβαση σε επιχειρηματικές εφαρμογές όπως CRM,ERP κ.λπ.
- Οι προσωπικές συσκευές που εισέρχονται στο χώρο εργασίας ποικίλουν έτσι δημιουργείτε η ανάγκη στους εργοδότες να σχεδιάζουν προσεκτικά την πολιτική χρήσης τους.
- Τέλος η πλειοψηφία των εταιρειών έχει ήδη αντιμετωπίσει μια παραβίαση ασφαλείας ως αποτέλεσμα του 'consumerization of IT'.

Η πολιτική BYOD έχει μικρό κόστος επένδυσης αλλά μακροπρόθεσμο λειτουργικό κόστος. Το BYOD δεν σημαίνει απλή άδεια στους εργαζόμενους να χρησιμοποιούν τις προσωπικές τους συσκευές ή τεχνολογίες για την πρόσβαση σε ευαίσθητους επιχειρηματικούς πόρους, σημαίνει επίσης ότι ο εργοδότης θα πρέπει να παρέχει κατάλληλη υποδομή για την ενσωμάτωση αποδοτικής λειτουργίας και επικοινωνίας μεταξύ διαφόρων τύπων συσκευών και τεχνολογιών στην υποδομή της επιχείρησης.

Υπάρχουν πολλά ζητήματα που πρέπει να επιλυθούν για τον σκοπό αυτόν. Πρέπει να ληφθούν υπόψιν διάφορες τεχνικές, νομικές, οικονομικές και άλλες πτυχές. Πρώτα απ' όλα, είναι μια γενική παρατήρηση ότι οι συσκευές που έχουν στην κατοχή τους οι υπάλληλοι είναι κατά κύριο λόγο πιο εξελιγμένες από τις συσκευές που ανήκουν στην εταιρεία. Ποικίλλουν ως προς τον τύπο, την τεχνολογία και πολλούς άλλους παράγοντες, ανάλογα με την άνεση των χρηστών να χρησιμοποιούν μια συσκευή. Συνεπώς, η τεχνική υποδομή του οργανισμού πρέπει να είναι αρκετά

καλή για να υποστηρίξει αυτή την πολυμορφία. Οι προσωπικές συσκευές περιέχουν δεδομένα, όπως φωτογραφίες, βίντεο, και άλλα ευαίσθητα δεδομένα. Όταν συνδέονται με το διακομιστή του οργανισμού, τα δεδομένα αυτά είναι προσβάσιμα από τον εργοδότη. Η ιδιοτικότητα των δεδομένων αυτών πρέπει να διατηρείται. Τα ζητήματα που αφορούν την απώλεια τέτοιων προσωπικών δεδομένων πρέπει να αντιμετωπίζονται δεόντως. Η απώλεια ευαίσθητων δεδομένων του εργοδότη είναι επίσης κρίσιμο ζήτημα. Τι θα συμβεί εάν ένας εργαζόμενος πωλήσει τη συσκευή του ή αποχωρήσει από την εταιρεία ή μοιραστεί τη συσκευή του με συγγενείς ή φίλους. Επιπλέον, ο εργοδότης μπορεί να εντοπίζει τη συμπεριφορά του εργαζομένου, τις αντιλήψεις και τις προτιμήσεις του από τις πληροφορίες που αποθηκεύονται στην προσωπική συσκευή του καθώς και να εντοπίζει τη θέση του/της χρησιμοποιώντας το εργαλείο εντοπισμού GPRS και άλλες εφαρμογές. Αυτό μπορεί να χρησιμοποιηθεί εναντίον των εργαζομένων. Οι προσωπικές συσκευές διαθέτουν επίσης συγκεκριμένο λογισμικό και προγράμματα, η χρήση των οποίων ενδέχεται να μην επιτρέπεται σε έναν οργανισμό. Επομένως, οι εργαζόμενοι πρέπει να εκπαιδεύονται σωστά. Εάν οι υπάλληλοι συνδέονται στο δίκτυο της εταιρείας μέσω των προσωπικών τους συσκευών, μπορούν να εργαστούν από οπουδήποτε, ακόμη και σε διακοπές. Σε τέτοιες περιπτώσεις, ο υπολογισμός του χρόνου εργασίας και των πληρωμών αποτελεί πρόβλημα. Έτσι, μια οργάνωση που υιοθετεί το σύστημα BYOD πρέπει να έχει μια άρτια πολιτική. Αυτό επιτυγχάνεται χρησιμοποιώντας οποιαδήποτε εφαρμογή Διαχείρισης κινητών συσκευών (MDM).

Υπάρχουν πολλοί πάροχοι πολιτικής MDM στην αγορά. Το AirWatch έχει αναδειχθεί ως ο κυριότερος ακολουθούν οι MobileIron, Citrix, Good technology και άλλες (surd, 2013). Είναι επίσης σημαντικό οι εργοδότες να οργανώσουν σεμινάρια κατάρτισης για τους υπαλλήλους σχετικά με πλεονεκτήματα και ζητήματα που μπορούν να αντιμετωπίσουν, με την υιοθέτηση του BYOD. Σύμφωνα με μια συνολική έρευνα που πραγματοποίησε ο OVUM (Adrian Drury, n.d.), το ποσοστό αποδοχής του BYOD στην Ινδία είναι μεγαλύτερο από 80% και η εφαρμογή του είναι περίπου 40%. Στις ΗΠΑ, το ποσοστό αποδοχής είναι 40% και η εφαρμογή της πολιτικής είναι περίπου 30%. Η έρευνα αυτή δείχνει ότι η έλλειψη μιας αποδοτικής πολιτικής BYOD αποτελεί ευρύτερο ζήτημα παγκοσμίως. Είναι επίσης ενδιαφέρον να σημειωθεί ότι η Ινδία εμφανίζει υψηλότερο ποσοστό αποδοχής σε σύγκριση με πιο ανεπτυγμένες και ώριμες αγορές.

1.3. Η πολιτική BYOD στην εκπαίδευση

Εδώ και μια δεκαετία τα εκπαιδευτικά ιδρύματα παρατηρούν την τάση μαθητές και καθηγητές να φέρουν μαζί τους φορητούς υπολογιστές, έξυπνα τηλέφωνα και tablet και εξετάζουν το πως αυτή η τάση μπορεί να χρησιμοποιηθεί ως πόρος για την ενίσχυση της μαθησιακής εμπειρίας. Σύμφωνα με τον Tom Murphy της Bradford Networks, «Τα εκπαιδευτικά ιδρύματα ήταν το προπύργιο της πολιτικής BYOD για χρόνια και κατά κάποιο τρόπο ήταν ένας προπομπός για τις επιχειρήσεις που έχουν αρχίσει να αποδέχονται την ελεύθερη επιλογή συσκευών εργασίας από την μεριά των υπαλλήλων". Σύμφωνα με έρευνα που διεξήχθη από την Educause to 2012, οι περισσότεροι φοιτητές (86%) διαθέτουν φορητούς υπολογιστές ως η κύρια συσκευή για ακαδημαϊκούς σκοπούς, αλλά όλο και περισσότερη σε σχέση με προηγούμενα έτη χρησιμοποιούν tablet (15%), έξυπνα τηλέφωνα (62%) ή/και ηλεκτρονικοί αναγνώστες (12%) (Dahlstrom, 2012). Μια άλλη έρευνα της Bradford Networks για εκπαιδευτικά ιδρύματα στις ΗΠΑ και το ΗΒ (Networks, n.d.) διαπίστωσαν ότι το 85% των ιδρυμάτων επιτρέπουν στους μαθητές και το προσωπικό να έχουν πρόσβαση στο σχολικό δίκτυο. Αυτό το ποσοστό αυξάνετε στην τριτοβάθμια εκπαίδευση (89%) και μειώνετε περίπου στο μισό στην πρωτοβάθμια και δευτεροβάθμια εκπαίδευση όπου είναι μόλις 44%. Επίσης βρέθηκε ότι οι συσκευές χρησιμοποιούνται τόσο για εκπαιδευτικούς σκοπούς όσο και για ιδιωτικούς σκοπούς: Το 78% ανέφερε ότι οι συσκευές χρησιμοποιούνται για προσωπική χρήση και Διεθνές ενώ το 72% δήλωσε ότι οι μαθητές χρησιμοποιεί τις συσκευές για να φέρει εις πέρας σχολικές εργασίες.

Οι έρευνες αυτές αντικατοπτρίζουν μια ευρεία αποδοχή του BYOD στην εκπαίδευση πόσο μάλλον σήμερα που τα παραπάνω ποσοστά έχουν αυξηθεί κατακόρυφα. Οι λόγοι για τους οποίους τα επίπεδα αποδοχής είναι τόσο υψηλά είναι ότι ο αυτοσκοπός της εκπαίδευσης είναι η παροχή γνώσεων που επιτυγχάνετε αποκτώντας όλο και περισσότερες πληροφορίες σχετικά με ένα συγκεκριμένο θέμα αντίθετα με τις επιχειρήσεις, όπου τα δεδομένα και οι πληροφορίες είναι

ευαίσθητα και δεν επιτρέπεται η πρόσβαση τους από κάποιο μη εξουσιοδοτημένο άτομο. Σήμερα το διαδίκτυο είναι η κύρια πηγή πληροφόρησης και η πληροφορία είναι διαθέσιμη σε αφθονία για κάθε θέμα υπό μορφή eBooks, ιστολόγιών και πολλών άλλων διαδικτυακών πόρων. Τα ιδρύματα μπορεί να έχουν συνδρομή σε ηλεκτρονικά περιοδικά και βιβλιοθήκες, η χρήση των οποίων παρέχετε δωρεάν στους φοιτητές. Με το BYOD οι φοιτητές μπορούν να έχουν εύκολη πρόσβαση σε αυτά από οπουδήποτε. Οι δάσκαλοι μπορούν να μοιραστούν τις γνώσεις τους εύκολα με τους μαθητές εντός ή εκτός της τάξης.Αυτή η πρακτική ωφελεί όλους τους σπουδαστές ανεξαρτήτως επιπέδου. Οι μαθητές με υψηλό επίπεδο IQ μπορούν να επιτύχουν εμπειρογνωμοσύνη στο θέμα που τους ενδιαφέρει σε σχετικά μικρή χρονική περίοδος. Εσωστρεφείς φοιτητές που διστάζουν να εκθέσουν τις απορίες τους στην αίθουσα διδασκαλίας, μπορούν να στείλουν email ή να δημοσιεύσουν τις απορίες αυτές προς τον καθηγητή τους είτε σε οποιοδήποτε διαδικτυακό φόρουμ.

Όπως και στη βιομηχανία, έτσι και στην εκπαίδευση υπάρχουν πολλές πολίτικες BYOD. Οι πολιτικές αυτές παρέχονται από εξειδικευμένους φορείς όπως η CISCO και άλλοι . Επίσης, υπάρχει πολλή εκπαίδευση καθοδήγηση και έχουν καθορίσει κατευθυντήριες γραμμές για το πώς να ενεργήσει ένα εκπαιδευτικό ίδρυμα που υποστηρίζει το BYOD. Ακολουθεί η διάταξη που παρέχεται από το Weevel για τα σχολεία.

		Action	Resource
REASEARCH	Step 1	• Build a small BYOD research team.	Colleagues who are interested
	Step 2	• Research what BYOD is, what it looks like in the classroom and what the issues are. • How have others schools implemented it? What were there challenges? • Which model of BYOD are schools using?	DEC 2013 Literature Review Internet research Twitter *#nswdecbyod* or *#BYOD* or *#BYOT* search
CONSULTATION	Step 3	• Survey for attitudes/type and quantity of devices • Develop own survey instrument or use all or part of existing survey tools • Interpret the data • Clarify the next steps, if BYOD is going ahead	Survey to key stakeholders: ✓ Staff ✓ Students ✓ parents/caregivers
	Step 4	• Hold a school staff meeting, P&C meeting, parent/caregiver/student forum after the surveys have been analysed	Present findings and conclusions to date.
POLICY DEVELOPMENT	Step 5	• Form a BYOD interest group	Representation from executive, staff. parents/caregivers (P&C) and, if appropriate, students (SRC for example)
	Step 6	• Develop a draft BYOD policy for the school	DEC 2013 Literature Review Other school policy documents
	Step 7	• Circulate the draft school policy for comment by the school community	Feedback form
	Step 8	• Develop the final version of the policy	BYOD interest group Advice from feedback form
	Step 9	• Communicate the school's BYOD policy to the school community	BYOD policy document and accompanying letter

Εικόνα 1.2. Εγχειρίδιο πολιτικής BYOD

Τα περισσότερα εκπαιδευτικά ιδρύματα έχουν επιτρέψει κάποια μορφή BYOD στην πανεπιστημιούπολή τους κυρίως μέσω ελέγχου πρόσβασης δικτύου (NAC) χωρίς εφαρμογή πολιτικής BYOD. Αυτό είναι πολύ επικίνδυνο καθώς (whitehouse.gov, 2012)εκθέτουν τα δίκτυά τους σε διάφορες απειλές, όπως μη εξουσιοδοτημένη πρόσβαση, επιθέσεις κακόβουλου λογισμικού και ιών από συσκευές μαθητών που είναι συνδεδεμένες στο δίκτυο των ιδρυμάτων. Τώρα, καθώς το BYOD αναδύεται ως μια ανεξάρτητη έννοια, θα καταστεί απαραίτητο για τα εκπαιδευτικά ιδρύματα να κατανοήσουν τι ακριβώς χρειάζεται να γίνει όταν αποφασιστεί να το χρησιμοποιήσουν.

1.4. Στρατηγική ασφάλειας BYOD

Όπως γίνεται αντιληπτό από την προηγούμενη υπό ενότητα είναι πολύ σημαντικό το ίδρυμα η ο οργανισμός που θα θελήσει να υιοθετήσει την χρήση του BYOD πρέπει να θεσπίσει μια άρτια πολιτική. Σημαντικό μέρος αυτής της πολιτικής είναι και η στρατηγική ασφάλειας που απαιτείται ώστε να διαχειριστεί τους κινδύνους που ελλοχεύουν με την χρήση προσωπικών συσκευών.

Οι πιο παραδοσιακές στρατηγικές ασφαλείας BYOD φαίνεται να επικεντρώνονται γύρω από δύο θέματα – τον περιορισμό της πρόσβασης και την εξ αποστάσεως εκκαθάριση δεδομένων. Οι κινητές συσκευές συχνά επιτρέπεται να συνδεθούν μόνο σε δίκτυο επισκέπτη. Ορισμένοι οργανισμοί αρκούνται στο να ορίζουν μια λίστα συσκευών οι οποία θεωρείται αξιόπιστη. Αυτή η στρατηγική περιορισμού χάνει έδαφος καθώς οι επιχειρήσεις όλο και περισσότερο στρέφονται στις cloud-based υπηρεσίες. Μια άλλη προσέγγιση είναι να διαγράφονται εξ αποστάσεως τα δεδομένα μιας συσκευής όταν θεωρηθεί αναξιόπιστη. Η απομακρυσμένη διαγραφή είναι μονόδρομος σε περιπτώσεις όπου οι λειτουργίες χωρίς σύνδεση είναι απαραίτητες και τα ευαίσθητα δεδομένα αποθηκεύονται τοπικά στις συσκευές. Ωστόσο, ένα μειονέκτημα αυτής της στρατηγικής είναι ότι μπορεί να διαγραφούν προσωπικά δεδομένα των εργαζομένων.

Είναι προφανές ότι για να είναι μια στρατηγική ασφάλειας BYOD κλιμακούμενη και οικονομικά προσιτή, απαιτείται μια πιο ρεαλιστική προσέγγιση - μια προσέγγιση που βασίζεται στην παροχή των σχετικών δεδομένων στους αρμόδιους ανθρώπους, όπου και αν βρίσκονται και όποτε τα χρειάζονται. Πώς λοιπόν οι εταιρείες παραδίδουν δεδομένα σήμερα; Μέσω των API . Εάν ένας οργανισμός παρέχει τα δεδομένα του μέσω API, τότε τα δεδομένα δεν βρίσκονται στην πραγματικότητα στην κινητή συσκευή. Αντίθετα, είναι προσβάσιμα από ελαφριές mobile εφαρμογές που εκτελούνται στη συσκευή. Εάν η συσκευή χαθεί ή ο χρήστης δεν έχει πλέον δικαιώματα πρόσβασης στα δεδομένα, η ομάδα ασφαλείας IT μπορεί απλά να απενεργοποιήσει την πρόσβαση και να καταστήσει τη συσκευή ακίνδυνη. Το αποτέλεσμα είναι ότι ένας οργανισμός μπορεί πλέον να χρησιμοποιεί API για να περιορίζει την πρόσβαση σε δεδομένα μόνο σε εξουσιοδοτημένα άτομα και τις κινητές συσκευές τους, και ως αποτέλεσμα, να εγγυάται την ασφάλεια μιας πολιτικής BYOD σε επίπεδο επιχείρησης.

Ας ρίξουμε μια ματιά στο τι είναι το API, καθώς μπορεί να σημαίνει διαφορετικά πράγματα για τους διάφορους επαγγελματίες, ανάλογα με την προγραμματιστική τους εμπειρία. Όταν μιλάμε σήμερα για API, αναφερόμαστε σε Representational State Transfer (REST) API's, ένα συγκεκριμένο στυλ web API. Αυτού του είδους τα API's χρησιμοποιούνται πλέον ευρέως για ενσωμάτωση, ειδικά για την ενεργοποίηση εφαρμογών για φορητές συσκευές. Στην πραγματικότητα, είναι ο τρόπος επικοινωνίας μεταξύ εφαρμογών κινητών συσκευών και διακομιστών για την ανταλλαγή δεδομένων. Οι εφαρμογές για φορητές συσκευές είναι ελαφριοί clients που κάνουν μικρή τοπική επεξεργασία. Σε ένα εταιρικό πλαίσιο, αυτές οι φορητές εφαρμογές πρέπει να συνδέονται τακτικά με συστήματα cloud ή εσωτερικής εγκατάστασης για να έχουν πρόσβαση σε δεδομένα και να ολοκληρώνουν την εκτέλεση εμπορικών συναλλαγών η άλλων συναλλαγών που μπορεί να αφορούν μια επιχείρηση. Ο χρήστης κινητής τηλεφωνίας μπορεί να βρίσκεται οπουδήποτε, ανά πάσα στιγμή, και επομένως ο CSO πρέπει να είναι σε θέση να εφαρμόζει πολιτικές που διασφαλίζουν ασφαλή και κλιμακούμενη πρόσβαση στα δεδομένα. Η εφαρμογή αυτών των πολιτικών σε επίπεδο API είναι ευκολότερη και πολύ πιο οικονομική από ό,τι σε επίπεδο συσκευής.

1.5. API και ασφάλεια BYOD

1.5.1. Εισαγωγή

Είναι σαφές ότι οι API διαδραματίζουν κρίσιμο ρόλο στην επιτυχία μιας στρατηγικής ασφάλειας BYOD. Ωστόσο, πρέπει να γίνεται σωστή διαχείριση ώστε να επιτυγχάνονται βέλτιστα αποτελέσματα. Μια προσέγγιση που βασίζεται στον έλεγχο ταυτότητας του χρήστη και την εξουσιοδότηση είναι ο πιο αποτελεσματικός τρόπος διαχείρισης των API. Μια αποτελεσματική λίστα ελέγχου ταυτότητας για τη διαχείριση ενός API αποτελείται από τις παρακάτω πρακτικές:

1.5.2. Εξουσιοδότηση και έλεγχος ταυτότητας

Η εξουσιοδότηση πρέπει να καλύπτει μια ολόκληρη γκάμα σεναρίων, από τους παραδοσιακούς κανόνες αδειοδότησης (ποιος εργαζόμενος μπορεί να κάνει τι), μέσω του ελέγχου πρόσβασης βάσει ρόλων έως την εξουσιοδότηση βάσει περιεχομένου, όπου τα API μπορούν να προσαρμόζουν δυναμικά την πρόσβαση σε δεδομένα και λειτουργίες βασιζόμενα σε παράγοντες όπως ο τύπος εφαρμογής, η γεωγραφική θέση της συσκευής, το δίκτυο, η ώρα της ημέρας και γενικότερα η συμπεριφορά πρόσβασης. Η προσθήκη ελέγχου ταυτότητας πολλών παραγόντων (multi-factor authentication) μπορεί να αυξήσει περαιτέρω το επίπεδο αξιοπιστίας της συσκευής και του χρήστη.

1.5.3. Κανόνες περιορισμού

Εφαρμόζοντας κανόνες περιορισμού της κυκλοφορίας ως μέρος μιας πολιτικής, μια επιχείρηση μπορεί να εντοπίσει εάν ένας συγκεκριμένος client κάνει κατάχρηση του δικαιώματος πρόσβασης ή των επιπέδων χρήσης των API. Για παράδειγμα, μια πολιτική διαχείρισης API θα μπορούσε να υπαγορεύσει ότι σε ένα συγκεκριμένο API θα πρέπει να έχουν πρόσβαση μόνο συγκεκριμένοι φορητοί clients με προκαθορισμένα όρια κυκλοφορίας και κανόνες ασφαλείας βάσει ταυτότητας.

1.5.4, Σωστή διαχείριση API

Ένας οργανισμός πρέπει να μπορεί να εντοπίζει και να αναφέρει αμέσως τυχόν ασυνήθιστα μοτίβα χρήσης. Ας πάρουμε για παράδειγμα έναν χρήστη κινητής συσκευής που συνήθως έχει πρόσβαση σε ορισμένες λειτουργίες API σε ένα συγκεκριμένο μοτίβο. Εάν η συμπεριφορά τους αλλάξει ξαφνικά -για παράδειγμα, ως αποτέλεσμα κλοπής- τότε πρέπει να σταλεί μια ειδοποίηση στην ομάδα ασφαλείας του οργανισμού.

1.5.5. Εφαρμογή στρατηγικής διαχείρισης

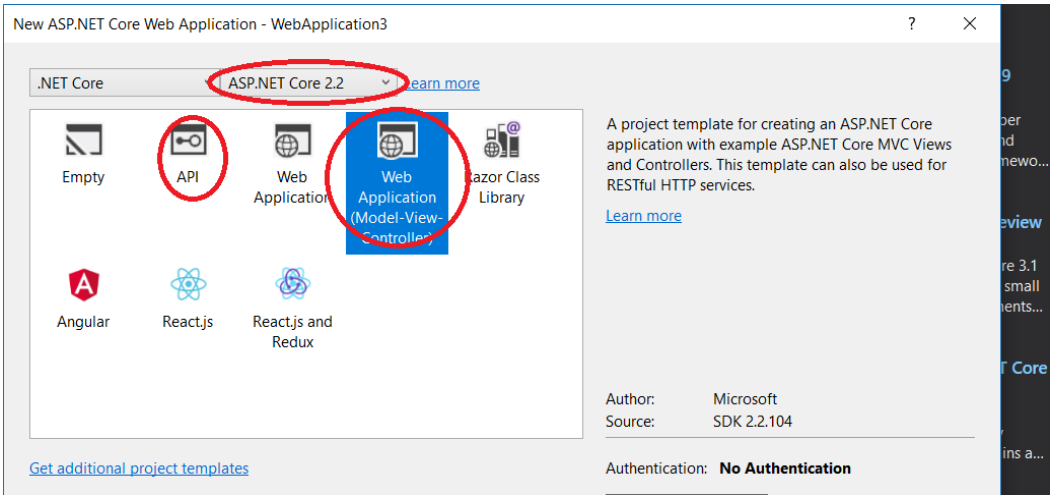
Μια καλή στρατηγική διαχείρισης API θα επιτρέψει την επιτυχή ασφάλεια. Υπάρχουν διάφορες επιλογές για την εφαρμογή μιας στρατηγικής διαχείρισης API. Ένας οργανισμός μπορεί να δημιουργήσει δυνατότητες ελέγχου ταυτότητας και διαχείρισης της κυκλοφορίας στα API. Εναλλακτικά, θα μπορούσε να αξιοποιήσει προϊόντα πλατφόρμας που βοηθούν στη διαχείριση των API. Αυτά τα προϊόντα διαχείρισης API διαθέτουν χαρακτηριστικά που περιλαμβάνουν τη δυνατότητα επιτάχυνσης της κυκλοφορίας και μπορούν επίσης να διασφαλίσουν την ασφάλεια, καθώς οι client πρέπει να χρησιμοποιούν ένα συγκεκριμένο token ασφάλειας, όπως ένα κλειδί API ή ένα OAuth token. Επιπλέον, τα προϊόντα διαχείρισης API παρέχουν δυνατότητες παρακολούθησης της χρήσης και εντοπισμού ασυνήθιστης δραστηριότητας κυκλοφορίας.

Κεφάλαιο 2.

Τεχνικές, πρότυπα και εργαλεία ανάπτυξης

2.1. Περίληψη κεφαλαίου

Σε αυτό το κεφάλαιο αναλύονται οι τεχνολογίες που χρησιμοποιήθηκαν για την ανάπτυξη του συστήματος. Παρουσιάζονται επιγραμματικά τα τέσσερα σκέλη τα οποία αποτελούν το σύστημα με μια σύντομη περιγραφή του ρόλου τους και αναφορά στις τεχνολογίες που χρησιμοποιήθηκαν για την υλοποίηση τους. Στην συνέχεια θα γίνει μια πιο εκτενής αναφορά σε αυτές τις τεχνολογίες και τέλος μια σύντομη παρουσίαση του προγραμματισμού σε .Net Core του οποίου το SDK είναι και το κύριο εργαλείο προγραμματισμού που χρησιμοποιήθηκε. Το IDE στο οποίο έλαβε χώρα ο προγραμματισμός του συνόλου του συστήματος είναι το Visual Studio 2017.



Εικόνα 2.1. ASP.NET Core 2.2 Web Application template

2.2. Τα σκέλη του συστήματος

2.2.1. Web API

Το Web API είναι το σκέλος που συνδέεται με την κεντρική βάση του συστήματος και ικανοποιεί τα HTTP Requests από τα υπόλοιπα σκέλη. Αποτελεί τον Resource Server του συστήματος. Αναπτύχθηκε την χρήση του .net core 2.2 SDK και συγκεκριμένα με το template 'API' που περιέχει έναν πρότυπο Controller που ικανοποιεί τις ανάγκες ενός RESTful HTTP Service.

2.2.2. MVC Client

Ο MVC Client είναι το στρώμα που παρεμβάλλεται μεταξύ του Web API και τον υπολοίπων δύο σκελών του συστήματος. Κατά κάποιο τρόπο αποτελεί το Front End του συστήματος. Όπως και το Web API έτσι και ο MVC Client αναπτύχθηκε σε .net core 2.2 SDK με την χρήση του template Web Application (Model-View-Controller) το οποίο είναι δομημένο πάνω στην αρχιτεκτονική MVC.

2.2.3. Identity Server

Πρόκειται για τον authorization Server του συστήματος που συνδέεται με την βάση που είναι αποθηκευμένα τα στοιχεία των χρηστών. Παρέχει εξουσιοδότηση και ταυτοποίηση στο σύστημα και είναι μια υλοποίηση του IdentityServer4, ενός Open Id Connect και Oauth2 framework για την ASP.NET Core.

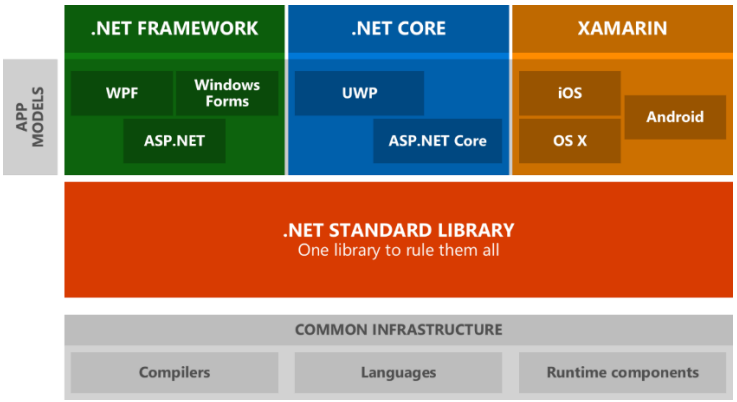
2.2.4. Mobile Εφαρμογή

Είναι η mobile εφαρμογή με την οποία εγγράφονται και συνδέονται οι χρήστες/μαθητές στο σύστημα. Αναπτύχθηκε με την χρήση του Xamarin Android.

2.3. Η πλατφόρμα .Net

2.3.1. Εισαγωγή

Η .Net είναι μια προγραμματιστική πλατφόρμα δηλαδή ένα σύνολο από γλώσσες προγραμματισμού (C#,F#,Visual Basic), εργαλεία και βιβλιοθήκες το οποίο μπορεί να χρησιμοποιηθεί για την ανάπτυξη desktop, web, mobile, gaming και iot εφαρμογών. Η αρχική υλοποίηση της .Net είναι το .NET Framework. Το .NET Framework υποστηρίζει την ανάπτυξη web και desktop εφαρμογών καθώς και services αποκλειστικά σε περιβάλλον Windows.



Εικόνα 2.2. Η πλατφόρμα .Net

Μετέπειτα δημιουργήθηκαν άλλες υλοποιήσεις όπως το Xamarin, το οποίο είναι μια υλοποίηση που υποστηρίζει την δημιουργία και εκτέλεση mobile εφαρμογών σε όλα τα διαδεδομένα mobile λειτουργικά συστήματα όπως το IOS και Android, και τέλος η .Net Core η οποία είναι μια open

source και cross platform (Windows, MacOs, Linux) υλοποίηση της .Net για την δημιουργία web,services και εφαρμογών κονσόλας. Κοινός παρονομαστής των υλοποιήσεων που αναφέρθηκαν είναι το πρότυπο .NET (.NET Standard) το οποίο είναι μια τυπική προδιαγραφή των API που είναι κοινά μεταξύ των υλοποιήσεων και επιτρέπει την χρήση κοινού κώδικα σε μορφή βιβλιοθηκών μεταξύ αυτών. Παρακάτω θα αναφερθούμε σε κάθε μια υλοποίηση ξεχωριστά δίνοντας ιδιαίτερη έμφαση στην .Net Core η οποία ήταν και η υλοποίηση που χρησιμοποιήθηκε κατ' εξοχήν για την δημιουργία του συστήματος

2.3.2. .Net Framework

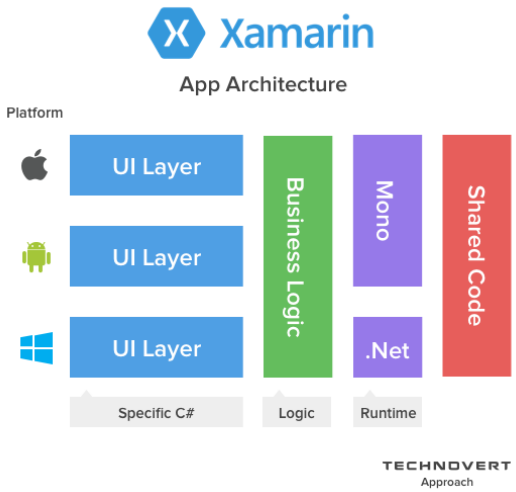
Το .NET Framework υποστηρίζει περισσότερες από 60 γλώσσες προγραμματισμού με κυριότερες την C# και την Visual Basic. Τα κύρια χαρακτηριστικά συστατικά του στοιχεία είναι τα παρακάτω :

Common Language Runtime(CLR): Το CLR είναι το βασικό συστατικό στοιχείο του .NET Framework. Είναι το περιβάλλον εκτέλεσης του .NET Framework που εκτελεί τον κώδικα και βοηθά στη διευκόλυνση της ανάπτυξης, παρέχοντας τις διάφορες υπηρεσίες όπως remoting, διαχείριση thread, διαχείριση μνήμης κλπ. Βασικά, είναι υπεύθυνο για τη διαχείριση της εκτέλεσης προγραμμάτων .NET ανεξάρτητα από οποιαδήποτε γλώσσα προγραμματισμού .NET.

Framework Class Library(FCL): Είναι η συλλογή επαναχρησιμοποιήσιμων, και αντικειμενοστραφών βιβλιοθηκών κλάσεων και μεθόδων που μπορούν να ενσωματωθούν με το CLR. Είναι ακριβώς όπως τα αρχεία κεφαλίδας στις C / C ++ και τα πακέτα στη Java. Η εγκατάσταση του .NET Framework είναι βασικά η εγκατάσταση του CLR και του FCL στο σύστημα.

2.3.3. Xamarin

Το Xamarin επεκτείνει την πλατφόρμα .NET με εργαλεία και βιβλιοθήκες ειδικά για την κατασκευή εφαρμογών σε iOS, Android, macOS και πολλά άλλα. Επί της ουσίας ο προγραμματιστής μπορεί να γράψει έναν κοινό κώδικα για διαφορετικές πλατφόρμες. Σε επίπεδο root, το Xamarin έχει μετατρέψει ολόκληρα τα υπάρχοντα SDK των Android και iOS σε C #. Όσον αφορά το UI παραμένει λίγο πολύ το ίδιο. Δημιουργείτε ξεχωριστά για διαφορετικές πλατφόρμες και στη συνέχεια συνδέεται με τον βασικό κώδικα. Στην παρακάτω εικόνα φαίνεται η αρχιτεκτονική μιας εφαρμογής Xamarin.



Εικόνα 2.3. Xamarin App Architecture

2.3.4. .Net Standard

Πρόκειται για το σύνολο των θεμελιωδών API (κοινώς αναφερόμενο ως βιβλιοθήκη κλάσεων βάσης ή BCL) που όλες οι εφαρμογές .NET πρέπει να εφαρμόσουν. Ουσιαστικά το πρότυπο .NET είναι μια προδιαγραφή. Κάθε έκδοση .NET Standard ορίζει το σύνολο των API που πρέπει να παρέχουν όλες οι εφαρμογές .NET για να συμμορφώνονται με αυτήν την έκδοση. Μπορεί να θεωρηθεί ως ένα ακόμα .NET stack, με την διαφορά ότι δεν μπορούν να δημιουργηθούν εφαρμογές για αυτό, παρά μόνο βιβλιοθήκες, οι οποίες εν συνεχεία μπορούν να χρησιμοποιηθούν από όλες τις υπόλοιπες υλοποιήσεις. Όλα τα stack της .NET εφαρμόζουν κάποια έκδοση του .NET Standard. Ο κανόνας είναι ότι όταν παράγεται μια νέα υλοποίηση .NET, θα χρησιμοποιήσει συνήθως την πιο πρόσφατη διαθέσιμη έκδοση του προτύπου .NET.

2.3.5. .NET Core

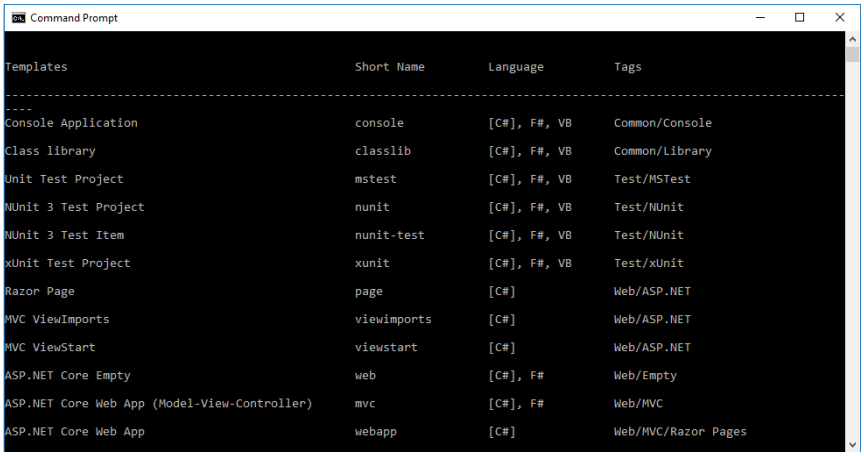
Η .NET Core είναι μια σχετικά νέα cross-platform και εξολοκλήρου open-source υλοποίηση της .NET που είναι διακριτά διαφοροποιημένη από το .NET Framework και το Silverlight και είναι βελτιστοποιημένη για εργασίες που εκτελούνται σε κινητά και διακομιστές. Η .NET Core έχει το δικό της command line interface(CLI), που ονομάζεται .NET Core CLI και έχει την δυνατότητα να καλύψει σχεδόν όλες τις πτυχές της ανάπτυξης (create, build, testing, packaging).

Παρακάτω θα δούμε κάποια παραδείγματα κώδικα ξεκινώντας με την δημιουργία και εκτέλεση μιας απλής Hello Word εφαρμογής στην συνέχεια θα ενσωματώσουμε λογική στην εφαρμογή και τέλος θα δημιουργήσουμε μια web εφαρμογή. Μπορούμε να χρησιμοποιήσουμε το command line των Windows η αντίστοιχα το Bash για MacOs η Linux.

```
C#

$ dotnet new console -o hello
$ cd hello
$ dotnet run
Hello World!
```

Εικόνα 2.4. Δημιουργία Hello World

Η εντολή “dotnet new” στο CLI είναι αντίστοιχη με την δημιουργία ενός νέου project στο Visual Studio. Πατώντας την μπορούμε να δούμε όλα τα διαθέσιμα project που μπορούμε να δημιουργήσουμε



Εικόνα 2.5,. Διαθέσιμα Project

Στη συνέχεια επιλέγοντας Class Library θα δούμε πως μπορούμε να δημιουργήσουμε λογική σε μια class library και να την ενσωματώσουμε στην console εφαρμογή που φτιάξαμε

```
C:\Users\Giannis>dotnet new classlib -o logic
The template "Class library" was created successfully.

Processing post-creation actions...
Running 'dotnet restore' on logic\logic.csproj...
  Restoring packages for C:\Users\Giannis\logic\logic.csproj...
  Generating MSBuild file C:\Users\Giannis\logic\obj\logic.csproj.nuget.g.props.
  Generating MSBuild file C:\Users\Giannis\logic\obj\logic.csproj.nuget.g.targets.
  Restore completed in 156.94 ms for C:\Users\Giannis\logic\logic.csproj.

Restore succeeded.
```

Εικόνα 2.6. Επιτυχής δημιουργία Class Library

Αλλάζουμε τα περιεχόμενα του αρχείου Class1.cs με τον παρακάτω κώδικα

```
logic.HelloWorld
using System;

namespace logic
{
    public static class HelloWorld
    {
        public static string GetMessage(string name) => $"Hello {name}!";
    }
}
```

Εικόνα 2.7. Class1.cs

Μετονομάζουμε το αρχείο Class1.cs σε HelloWorld.cs

```
C:\Users\Giannis\logic>mv Class1.cs HelloWorld.cs

C:\Users\Giannis\logic>
```

Εικόνα 2.8.Μετονομασία Class1.cs

Προσθέτουμε σαν reference την βιβλιοθήκη που φτιάξαμε στην console εφαρμογή και στη συνέχεια ενημερώνουμε το αρχείο Program.cs της εφαρμογής ώστε να χρησιμοποιήσει την βιβλιοθήκη

```
C:\Users\Giannis>cd hello

C:\Users\Giannis\hello>dotnet add reference ../logic/logic.csproj
Reference `..\logic\logic.csproj` added to the project.

C:\Users\Giannis\hello>
```

Εικόνα 2.9. Προσθήκη reference

hello.Program

```
using System;
using logic;
namespace hello
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.Write("What's your name: ");
            var name = Console.ReadLine();
            var message = HelloWorld.GetMessage(name);
            Console.WriteLine(message);
        }
    }
}
```

Εικόνα 2.10. Ενημέρωση Program.cs

Τρέχουμε την εφαρμογή και βλέπουμε τα αποτελέσματα

```
C:\Users\Giannis\hello>dotnet run
What's your name: Giannis
Hello Giannis!
```

Εικόνα 2.11. Αποτελέσματα

Παρακάτω θα δούμε πως μπορούμε να δημιουργούμε μια απλή ASP .NET Core ιστοσελίδα και να χρησιμοποιήσουμε την βιβλιοθήκη που φτιάξαμε πριν.

Επιλέγουμε το template “ASP.NET Core Empty”

```
C:\Users\Giannis>dotnet new web -o web
The template "ASP.NET Core Empty" was created successfully.

Processing post-creation actions...
Running 'dotnet restore' on web\web.csproj...
  Restoring packages for C:\Users\Giannis\web\web.csproj...
  Generating MSBuild file C:\Users\Giannis\web\obj\web.csproj.nuget.g.props.
  Generating MSBuild file C:\Users\Giannis\web\obj\web.csproj.nuget.g.targets.
  Restore completed in 6.08 sec for C:\Users\Giannis\web\web.csproj.

Restore succeeded.


C:\Users\Giannis>cd web

C:\Users\Giannis\web>dotnet add reference ../logic/logic.csproj
Reference `..\logic\logic.csproj` added to the project.

C:\Users\Giannis\web>
```

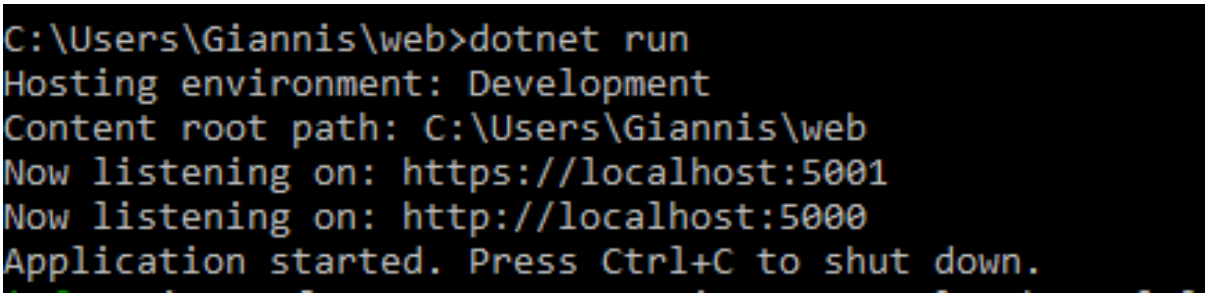
Εικόνα 2.12. Επιτυχής Δημιουργία

Αλλάζουμε το αρχείο Startup.cs για να χρησιμοποιήσει την βιβλιοθήκη που φτιάξαμε πριν :

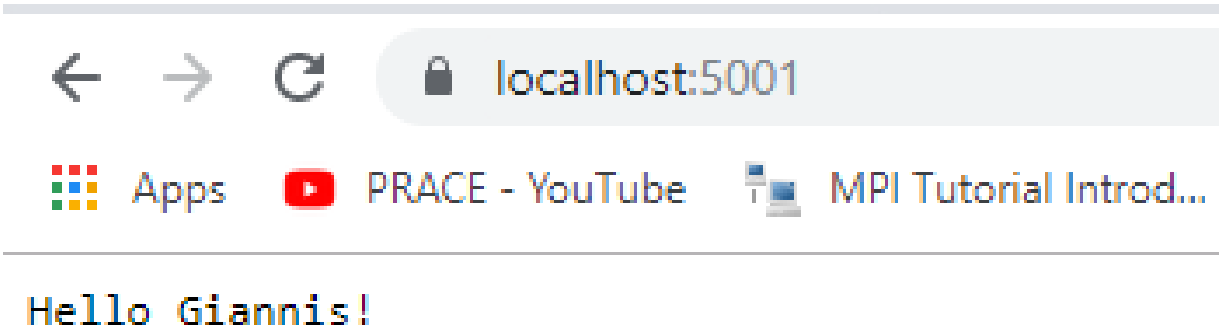
```
using Microsoft.AspNetCore.Builder;
using Microsoft.AspNetCore.Hosting;
using Microsoft.AspNetCore.Http;
using Microsoft.Extensions.DependencyInjection;

namespace web
{
    public class Startup
    {
        // This method gets called by the runtime. Use this method to add services to the container.
        // For more information on how to configure your application, visit https://go.microsoft.com/fwlink/?LinkID=398940
        public void ConfigureServices(IServiceCollection services)
        {
        }

        // This method gets called by the runtime. Use this method to configure the HTTP request pipeline.
        public void Configure(IApplicationBuilder app, IHostingEnvironment env)
        {
            if (env.IsDevelopment())
            {
                app.UseDeveloperExceptionPage();
            }

            app.Run(async (context) =>
            {
              var name = Environment.UserName;
              var message = logic.HelloWorld.GetMessage(name);
              await context.Response.WriteAsync(message);
            });
        }
    }
}
```

Εικόνα 2.13. Αλλαγή Strartup.cs

Τρέχουμε την εφαρμογή και προηγούμαστε στον Browser για να δούμε τα αποτελέσματα



Εικόνα 2.14. Εκτέλεση εφαρμογής



Εικόνα 2.15, Αποτελέσματα Browser

Κεφάλαιο 3.

Η ανάπτυξη του συστήματος

3.1. Περίληψη κεφαλαίου

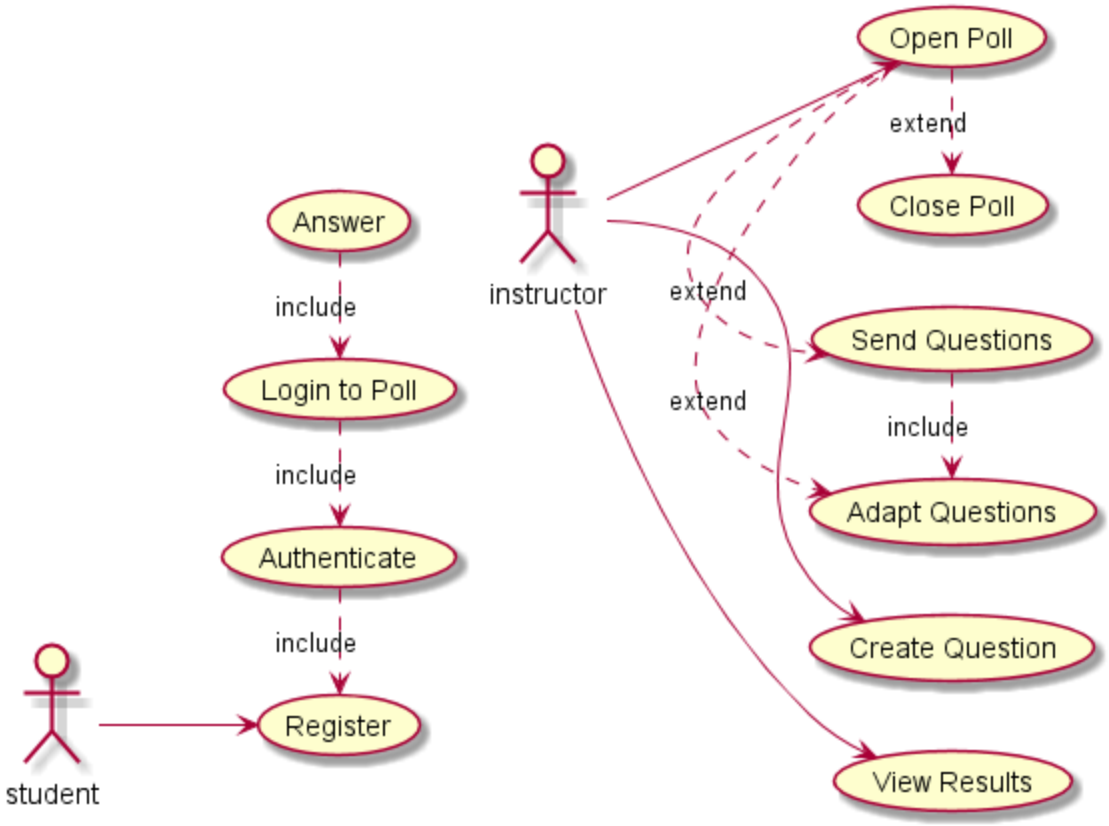
Το κεφάλαιο αυτό ξεκινάει με μια σύντομη περιγραφή των περιπτώσεων χρήσης του συστήματος. Συνεχίζει με την περιγραφή της αρχιτεκτονικής του συστήματος, πως δηλαδή τα σκέλη που αναφέρθηκαν στο προηγούμενο κεφάλαιο δένουν μεταξύ τους και συνθέτουν το τελικό σύστημα. Τέλος θα αναλύσουμε την ανάπτυξη του κάθε σκέλους ξεχωριστά με παραδείγματα κώδικα, εικόνες από το interface και αναφορά στο θεωρητικό υπόβαθρο των τεχνικών που χρησιμοποιήθηκαν.

3.2. Σύντομη περιγραφή

Το σύστημα που υλοποιήθηκε αποτελεί ένα ARS βασισμένο στην BYOD φιλοσοφία. Οι τύποι χρηστών του συστήματος είναι δύο :

1. Καθηγητές
2. Μαθητές/φοιτητές

Οι καθηγητές μπορούν να δημιουργούν ερωτήσεις και polls και να στέλνουν σε πραγματικό χρόνο τα ερωτήματα που επιθυμούν στα poll που έχουν δημιουργήσει. Όλες οι ερωτήσεις που έχουν δημιουργηθεί αποτελούν μια κοινή βάση ερωτημάτων οι οποία δεν είναι αλληλένδετη με το εκάστοτε poll, με λίγα λόγια ο καθηγητής μπορεί να αναζητήσει και να επιλέξει οποιαδήποτε ερώτηση μέσα από την βάση ερωτημάτων και να την αποστείλει στο poll που δημιούργησε. Οι μαθητευόμενοι με την σειρά του μέσω της κινητής τους συσκευής αφού πρώτα εγγραφούν στο σύστημα ταυτοποίησης μπορούν να δουν τα διαθέσιμα ανοιχτά poll και να συνδεθούν σε αυτά. Εκεί λαμβάνουν σε πραγματικό χρόνο τα ερωτήματα και μπορούν να δώσουν την απάντηση τους. Τέλος ο καθηγητής μπορεί να δει μέσω διαγραμμάτων το ποσοστό επιτυχίας των φοιτητών σε κάθε ερώτημα. Παρακάτω παρουσιάζεται το διαγράμματα περιπτώσεων χρήσης του συστήματος για μαθητές και καθηγητές.



Εικόνα 3.1 Διάγραμμα περιπτώσεων χρήσης

3.3. Η αρχιτεκτονική του συστήματος

Στο προηγούμενο κεφάλαιο αναφέραμε συνοπτικά τα τέσσερα σκέλη που αποτελούν το σύστημα (Web Api, Mvc Client, Identity Server, Mobile App) και τον ρόλο που επιτελούν. Σε αυτήν την υπό ενότητα θα δούμε εν συντομία πως αυτά τα σκέλη δένουν μεταξύ του και συνθέτουν το τελικό σύστημα. Ξεκινώντας από το Web API να αναφέρουμε ότι πρόκειται για ένα service layer που καλύπτει το data layer του συστήματος δηλαδή συνδέεται με την βάση του συστήματος και εξυπηρετεί requests από τα υπόλοιπα σκέλη. Για παράδειγμα όταν δημιουργείτε μία ερώτηση μέσω του MVC Client στέλνεται ένα POST request στο αντίστοιχο endpoint του Web API και αποθηκεύεται η ερώτηση στην βάση, ομοίως ένα GET request όταν ζητείται μια ερώτηση.

Το Web API επιδρά κατά κύριο λόγο με τον MVC Client που είναι επί της ουσίας το front end του συστήματος και διαχειρίζεται τα διάφορα request.Έτσι στον MVC Client γίνονται οι διαδικασίες δημιουργίας ερωτήσεων, poll, προβολή αποτελεσμάτων και άλλα σε επίπεδο χρήστη. Μέσω του MVC Client και της mobile εφαρμογής γίνεται η διαδικασία εγγραφής του φοιτητή στον Identity Server και με ένα GET request η mobile εφαρμογή τραβάει τα διαθέσιμα ανοιχτά poll από το API και εν συνεχεία μπορεί να συνδεθεί στο επιλεγμένο poll μεταβαίνοντας μέσω του browser της κινητής συσκευής πρώτα στον identity server για να γίνει ταυτοποίηση ο οποίος μετά κάνει redirect τον χρήστη στον MVC Client όπου τελικά μπορεί να δει και να απαντήσει τις ερωτήσεις. Στις παρακάτω ενότητες θα δούμε το κάθε σκέλος ξεχωριστά περιγράφοντας τον ρόλο και τον τρόπο ανάπτυξης τους και βλέποντας παραδείγματα κώδικα.

3.4. Το Web API

Όπως ήδη έχει γραφτεί το Web API καταναλώνει τα HTTP requests των υπολοίπων σκελών της εφαρμογής. Πρόκειται για ένα σύνολο από controllers οι οποίοι πρακτικά ορίζουν τα endpoint του Web API και domain classes οι οποίες με την βοήθεια του Entity Framework Core συνθέτουν την βάση του συστήματος.

```
namespace AudienceResponseAPI.Models
{
    public class Questions
    {
        [Key]
        public int Qid { get; set; }
        public string Question { get; set; }
        public string C1 { get; set; }
        public string C2 { get; set; }
        public string C3 { get; set; }
        public string C4 { get; set; }
        public string C5 { get; set; }
        public int Right { get; set; }


    }
}
```

Παράδειγμα κώδικα 3.1. Questions Model

Το Entity Framework Core είναι ένα ORM(Object Relational Mapper) Framework το οποίο υποστηρίζεται από την Microsoft και δίνει την δυνατότητα στους προγραμματιστές να δουλέψουν με μια βάση δεδομένων χρησιμοποιώντας αντικείμενα .Net. Συγκεκριμένα περιέχει βιβλιοθήκες οι οποίες μπορούν κατά κύριο λόγο να :

1.Συσχετίσουν μια domain class με ένα database object

```
namespace AudienceResponseAPI.Data
{
    public class ApplicationDBContext : DbContext
    {
        public ApplicationDBContext(DbContextOptions<ApplicationDBContext> options)
        : base(options)
        {
        }
        public DbSet<VoteSession> VoteSession { get; set; }
        public DbSet<ConnectedUsers> ConnectedUsers { get; set; }
        public DbSet<Questions> Questions { get; set; }
        public DbSet<QuestionAdapter> QuestionAdapter { get; set; }
        public DbSet<Answer> Answer { get; set; }
        public DbSet<User> User { get; set; }

    }
}
```

Παράδειγμα κώδικα 3.2. DbContext

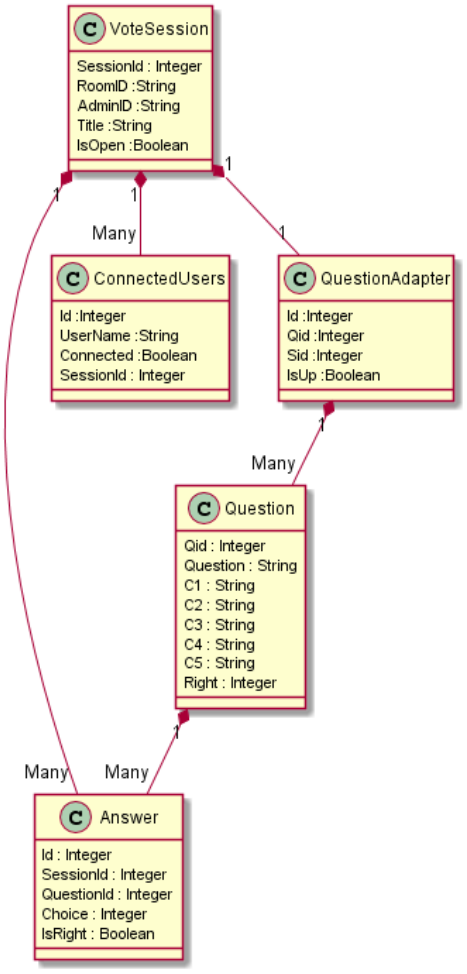
2.Να δημιουργήσουν μια βάση δεδομένων και να εναρμονίσουν το σχήμα της σε συνάρτηση με τις αλλαγές που γίνονται στα μοντέλα (domain classes)

```
using Microsoft.EntityFrameworkCore.Metadata;
using Microsoft.EntityFrameworkCore.Migrations;
using System;
using System.Collections.Generic;

namespace AudienceResponseAPI.Migrations
{
    public partial class initial : Migration
    {
        protected override void Up(MigrationBuilder migrationBuilder)
        {
            migrationBuilder.CreateTable(
                name: "Vo
                columns: 
                {
                    SessionId = table.Column<int>(nullable: false)
                        .Annotation("SqlServer:ValueGenerationStrategy", SqlServerValueGenerati
                    IsOpen = table.Column<bool>(nullable: false),
                    MinutesAlive = table.Column<int>(nullable: false),
                    RoomID = table.Column<string>(maxLength: 4, nullable: true)
                },
                constraints: table =>
                {
                    table.PrimaryKey("PK_VoteSession", x => x.SessionId);
                });
```

(parameter) MigrationBuilder migrationBuilder

Παράδειγμα κώδικα 3.3, Migrations

Παρακάτω βλέπουμε το διάγραμμα κλάσεων των μοντέλων του Web API και την συσχέτιση μεταξύ τους.



Εικόνα 3.2. Διάγραμμα κλάσεων

3.4.1. Models

Όπως βλέπουμε στο σχήμα το Web API αποτελείται από πέντε domain classes η οποίες σχηματίζουν και τους αντίστοιχους πίνακες στην βάση δεδομένων. Η βασική κλάση στην οποία καταλήγουν όλες η υπόλοιπες είναι η VoteSession η οποία αντιπροσωπεύει ένα poll. Η VoteSession αποτελείται από πέντε properties :

- **SessionID** : Το κλειδί του πίνακα.
- **RoomID** : Ο αναγνωριστικός κωδικός του poll.
- **AdminID** : Το κλειδί του χρήστη που ανοίγει το poll και απαιτείται τις για διάφορες εργασίες που το αφορούν.
- **Title** : Η περιγραφή του poll.
- **IsOpen** : Boolean μεταβλητή η οποία δηλώνει εάν το poll είναι ανοιχτό ή έχει κλείσει.

Η επόμενη κλάση που είναι ιδιαίτερης σημασίας είναι η QuestionAdapter. Πρόκειται επί της ουσίας για έναν «καθρέφτη» μιας ερώτησης ο οποίος κάνει την σύνδεση μιας ερώτησης με ένα poll. Αυτός ο σχεδιασμός επιτρέπει στον χρήστη να μπορεί να επιλέξει από όλες τις ερωτήσεις και όχι να δημιουργεί ερωτήσεις για ένα συγκεκριμένο poll μόνο. Έτσι όταν ο χρήστης επιλέγει να εισάγει μια ερώτηση στο poll αυτόματα δημιουργείται μια εγγραφή στον πίνακα QuestionAdapter που περιέχει της παρακάτω στήλες :

- **Id** : Το κλειδί του πίνακα
- **Qid** : Το κλειδί της ερώτησης που δηλώνει ποια ερώτηση αφορά ο QuestionAdapter.
- **Sid**: Το κλειδί του Poll.
- **IsUp**: Boolean μεταβλητή που δηλώνει αν η ερώτηση που έχει συνδεθεί με τον QuestionAdapter είναι η τρέχουσα ερώτηση που βλέπουν οι μαθητές.

Οι υπόλοιπες τρείς κλάσεις είναι οι ερωτήσεις (Questions) , οι απαντήσεις και οι συνδεδεμένοι χρήστες.

3.4.2. Controllers

Κατά κανόνα ένας controller διαχειρίζεται τα HTTP Request για μια domain class.Για παράδειγμα ο QuestionsController ορίζει το endpoint των ερωτήσεων και διαχειρίζεται τα requests για αυτό το μοντέλο όπως φαίνεται από τον παρακάτω κώδικα :

```
namespace AudienceResponseAPI.Controllers
{
//Επιστροφή των μοντέλων σε Json
[Produces("application/json")]
//Ορισμός του endpoint
[Route("api/Questions")]
public class QuestionsController : Controller
{
  private readonly ApplicationDBContext _context;

//Dependency Injection του ApplicationDBContextDbContext(Σύνολο των database o
bjects)
  public QuestionsController(ApplicationDBContext context)
  {
    _context = context;
  }
 //GET: api/Questions (Πχ. www.example.com/api/Questions)
  [HttpGet]
```

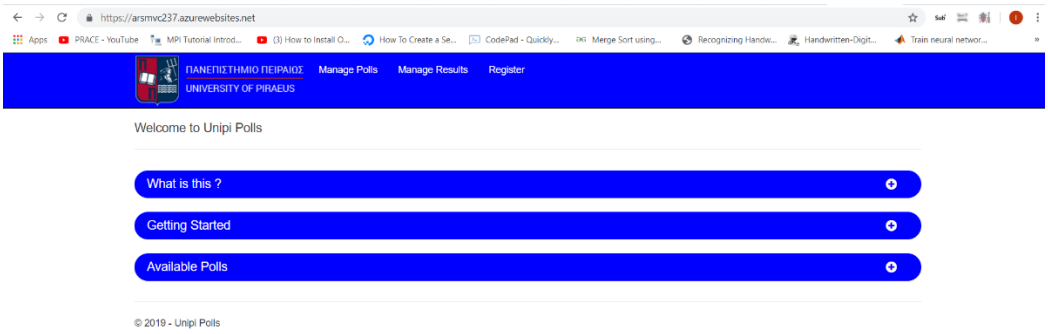
```
public IEnumerable<Questions> GetQuestions()
{
  //Επιστροφή όλων των ερωτήσεν σε Json
  return _context.Questions;
}
```

Παράδειγμα κώδικα 3.4, Questions Controller

3.5, MVC Client

3.5.1. Εισαγωγή

Αυτό το σκέλος δεν είναι τίποτα παραπάνω από μια συνηθισμένη web εφαρμογή. Μπαίνοντάς κάποιος χρήστης σε αυτήν την εφαρμογή θα δει την παρακάτω αρχική σελίδα :



Εικόνα 3.3. Αρχική σελίδα

Η σελίδα αυτή πέρα τον διαφόρων πληροφοριακών στοιχείων έχει τρία navigation link, δύο από τα οποία αφορούν τον καθηγητή (Manage Polls, Manage Results) και ένα το οποίο αφορά τον μαθητή (Register) και θα ασχοληθούμε με αυτό στην ενότητα που περιγράφει την διαδικασία εγγραφής στον Identity Server.

3.5.2. Manage Polls

Σε αυτή την σελίδα ο χρήστης μπορεί να προχωρήσει στις παρακάτω ενέργειες

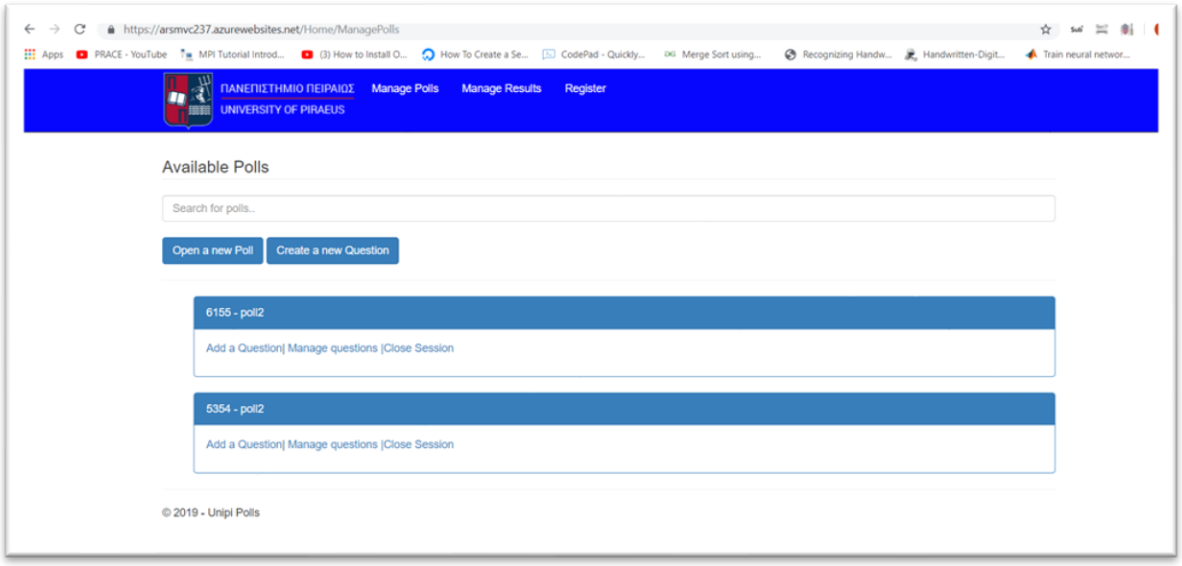
- Να δει τα ανοιχτά poll
- Να ψάξει στην λίστα των ανοιχτών poll
- Να δημιουργήσει ένα καινούργιο poll
- Να δημιουργήσει μια καινούργια ερώτηση

Αναφορικά με τα ανοιχτά poll ο χρήστης μπορεί :

- Να προσθέσει μια ερώτηση στο poll
- Να στείλει τις ερωτήσεις που έχουν προστεθεί στους χρήστες που είναι συνδεδεμένοι
- Να κλείσει το poll

3.5.3. Δημιουργία Polls

Ξεκινώντας με την δημιουργία ενός νέου poll ο καθηγητής πρέπει να εισάγει ένα AdminID το οποίο πρόκειται για έναν πενταψήφιο αριθμό ο οποίος ζητείται να εισαχθεί πριν ο χρήστης προβεί σε ενέργειες που αφορούν το συγκεκριμένο poll (προσθήκη ερώτησης, κλείσιμο κλπ.), επίσης μπορεί να βάλει μια σύντομη περιγραφή για το poll. Πατώντας δημιουργία αυτόματα δημιουργείται και ένας τετραψήφιος τυχαίος αριθμός που αποτελεί αναγνωριστικό του poll.



Εικόνα 3.4. Ανοιχτά poll

Στο backend του προγράμματος η διαδικασία έχει ως εξής :

1.Συλλέγονται τα Id των ανοιχτών poll

```
HttpClient client = _api.Initial();
List<VoteSession> OpenSessions = await _reuse.GetOpenSessionsAsync();
List<string> rids = new List<string>();
foreach (var item in OpenSessions)
{
  if (item.IsOpen == true)
 {
    rids.Add(item.RoomID);
 }
}
```

Παράδειγμα κώδικα 3.5. Ανοιχτά poll

2.Δημιουργείτε ένας τυχαίος τετραψήφιος αριθμός που διαφέρει από αυτούς των ανοιχτών poll

```
Random _random = new Random();
string randrid = _random.Next(0, 9999).ToString("D4");
while (rids.Contains(randrid))
{
  randrid = _random.Next(0, 9999).ToString("D4");
}
```

Παράδειγμα κώδικα 3.6. Δημιουργία RoomID

3.Δημιουργείται ένα αντικείμενο VoteSession (poll) ,το οποίο παίρνει το AdminID και την περιγραφή από το View Model (σελίδα δημιουργίας του poll) το τυχαίο poll id και την σηματοδότηση ότι το poll είναι ανοιχτό.

```
VoteSession NewSession = new VoteSession()
{
  IsOpen = true,
  RoomID = randrid,
  AdminID = vmodel.AdminID,
  Title = vmodel.Title
};
```

Παράδειγμα κώδικα 3.7. Ορισμός Στοιχείων Poll

4.Το αντικείμενο μετασχηματίζεται σε ένα json object και στην συνέχεια πραγματοποιείται ένα post request στο αντίστοιχο endpoint to Web Api.

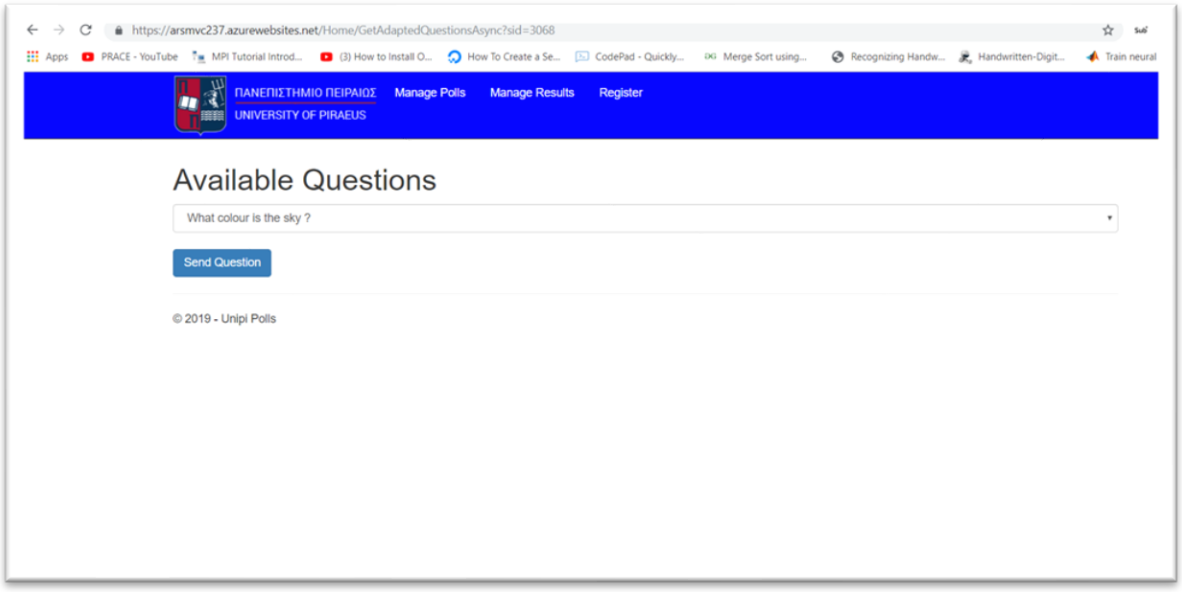
```
string json = JsonConvert.SerializeObject(NewSession, Formatting.Indented);
var httpContent = new StringContent(json, Encoding.UTF8, "application/json");
HttpResponseMessage res = await client.PostAsync("api/VoteSessions", httpContent);
if (res.IsSuccessStatusCode)
{
  HttpClient client2 = _api.Initial();
  HttpResponseMessage res2 = await client.GetAsync("api/VoteSessions");
  if (res2.IsSuccessStatusCode)
  {
    var result = res2.Content.ReadAsStringAsync().Result;
    List<VoteSession> sessionList = new List<VoteSession>();
    SessionList = JsonConvert.DeserializeObject<List<VoteSession>>(result);
   }

}
```

Παράδειγμα κώδικα 3.8. Post στο API

Τέλος μέσω του Web Api αποθηκεύεται το poll στην βάση δεδομένων του συστήματος. Παρόμοια λογική ακολουθούν και οι διαδικασίες δημιουργίας ερωτήσεων, κλεισίματος poll και προσάρτησης ερωτήσεων. Περισσότερο θα ασχοληθούμε με την διαδικασία αποστολής ερωτήσεων στο poll και τη προβολή του σε πραγματικό χρόνο από τους φοιτητές.

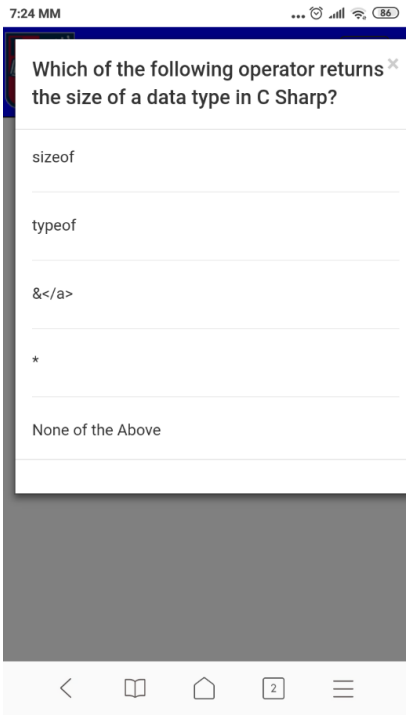
3.5.4. Αποστολή ερωτήσεων

Για να στείλει μια ερώτηση ο καθηγητής πρέπει πρώτα να προσαρτήσει ερωτήσεις στο poll που έχει δημιουργήσει είτε δημιουργώντας καινούργιες είτε επιλέγοντας από τις ήδη υπάρχουσες μέσω του συνδέσμου Add a question. Μπαίνοντας στον σύνδεσμο Manage Questions ενός poll αφού πρώτα εισάγει το Admin Id ο καθηγητής βλέπει μια λίστα με τις ερωτήσεις που έχει ήδη προσαρτήσει στο poll.



Εικόνα 3.5. Αποστολή ερώτησης

Επιλέγοντας μία ερώτηση και πατώντας Send Question η ερώτηση εμφανίζεται σε πραγματικό χρόνο στους φοιτητές που έχουν συνδεθεί στο συγκεκριμένο poll μέσω της κινητής τους συσκευής , διαδικασία την οποία θα περιγράψουμε σε επόμενη ενότητα. Η αποστολή και προβολή της ερώτησης σε πραγματικό χρόνο επιτυγχάνεται μέσω της χρήσης της open source βιβλιοθήκης ASP.NET Core SignalR η οποία προσθέτει λειτουργικότητα πραγματικού χρόνου σε Web εφαρμογές. Πριν προχωρήσουμε σε περιγραφή της υλοποίησης θα εξηγήσουμε πως λειτουργεί η SignalR.

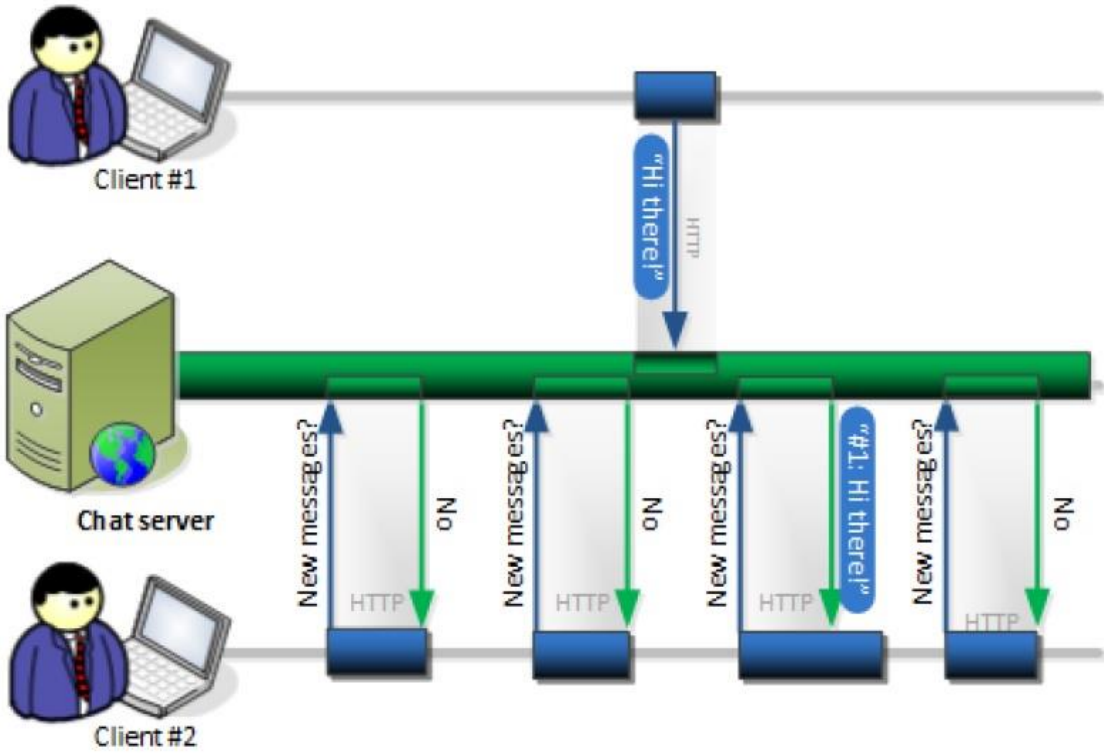


Εικόνα 3.6. Εμφάνιση ερώτησης στον clicker

Για την δημιουργία μιας real time εφαρμογής όπως για παράδειγμα μια εφαρμογή online voting παρόμοια με το σύστημα μας πρέπει να σκεφτούμε πως θα ανακτήσουμε την τελευταία ερώτηση που έχει σταλεί. Παραδοσιακά το id της ερώτησης και το id του poll θα μπορούσαν να αποθηκεύονται σε έναν πίνακα της βάσης του web service μαζί με ένα αναγνωριστικό που να υποδηλώνει ότι είναι η τελευταία ερώτηση που έχει σταλεί για το συγκεκριμένο poll και με έναν timer μέσω του client κάθε ένα συγκεκριμένο χρονικό διάστημα να καλείται ένα request στο web service που να ζητάει να επιστραφούν τα στοιχεία της τελευταίας ερώτησης του poll και μετά να εμφανίζονται αυτά στον χρήστη. Αυτή η προσέγγιση (periodic polling) μπορεί να δημιουργήσει τα εξής προβλήματα :

- Αύξηση της κίνησης του δικτύου
- Η σύνδεση HTTP μεταξύ του client και του Server ξανά εγκαθιδρύεται με κάθε request.
- Υπάρχουν καθυστερήσεις λόγω του χρόνου συγκέντρωσης των request (pooling time period)

Σε γενική ανάλυση πολλά request γίνονται άσκοπα χωρίς να υπάρχει κάποιο response από τον Server όπως φαίνεται στο παρακάτω παράδειγμα.



Εικόνα 3.7. Periodic polling schema

Με την Signal R δημιουργείτε μια συνεχή σύνδεση στην αρχή της επικοινωνίας οι οποία παραμένει και στέλνει δεδομένα στον client όταν γίνει κάποια ενημέρωση (πχ. Αποστολή ερώτησης).Για την επίτευξη της επικοινωνίας σε πραγματικό χρόνο η Signal R χρησιμοποιεί τις παρακάτω μεθόδους :

- WebSocket
- Server Sent Events
- Long Polling

Η Signal R θα επιλέξει την καλύτερη μέθοδο μεταφοράς ανάλογα με τις δυνατότητες του Server και του Client.

3.5.5. Web Sockets

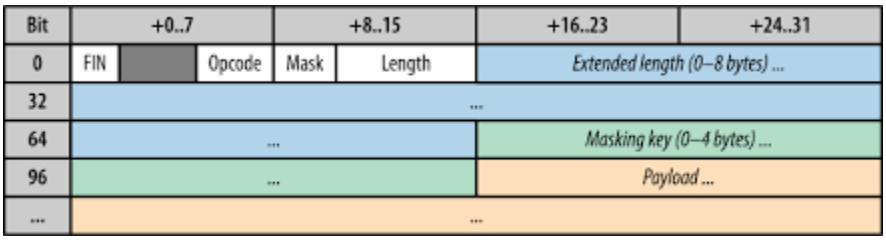
Τα WebSocket έχουν σαν σκοπό να επιτύχουν μια συνεχή σύνδεση μεταξύ Client και Server ώστε και τα δύο μέρη να μπορούν να στείλουν και να λάβουν δεδομένα ανά πάσα στιγμή. Η σύνδεση αυτή επιτυγχάνεται μέσω μιας διαδικασίας γνωστής και ως WebSocket Handsake. Η διαδικασία ξεκινά με τον Client να στέλνει ένα κοινό HTTP Request στον Server . Σε αυτό το Request περιέχετε μια κεφαλίδα που ονομάζεται “Upgrade” η οποία ενημερώνει τον Server ότι ο Client επιθυμεί να εγκαθιδρύσει μια σύνδεση WebSocket. Παρακάτω βλέπουμε απλοποιημένο παράδειγμα των αρχικών κεφαλίδων του Request.

```
GET ws://websocket.example.com/ HTTP/1.1

Origin: http://example.com

Connection: Upgrade

Host: websocket.example.com

Upgrade: websocket
```

Σε περίπτωση που ο Server υποστηρίζει το πρωτόκολλο WebSocket συμφωνεί και το επικοινωνεί στο Client στέλνοντας επίσης ένα Upgrade Header στο Response όπως φαίνεται παρακάτω :

```
HTTP/1.1 101 WebSocket Protocol Handshake

Date: Wed, 16 Oct 2013 10:07:34 GMT

Connection: Upgrade

Upgrade: WebSocket
```

Τώρα που έχει ολοκληρωθεί το Handshake η αρχική σύνδεση HTTP αντικαθιστάτε με μια σύνδεση WebSocket. Με τα WebSocket μπορεί να γίνει απεριόριστη μεταφορά δεδομένων χωρίς την επιβάρυνση overhead των κοινών HTTP Request. Τα δεδομένα μεταφέρονται μέσω των WebSocket σαν μηνύματα, το καθένα από τα οποία αποτελείται από ένα η περισσότερα κομμάτια που περιέχουν τα δεδομένα που στέλνονται(payload). Για να εξασφαλιστεί η ανακατασκευή του μηνύματος όταν φτάσει στον Client κάθε κομμάτι περιέχει μόνο 4-12 bytes που αφορούν το payload. Χρησιμοποιώντας αυτό το σύστημα το οποίο είναι βασισμένο σε κομμάτια για την αποστολή του payload μειώνεται η ποσότητα των δεδομένων που δεν αποτελούν κομμάτι του payload έτσι μειώνεται σημαντικά ο χρόνος καθυστέρησης (latency).



Εικόνα 3.8. Web Socket Message

Παρακάτω θα δούμε ένα απλό παράδειγμα WebSocket :

HTML

```
<div id="page-wrapper">
  <h1>WebSockets Demo</h1>
  <div id="status">Connecting...</div>
  <ul id="messages"></ul>
  <form id="message-form" action="#" method="post">
<textarea id="message" placeholder="Write your message here..." required></textarea>
    <button type="submit">Send Message</button>
    <button type="button" id="close">Close Connection</button>
  </form>111
</div>
```

Παράδειγμα κώδικα 3.9. Web Socket front end

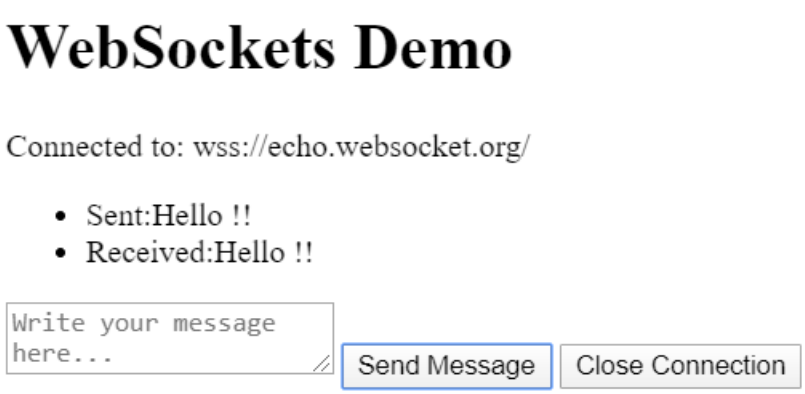
JAVASCRIPT

```
window.onload = function() {
  // Αναφορές στα elements της σελίδας.
  var form = document.getElementById('message-form');
  var messageField = document.getElementById('message');
  var messagesList = document.getElementById('messages');
  var socketStatus = document.getElementById('status');
  var closeBtn = document.getElementById('close');
  // Δημιουργία νέου WebSocket.
  var socket = new WebSocket('wss://echo.websocket.org');
  // Χειρισμός λαθών.
  socket.onerror = function(error) {
```

```
    console.log('WebSocket Error: ' + error);
  };
  //Μήνυμα επιτυχής σύνδεσης.
  socket.onopen = function(event) {
    socketStatus.innerHTML = 'Connected to: ' + event.currentTarget.url;
    socketStatus.className = 'open';
  };
  // Χειρισμός των μηνυμάτων που έρχονται από τον Server.
  socket.onmessage = function(event) {
    var message = event.data;
    messagesList.innerHTML += '<li class="received"><span>Received:</span>' +
message + '</li>';
  };
  // Μήνυμα αποσύνδεσης του WebSocket.
  socket.onclose = function(event) {
    socketStatus.innerHTML = 'Disconnected from WebSocket.';
    socketStatus.className = 'closed';
  };
  // Αποστολή μηνύματος όταν η φόρμα γίνεται Submit.
  form.onsubmit = function(e)
     e.preventDefault();
    // Απόκτηση του μηνύματος από την textarea.
    var message = messageField.value;
    // Αποστολή του μηνύματος μέσω του WebSocket.
    socket.send(message);
    // Προσθήκη του μηνύματος στην λίστα μηνυμάτων.
    messagesList.innerHTML += '<li class="sent"><span>Sent:</span>' + message +
'</li>';
    // Καθαρισμός του πεδίου.
    messageField.value = '';
    return false;
  };
 // Κλείσιμο του WebSocket με το πάτημα κουμπιού
closeBtn.onclick = function(e) {
```

```
    e.preventDefault();
    socket.close();
    return false;
  };
};
```

Παράδειγμα κώδικα 3.10. Web Sockets JS code



Εικόνα 3.9. Web Socket running example

3.5.6. Server sent events

Το SSE (Server Sent Events) είναι ένα πρότυπο που περιγράφει τον τρόπο με τον οποίο οι διακομιστές μπορούν να εκκινήσουν τη μετάδοση δεδομένων προς τους clients μόλις δημιουργηθεί μια αρχική σύνδεση client. Παρέχει μια αποδοτική εφαρμογή της ροής XHR. Σε αντίθεση με μια ακατέργαστη σύνδεση XHR, η οποία απομνημονεύει την πλήρως ληφθείσα απόκριση μέχρι να πέσει η σύνδεση, μια σύνδεση SSE μπορεί να απορρίψει τα επεξεργασμένα μηνύματα χωρίς να τα συσσωρεύσει στη μνήμη.

Το SSE έχει σχεδιαστεί για να χρησιμοποιεί το JavaScript API EventSource για να εγγραφεί σε μια ροή δεδομένων σε οποιοδήποτε δημοφιλές πρόγραμμα περιήγησης. Μέσω αυτής της διασύνδεσης, ένας client ζητά μια συγκεκριμένη διεύθυνση URL για να λάβει μια ροή συμβάντων. Το SSE χρησιμοποιείται συνήθως για την αποστολή ενημερωμένων μηνυμάτων ή συνεχών ροών δεδομένων σε ένα πρόγραμμα-πελάτη προγράμματος περιήγησης. Με λίγα λόγια, ένα server sent event είναι όταν οι ενημερώσεις ωθούνται (αντί να τραβηχτούν ή ζητηθούν) από ένα διακομιστή σε ένα πρόγραμμα περιήγησης.

Μια σύνδεση μέσω SSE αρχίζει συνήθως με την επικοινωνία που ξεκίνησε από τον client μεταξύ του client και του διακομιστή. Ο client δημιουργεί ένα νέο αντικείμενο JavaScript EventSource, μεταφέροντας τη διεύθυνση URL ενός endpoint στον διακομιστή μέσω ενός κανονικού αιτήματος HTTP. Ο client αναμένει μια απάντηση με μια ροή μηνυμάτων συμβάντων με την πάροδο του χρόνου. Ο διακομιστής αφήνει ανοικτή την απόκριση HTTP έως ότου δεν έχει πλέον να στείλει συμβάντα, αποφασίσει ότι η σύνδεση έχει ανοιχτεί αρκετά και μπορεί να θεωρηθεί καθυστερημένη ή έως ότου ο πελάτης κλείσει ρητά το αρχικό αίτημα. Παρακάτω θα δούμε ένα

παράδειγμα στο οποίο ο διακομιστής στέλνει την ώρα του μηχανήματος και ο client λαμβάνει και εμφανίζει το μήνυμα

PHP

```php
<?php
header('Content-Type: text/event-stream');
header('Cache-Control: no-cache'); // recommended to prevent caching of event data.

/**
* Constructs the SSE data format and flushes that data to the client.
*
* @param string $id id of this connection.
* @param string $message Line of text that should be transmitted.
*/
function sendMsg($id, $message ) {
echo "id: $id" . PHP_EOL;
echo "data: $message" . PHP_EOL;
echo PHP_EOL;
ob_flush();
flush(); //don't forget to flush so the server sends it out
}


//NO Need to LOOP this message as browser will re-connect every 3 seconds
sendMsg( time() , 'server time: ' . date("h:i:s", time()));

```

Παράδειγμα κώδικα 3.11. SSE PHP code

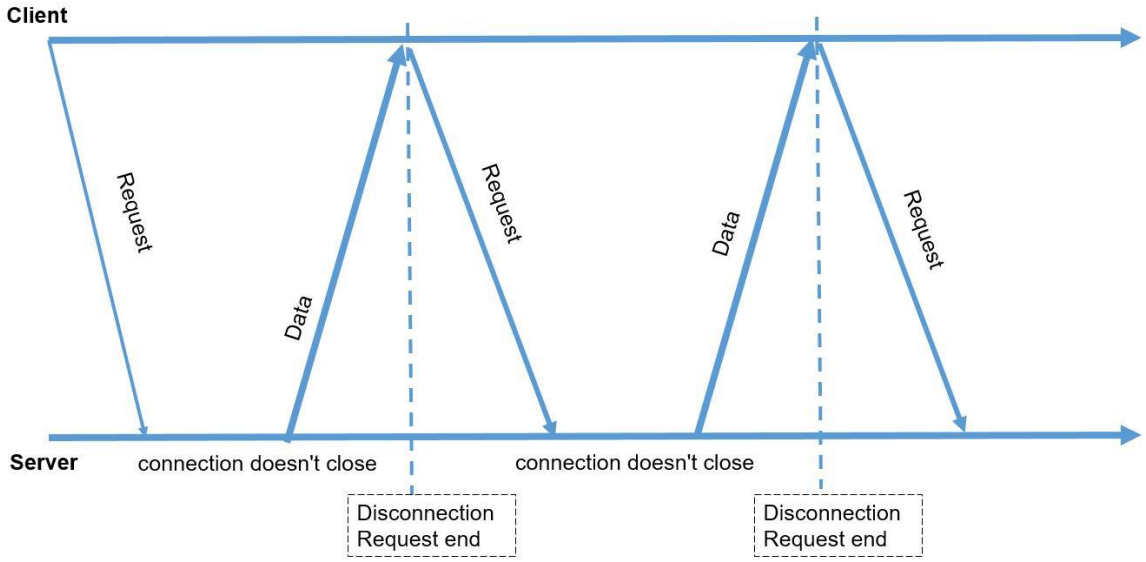
JAVASCRIPT

```
<script>
if(typeof(EventSource) !== "undefined") {
  var source = new EventSource("demo_sse.php");
  source.onmessage = function(event) {
    document.getElementById("result").innerHTML += event.data + "<br>";
  };
} else {
  document.getElementById("result").innerHTML = "Sorry, your browser does not support server-sent events...";
}</script>
```

Παράδειγμα κώδικα 3.12, SSE HTML/ JS code

3.5.7. Long polling

Η μέθοδος αυτή βασίζεται στην παραδοσιακή μέθοδο που αναφέραμε αρχικά (regular polling) με την διαφορά ότι αντί να στέλνονται request στον client ανά τακτά χρονικά διαστήματα στέλνεται ένα request και η σύνδεση με τον Server παραμένει ανοικτή μέχρι ο server να στείλει απάντηση και μετά άμεσα επαναλαμβάνεται ένα νέο request. Εάν η σύνδεση χαθεί λόγω κάποιου σφάλματος δικτύου ο browser αμέσως στέλνει ένα νέο request.



Εικόνα 3.10. Long polling schema

3.5.8. Signal R hubs

Προηγουμένως είδαμε τους τρόπους με τους οποίους παρασκηνιακά η SignalR προσδίδει real-time λειτουργικότητα στις Web εφαρμογές. Σε αυτήν την υπό ενότητα θα δούμε πως επιτυγχάνεται πρακτικά αυτή η λειτουργικότητα. Αφού λοιπόν εγκατασταθεί η Signal R σε μια .net Core εφαρμογή πρέπει να προστεθεί στα services της εφαρμογής στην κλάση Startup (Startup.cs) στην μέθοδο ConfigureServices.

```
services.AddSignalR();
```

Για να προσδώσουμε μια λειτουργία στην εφαρμογή πρέπει να δημιουργήσουμε ένα Hub. Το Hub είναι μια κλάση της Signal R η οποία δίνει την δυνατότητα κλήσης μεθόδων του server μέσω του client.Στον κώδικα του Server δημιουργούνται μέθοδοι οι οποίες καλούνται μέσω του client και αντίστοιχα στον κώδικα του client δημιουργούνται μέθοδοι η οποίες καλούνται μέσω του server. Αφού δημιουργηθεί το Hub πρέπει να γίνει map το route του Hub στην μέθοδο Configure της κλάσης Startup όπως φαίνεται στον παρακάτω κώδικα ώστε να μπορεί να επιτευχθεί στην συνέχεια σύνδεση στο συγκεκριμένο Hub στον κώδικα του client.

```
app.UseRouting();
app.UseEndpoints(endpoints =>
{
    endpoints.MapHub<HubClassName>("/desiredname");
});
```

3.5.9. Αποστολή ερώτησης με την Signal R

Για την επίτευξη της αποστολής της ερώτησης στους συνδεδεμένους clients δημιουργήθηκε η κλάση QuestionHub η οποία περιέχει την μέθοδο SendQuestion που δέχεται σαν παραμέτρους την ερώτηση σε μορφή string η οποία όπως φαίνεται στην εικόνα 3.4 επιλέγεται από το combobox που περιέχει τις διαθέσιμες ερωτήσεις του poll και το id του poll.Η κλίση της μεθόδου γίνεται μέσω του κώδικα του client της σελίδας που φαίνεται στην εικόνα 3.4 (GetAdaptedQuestionsAsync.cshtml).

Ο κώδικας της σελίδας μέσω της βιβλιοθήκης SignalR δημιουργεί μια μόνιμη σύνδεση με τον server ώστε να καλείται η μέθοδος SendQuestion κάθε φορά που πατιέται το κουμπί Send Question με την βοήθεια ενός click event listener που έχει προστεθεί στο κουμπί.

```
//Σύνδεση στο Hub
var connection = new signalR.HubConnectionBuilder().withUrl("/questionHub").bu
ild();
//Error Handling
document.getElementById("sendQ").disabled = true;
connection.start().then(function () {
  document.getElementById("sendQ").disabled = false;}).catch(function (err) {
  return console.error(err.toString());
  });
document.getElementById("sendQ").addEventListener("click", function (event) {
  var e = document.getElementById("qSelector");
  //Ορίζεται σε μεταβλητή το String της ερώτησης
  var question = e.options[e.selectedIndex].text;
  //Ορίζεται σε μεταβλητή το Id του Poll
  var sid = @TempData["Sessid"]
  @*invoke SendQuestion*@
  //Κλήση της μεθόδου SendQuestion του Hub
  connection.invoke("SendQuestion", question,sid).catch(function (err) {
              return console.error(err.toString());
  });
  event.preventDefault();
});
```

Παράδειγμα κώδικα 3.13. JS στη σελίδα αποστολής της ερώτησης

Αφού λοιπόν δεχτεί τις παραμέτρους η SendQuestions στέλνει ένα GET request σε ένα endpoint του API που περιέχει την ερώτηση σε μορφή string.Το endpoint αυτό επιστρέφει ένα Json αρχείο που περιέχει το μοντέλο της ερώτησης (id της ερώτησης, διαθέσιμες απαντήσεις, σωστή απάντηση κλπ.). Στην συνέχεια η μέθοδος κάνει update μια μικρή βάση sqllite που περιέχεται στα αρχεία του Server και κρατάει σε έναν πίνακα τα id των poll και τα id των τελευταίων ερωτήσεων που έχουν σταλεί σε αυτά ώστε ένας χρήστης που ενδεχομένως συνδεθεί αργότερα στο poll να μπορεί να λάβει την τρέχουσα ερώτηση μια διαδικασία της οποίας τα επόμενα βήματα είναι παρόμοια με αυτά που περιγράφονται οπότε δεν θα δοθεί ιδιαίτερη σημασία. Τέλος μέσω της SignalR η Send Questions καλεί την μέθοδο RecieveQuestions η οποία υλοποιείται στον κώδικα του client της σελίδας την οποία βλέπουν οι συνδεδεμένοι μαθητές (εικόνα 3.5) και δέχεται σαν παραμέτρους το

id του poll και τα στοιχεία της ερώτησης. Παρακάτω φαίνεται αναλυτικά ο κώδικας της SendQuestion

```
public async Task SendQuestion(string question,int sid)
{
  //GET Request για την απόκτηση του μοντέλου της ερώτησης
  Questions q = new Questions();
  HttpClient client = _api.Initial();
  HttpResponseMessage res =
  await client.GetAsync("api/Search/" + Uri.EscapeDataString(question));
  if (res.IsSuccessStatusCode)
  {
    var result = res.Content.ReadAsStringAsync().Result;
    q = JsonConvert.DeserializeObject<Questions>(result);
  }
  // Αποθήκευση/Ενημέρωση των id της ερώτησης και του poll στη SqlLite βάση
  using (var db = new CurrentPoll(env))
  {
    if (!db.CPoll.Select(a => a.PollId).Contains(sid))
    {
      db.CPoll.Add(new CPoll { PollId = sid, Qid = q.Qid });
    }
    else {
      var qtu = db.CPoll.Where(a => a.PollId == sid).FirstOrDefault();
      qtu.Qid = q.Qid;
          db.CPoll.Update(qtu);
        }
        var count = db.SaveChanges();
     }
  //Κλίση της ReceiveQuestion
  await Clients.All.SendAsync("ReceiveQuestion", question,q.C1,q.C2,q.C3,q.C4,
q.C5,sid,q.Qid);
  }
}
```

Παράδειγμα κώδικα 3.14. Η μέθοδος SendQuestion στο backend

Η μέθοδος ReceiveQuestion αφού ελέγξει ότι τα στοιχεία που έχουν σταλεί αφορούν το συγκεκριμένο poll μέσω του poll id ενημερώνει το front end του MVC Client ανοίγοντας ένα modal το που προβάλει την ερώτηση στον χρήστη και του δίνει την δυνατότητα να απαντήσει όπως φαίνεται στην εικόνα. Πιο κάτω βλέπουμε των κώδικα της RecieveQuestion.

```
//Connect to Hubs(Το questionHub αφορά την διαδικασία που περιγράφεται)
var connection =
new signalR.HubConnectionBuilder().withUrl("/questionHub").build();
var connection2 =
new signalR.HubConnectionBuilder().withUrl("/resultHub").build();
var connection3 =
new signalR.HubConnectionBuilder().withUrl("/currentQHub").build();
//Κατά την αποστολή της ερώτησης
connection.on("ReceiveQuestion", function (question, c1, c2, c3, c4, c5, sid,
qid) {
  //Έλεγχος αν η ερώτηση προορίζεται για το συγκεκριμένο poll
  if (sid == @Model.SessionId) {
  //Ενημέρωση των πεδίων του modal
  document.getElementById("qTitle").textContent = question
```

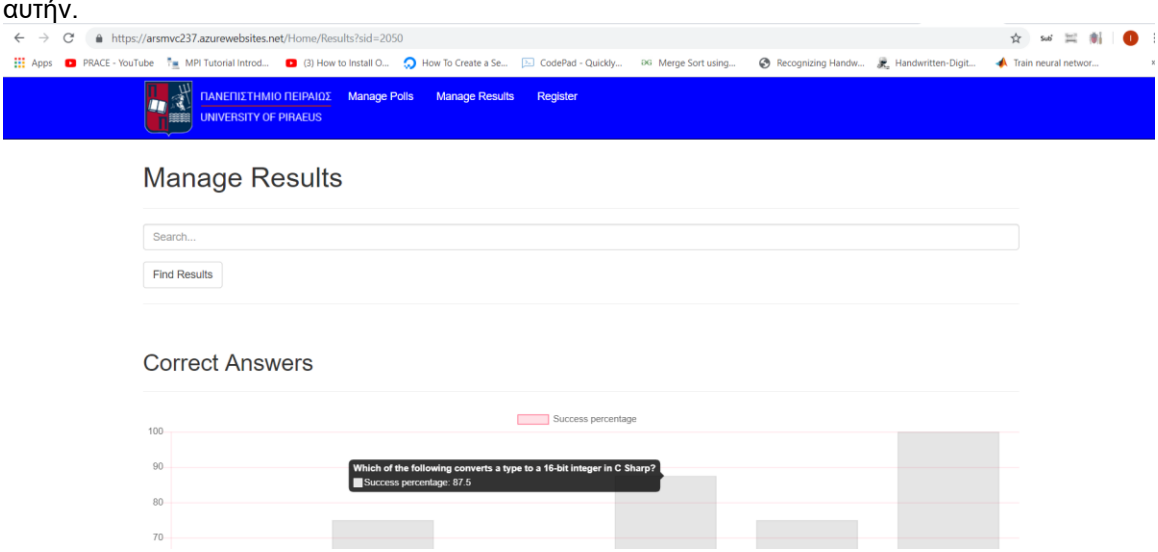
```
  document.getElementById("ch1").textContent  = c1
  document.getElementById("ch2").textContent  = c2
  document.getElementById("ch3").textContent  = c3
  document.getElementById("ch4").textContent = c4
  document.getElementById("ch5").textContent = c5
  document.getElementById("Aresult").innerHTML = "<div class='alert alert-
info alert-dismissible fade in'><a href = '#' class='close' data-
dismiss='alert' aria-
label='close'>&times;</a><strong>Here you go!</strong>Choose the correct answe
r !!</div>"
    Quid = qid;
    //Εμφάνιση του modal
    $('#myModal').modal('show');
 }
});
```

Παράδειγμα κώδικα 3.15. JS στην σελίδα του poll

3.5.10. Προβολή αποτελεσμάτων

Προηγουμένως αναφέραμε ότι μέσω της σελίδας Manage Results ο καθηγητής μπορεί να δει το ποσοστό επιτυχίας των απαντήσεων των μαθητών. Μπαίνοντας σε αυτή την σελίδα υπάρχει ένα πεδίο autocomplete στο οποίο ο καθηγητής μπορεί να αναζητήσει αποτελέσματα συμπληρώνοντας το id του poll για το οποίο επιθυμεί να δει αποτελέσματα.

Τα αποτελέσματα εμφανίζονται σε ένα View Component μέσω της μορφής ενός διαγράμματος όπου κάθε μπάρα αντιπροσωπεύει και μια ερώτηση που έχει σταλεί. Στην παρακάτω εικόνα μπορούμε να δούμε την σελίδα Manage Results και πως εμφανίζονται τα αποτελέσματα σε αυτήν.



Εικόνα 3.11. Σελίδα προβολής αποτελεσμάτων

Για να επιτευχθεί η παραπάνω διαδικασία έχει δημιουργηθεί ένα View Model με τα εξής properties :

```
public class StatsViewModel
    {
```

```
            public string DimensionOne { get; set; }
            public double Quantity { get; set; }

    }
```

Παράδειγμα κώδικα 3.16. View model στατιστικών

Το DimensionOne αντιπροσωπεύει το κείμενο της ερώτησης και το Quantity το ποσοστό επιτυχίας. Αφού επιλεγεί το id του poll στο autocomplete της σελίδας καλείται το View Component του διαγράμματος με παράμετρο το id του poll.

```
<p> @await Component.InvokeAsync("Results", TempData["SessionId"])</p>
```

Παράδειγμα κώδικα 3.17. Κλήση της μεθόδου Results

Στον Controller του View Component στέλνονται δύο Requests στο Web Api που επιστρέφουν τις ερωτήσεις και τις απαντήσεις . Ανατρέχοντας στις απαντήσεις βρίσκεται ποιες από αυτές αφορούν το επιλεγμένο poll και εισάγονται τα id τους σε μια λίστα. Στην συνέχεια ανατρέχετε αυτή η λίστα και για κάθε ερώτηση βρίσκεται πόσες ήταν οι απαντήσεις και πόσες από αυτές ήταν σωστές. Στην συνέχεια βγαίνει το ποσοστό επιτυχίας και αποθηκεύεται μαζί με το κείμενο της ερώτησης σαν αντικείμενο σε μια λίστα StatsViewModel η οποία τελικά επιστρέφεται στο View του View Component.

```
var lstModel = new List<StatsViewModel>();
          List<Answer> answers = new List<Answer>();
          List<Questions> ques = new List<Questions>();
          List<string> results = new List<string>();
          List<int> qids = new List<int>();
          HttpClient client = _api.Initial();
          HttpResponseMessage res = await client.GetAsync("api/Answers");
          HttpResponseMessage res2 = await client.GetAsync("api/Questions");
          if (res.IsSuccessStatusCode){
              var result = res.Content.ReadAsStringAsync().Result;
              answers = JsonConvert.DeserializeObject<List<Answer>>(result);
          }
          if (res2.IsSuccessStatusCode){
              var result2 = res2.Content.ReadAsStringAsync().Result;
              ques = JsonConvert.DeserializeObject<List<Questions>>(result2);
          }
          foreach (var item in answers.Where(a => a.SessionId == sid).ToList()
) {
              if (!qids.Contains(item.QuestionId)){
                  qids.Add(item.QuestionId);
              }
          }
          foreach(var id in qids) {
              int all = answers.Where(a => a.QuestionId == id).Where(c => c.Se
ssionId == sid).ToList().Count;
              int right = answers.Where(a => a.QuestionId == id).Where(c => c.
SessionId == sid).Where(b => b.IsRight == true).ToList().Count;
              double result = (double)right / all;
              lstModel.Add(new StatsViewModel
              {
                  DimensionOne = ques.Where(a => a.Qid == id).Select(b => b.Qu
estion).FirstOrDefault(),
```

```
                Quantity = result * 100
            });}return View(lstModel);
```

Παράδειγμα κώδικα 3.19. Results view component

Στον client κώδικα του View Component τα κείμενα των ερωτήσεων και τα αποτελέσματα αποθηκεύονται σε 2 μεταβλητές λίστας.

```
@model List<MainApp.Models.ViewModels.StatsViewModel>
@{
    var XLabels2 = Newtonsoft.Json.JsonConvert.SerializeObject(Model.Select(x
=> x.DimensionOne).ToList());
    var YValues2 = Newtonsoft.Json.JsonConvert.SerializeObject(Model.Select(x
=> x.Quantity).ToList());
    ViewData["Title"] = "Bar Chart";
}
```

Παράδειγμα κώδικα 3.20. Bar chart JS I

Τέλος με την βοήθεια της βιβλιοθήκης Chart.js εμφανίζεται το διάγραμμα στον χρήστη.

```
<script type="text/javascript">
      $(function () {
    var chartName = "chart2";
        var ctx = document.getElementById(chartName).getContext('2d');
        var data = {
                labels: @Html.Raw(XLabels2),
                datasets: [{
                    label: "Success percentage",
                    backgroundColor: [
                        'rgba(255, 99, 132, 0.2)',

                    ],
                    borderColor: [
                        'rgba(255,99,132,1)',

                    ],
                    borderWidth: 1,
                    data: @Html.Raw(YValues2)
    }]
            };

var options = {
                maintainAspectRatio: false,
                scales: {
                    yAxes: [{

                        ticks: {
                            min: 0,
                            beginAtZero: true

                        },
                        gridLines: {
                            display: true,
                            color: "rgba(255,99,132,0.2)"
                        }
```

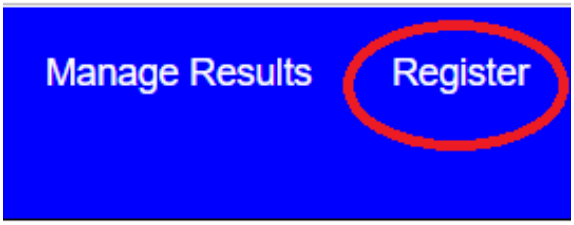
```
}],
                    xAxes: [{
                        ticks: {
                            display: false,
                            min: 0,
                            beginAtZero: true
                        },
                        gridLines: {
                            display: false
                        }
                    }]
                }
            };

          var myChart = new Chart(ctx, {

                options: options,
                data: data,
                type:'bar'

            });
        });
</script>
```

Παράδειγμα κώδικα 3.21. Bar chart JS II

3.6. Mobile application – Identity Server

3.6.1. Περιγραφή διαδικασίας σύνδεσης

Έχει αναφερθεί ότι οι μαθητές συνδέονται σε ένα ανοιχτό poll μέσω της κινητής συσκευής τους αφού πρώτα ολοκληρώσουν την διαδικασία εγγραφής στον Identity Sever με την χρήση ενός ηλεκτρονικού υπολογιστή και της κινητής συσκευής. Η διαδικασία αποτελείται από δύο κομμάτια. Το πρώτο βήμα είναι ο μαθητής να μεταβεί στην σελίδα της εφαρμογής (MVC Client) και να μπει στον σύνδεσμο Register.

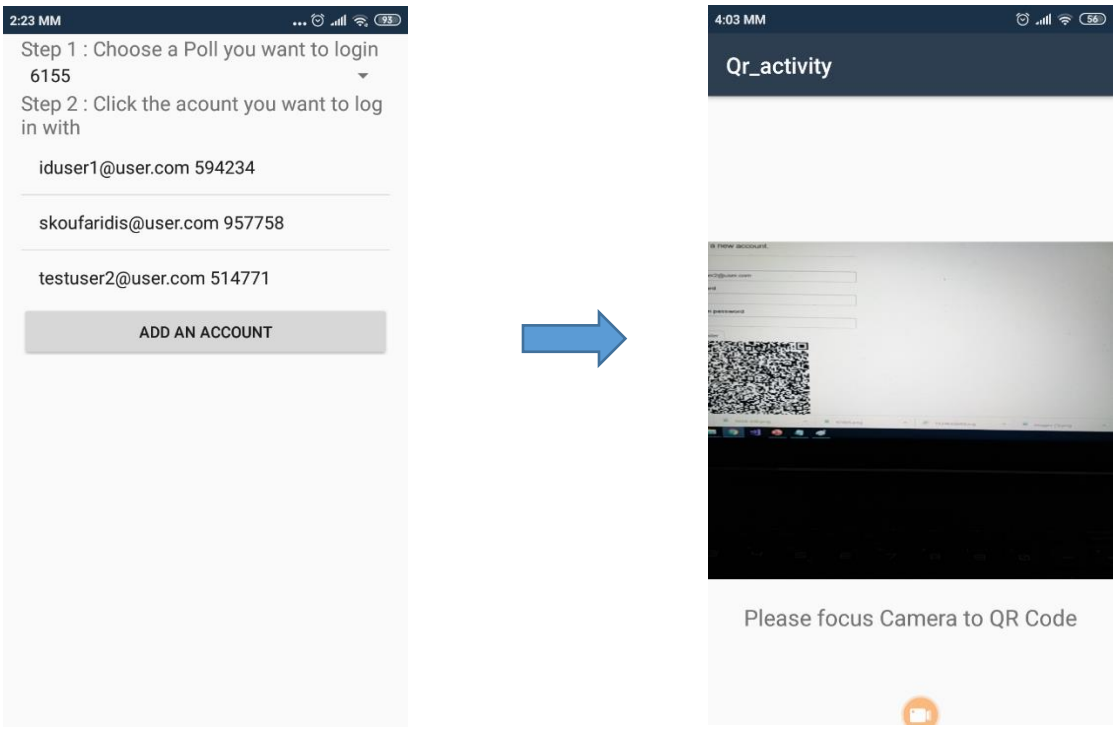


Εικόνα 3.12. Σύνδεσμος Register MVC Client

Με το πάτημα του συνδέσμου μεταφέρεται στην σελίδα του Identity Server όπου εκεί μπορεί να εγγραφεί στον Identity Server σαν χρήστης. Κατά την ολοκλήρωση της εγγραφής δημιουργείται ένα bar code το οποίο περιέχει ένα Hash κλειδί και το email του χρήστη.

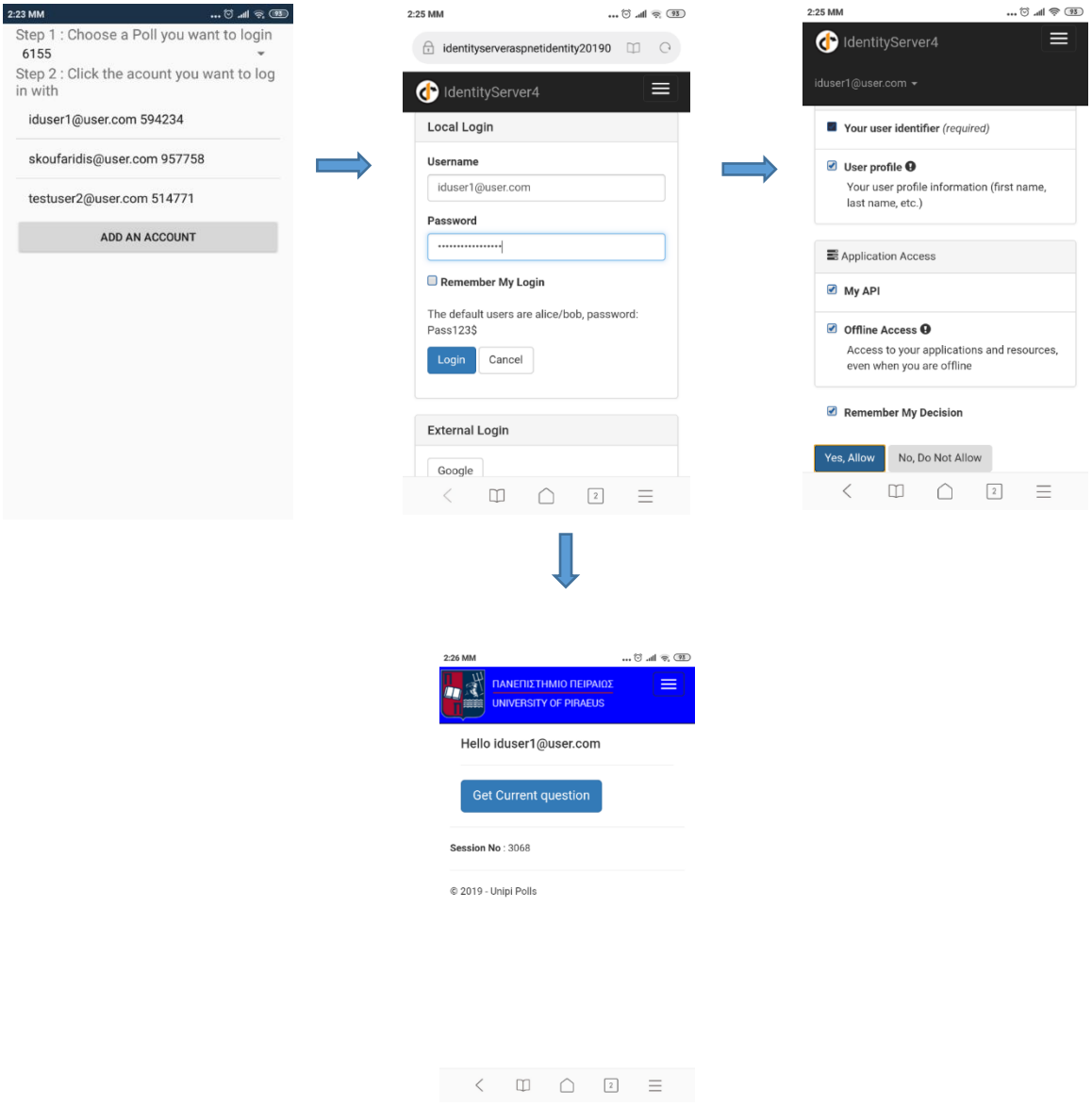
Εικόνα 3.13. Εγγραφή Identity Server

Μετά την δημιουργία του Hash κλειδιού περνάμε στο δεύτερο βήμα της εγγραφής το οποίο είναι η αποθήκευση του email και του κλειδιού στην κινητή συσκευή. Ο Μαθητής, αφού πρώτα ταυτοποιηθεί μέσω δαχτυλικού αποτυπώματος, μπαίνοντας στην mobile εφαρμογή και πατώντας «Add an Account» ανοίγει μια activity η οποία μέσω της κάμερας του κινητού μπορεί να σκανάρει το bar code που έχει δημιουργηθεί και να αποθηκεύσει αυτόματα το hash κλειδί και το email σε μια κωδικοποιημένη βάση sqlite.



Εικόνα 3.14. Προσθήκη λογαριασμού στην εφαρμογή

Με το πέρας της διαδικασίας στην αρχική οθόνη της εφαρμογής προστίθεται το email του χρήστη και δίπλα ένας one time password που αλλάζει κάθε συγκεκριμένο χρονικό διάστημα ο οποίος δημιουργείτε βάση του hash κλειδιού με μια διαδικασία την οποία θα περιγράψουμε στην συνέχεια. Επίσης στην αρχική οθόνη φαίνονται και τα διαθέσιμα ανοιχτά poll στα οποία μπορεί να συνδεθεί ο μαθητής. Επιλέγοντας ένα poll και κάνοντας click σε έναν λογαριασμό ανοίγει ένας browser στην κινητή συσκευή ο οποίος μεταφέρει τον χρήστη στον Identity Server στον οποίο έχει εγγραφεί προηγουμένως εκεί δίνοντας τα στοιχεία σύνδεσης του λογαριασμού του ο χρήστης και σε συνδυασμό με τον one time password γίνεται ταυτοποίηση του χρήστη ο οποίος τελικά μεταφέρεται στην σελίδα του MVC Client όπου μπορεί να δει τις ερωτήσεις που στέλνονται στο poll που έχει συνδεθεί και να απαντήσει σε αυτές.



Εικόνα 3.15. Σύνδεση στο poll μέσω εφαρμογής

Συμπερασματικά για την είσοδο στην εφαρμογή ο χρήστης περνάει από μια διαδικασία ταυτοποίησης τριών παραγόντων (3-factor authentication) που αποτελείται από την εισαγωγή του δαχτυλικού αποτυπώματος του χρήστη , τον otp και τα credentials του χρήστη στον identity server. Στις επόμενες ενότητες αφού πρώτα αναλυθεί πως λειτουργεί ο αλγόριθμος TOTP αλλά και ο προσδιορισμός OAuth 2.0 θα δούμε την υλοποίηση τους στο παρόν σύστημα.

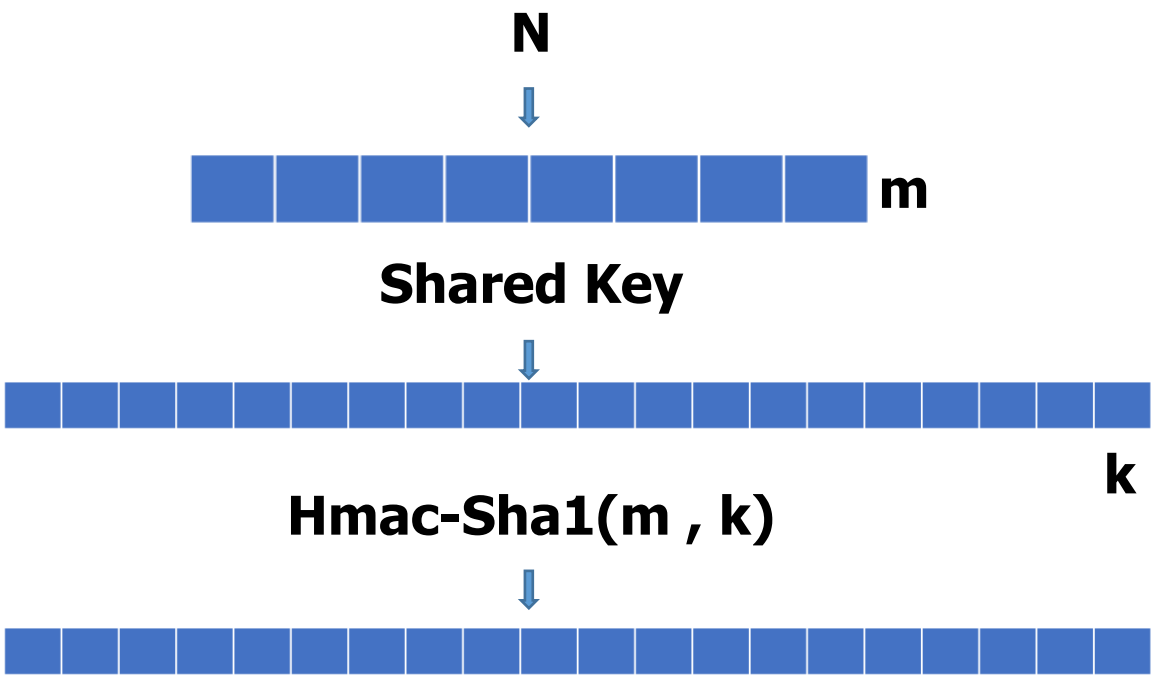
3.6.2. Ο αλγόριθμος TOTP (Time based one time password)

Ο αλγόριθμος αυτός είναι υπεύθυνος για την δημιουργία κωδικών οι οποίοι είναι σε ισχύ για ένα συγκεκριμένο χρονικό διάστημα με γνώμονα ένα κοινό hash κλειδί. Σε αυτήν την ενότητα θα εξηγήσουμε πως λειτουργεί αυτός ο αλγόριθμος. Το πρώτο βήμα του αλγόριθμου είναι να μετατρέψει την διεθνή ώρα (UTC) την δεδομένη στιγμή σε ώρα Unix. Η ώρα Unix η αλλιώς epoch time είναι ο αριθμός δευτερολέπτων που παρεμβάλλονται μεταξύ της 1 Ιανουαρίου 1970 00:00 UTC και της δεδομένης διεθνής ώρας. Στην συνέχεια με βάση την παρακάτω εξίσωση ορίζουμε των αριθμό των time steps στα οποία χωρίζεται η ώρα Unix και τα οποία είναι επί της ουσίας ο χρόνος που ορίζει την περίοδο δημιουργίας otp κωδικών και είναι συνήθως 30 δευτερόλεπτα.

N = floor(Tunix / ts)

- **N =** Ο αριθμός των time steps στα οποία χωρίζεται η ώρα Unix.
- **floor =** Προς τα κάτω στρογγυλοποίηση σε ακέραιο.
- **Tunix =** Ώρα Unix.
- **ts =** time step.

Σε δεύτερη φάση το N μετατρέπεται σε δεκαεξαδικό αριθμό ο οποίος αντιπροσωπεύει έναν πίνακα 8 byte ο οποίος αποθηκεύεται σε μια μεταβλητή m(= message). Το κοινό κλειδί το οποίο μοιράζονται οι δύο πλευρές είναι ένας τυχαίος αριθμός 20 byte κωδικοποιημένος με base-32 κωδικοποίηση και αντιπροσωπεύει έναν πίνακα 20 byte που αποθηκεύεται σε μια μεταβλητή K. Οι μεταβλητές K, m μέσω του αλγόριθμου HMAC-SHA1 μετατρέπονται σε έναν HMAC hash μήκους 20 byte.



Το επόμενο βήμα είναι να βρούμε το offset το οποίο είναι η ακέραια τιμή των τελευταίων 4 bit του HMAC hash. Για την οικονομία του παραδείγματος ας υποθέσουμε ότι έχουμε σαν αποτέλεσμα το παρακάτω HMAC hash.

AF	16	86	8F	E5	DB	00	CI	58	75	F6	A7	F8	99	F5	28	AB	80	5E	9A

Τα τελευταία 4 bit είναι ο δεκαεξαδικός αριθμός 0xA ο οποίος αντιπροσωπεύει τον δεκαδικό αριθμό 10. Έτσι λοιπόν ξεκινώντας από την θέση 10 του hash παίρνουμε τα πρώτα 4 byte.

Offset = 10

AF	16	86	8F	E5	DB	00	CI	58	75	F6	A7	F8	99	F5	28	AB	80	5E	9A

Τέλος εφαρμόζοντας μια δυαδική πράξη (binary operation) για το καθένα από τα 4 αυτά byte καταλήγουμε σε έναν δεκαεξαδικό αριθμό ο οποίος μετατρέπεται σε ακέραιο ο οποίος τελικά διαιρείται με τον 10 υψωμένο στον αριθμό των ψηφίων που θέλουμε να έχει ο otp. Η διαίρεση αυτή μας δίνει σαν αποτέλεσμα τον τελικό otp κωδικό (token).

F6	A7	F8	99

0x76A7F899

1990719641

$\% 10^{T=6}$

719641

Token

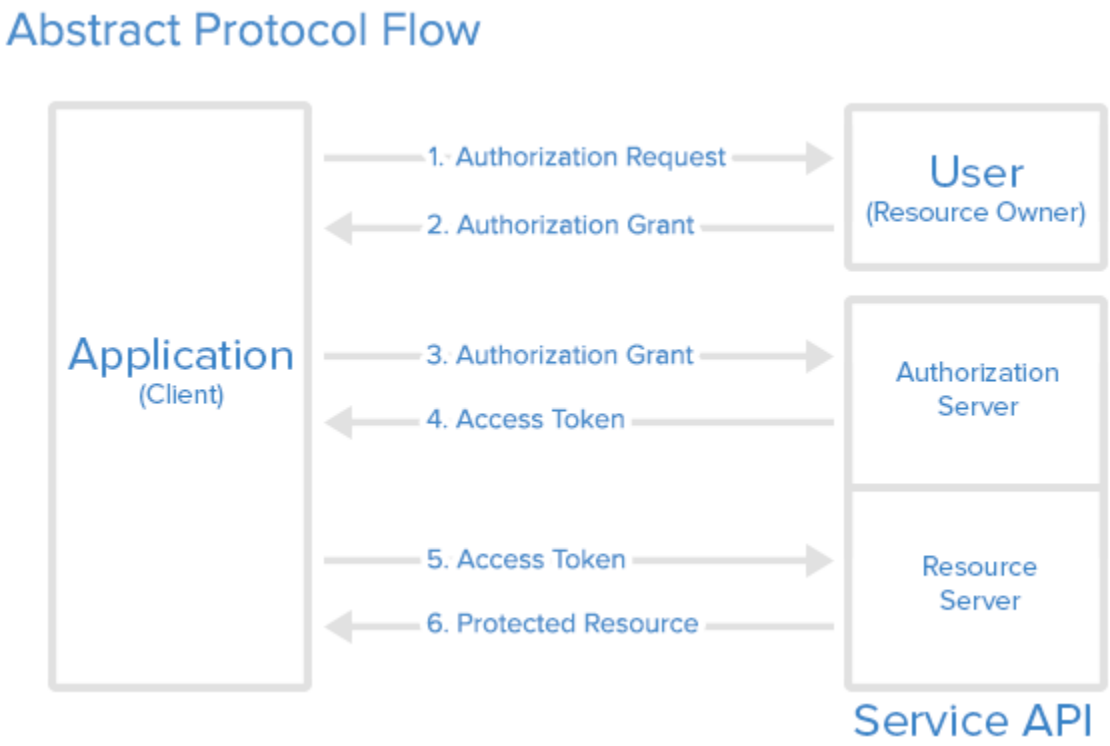
Η ίδια διαδικασία ακολουθείται και για την επαλήθευση του κωδικού καθώς το κλειδί είναι κοινό οπότε μέσα στον χρόνο ζωής του token θα παραχθεί ο ίδιος κωδικός.

3.6.3. Oauth 2.0 Authentication

Στο παραδοσιακό μοντέλο ταυτοποίησης client-server, ο client ζητά έναν προστατευμένο πόρο που βρίσκεται στον server χρησιμοποιώντας για ταυτοποίηση τα διαπιστευτήρια του ιδιοκτήτη του πόρου (συνήθως φυσικός χρήστης).Έτσι προκειμένου οι εφαρμογές τρίτων να αποκτήσουν πρόσβαση σε περιορισμένες πηγές, ο κάτοχος πόρων μοιράζεται τα διαπιστευτήριά του με αυτές. Αυτό δημιουργεί αρκετά προβλήματα και περιορισμούς:

- Οι εφαρμογές τρίτων απαιτείται να αποθηκεύσουν των διαπιστευτήρια του κατόχου για μελλοντική χρήση, συνήθως έναν κωδικό πρόσβαση καθαρού κείμενου.
- Οι διακομιστές χρειάζεται να υποστηρίξουν τον έλεγχο ταυτότητας με κωδικό πρόσβασης παρά τις αδυναμίες ασφαλείας που είναι εγγενείς στους κωδικούς πρόσβασης.
- Οι εφαρμογές τρίτου μέρους αποκτούν υπερβολικά ευρεία πρόσβαση στους προστατευμένους πόρους του ιδιοκτήτη, αφήνοντας τους ιδιοκτήτες των πόρων χωρίς καμία δυνατότητα περιορισμού της διάρκειας ή της πρόσβασης σε ένα περιορισμένο υποσύνολο πόρων.
- Οι κάτοχοι πόρων δεν μπορούν να ανακαλέσουν την πρόσβαση σε ένα τρίτο μέρος χωρίς να ανακαλέσουν την πρόσβαση σε όλα τα τρίτα μέρη.
- Η παραβίαση οποιασδήποτε τρίτης εφαρμογής προκαλεί παραβίαση του κωδικού πρόσβασης του τελικού χρήστη και όλα τα δεδομένων που προστατεύονται από αυτό τον κωδικό πρόσβασης.

Το OAuth αντιμετωπίζει αυτά τα προβλήματα εισάγοντας ένα επίπεδο εξουσιοδότησης που αποσκοπεί στον διαχωρισμό του ρόλου του client από τον ρόλο του ιδιοκτήτη πόρων. Στο OAuth, ο client ζητά πρόσβαση σε ελεγχόμενους πόρους από τον ιδιοκτήτη του πόρου που φιλοξενούνται από έναν διακομιστή, και του εκδίδεται έναν διαφορετικό σύνολο διαπιστευτηρίων από αυτά του ιδιοκτήτη.



Εικόνα 3.16. Oauth abstract flow

Αντί να χρησιμοποιούνται τα διαπιστευτήρια του ιδιοκτήτη για πρόσβαση στους προστατευμένους πόρους, ο client αποκτά ένα αναγνωριστικό πρόσβασης (access token) - μια συμβολοσειρά που υποδηλώνει συγκεκριμένο πεδίο εφαρμογής, διάρκεια ζωής και άλλα χαρακτηριστικά πρόσβασης. Τα access tokens εκδίδονται στους clients από έναν διακομιστή εξουσιοδότησης με την έγκριση του ιδιοκτήτη. Ο client χρησιμοποιεί το access token για πρόσβαση στους προστατευμένους πόρους που φιλοξενεί ο διακομιστής πόρων.

3.6.4. Ρόλοι

Το OAuth ορίζει τέσσερις ρόλους:

1.Ιδιοκτήτης πόρων (Resource Owner)

Μια οντότητα ικανή να παρέχει πρόσβαση σε προστατευμένο πόρο. Όταν ο ιδιοκτήτης πόρων είναι ένα άτομο, αναφέρεται ως τελικός χρήστης(End user).

2.Διακομιστή πόρων(Resource Server)

Ο διακομιστής που φιλοξενεί τους προστατευμένους πόρους, είναι σε θέση να αποδεχθεί και να ανταποκριθεί σε αιτήματα προστατευόμενων πόρων με τη χρήση access tokens.

3.Πελάτης (Client)

Μια εφαρμογή που κάνει αιτήσεις προστατευόμενων πόρων για λογαριασμό του ιδιοκτήτη πόρων αφού πρώτα του δοθεί εξουσιοδότηση. Ο όρος "client" δεν υπονοεί κάποια ιδιαίτερα χαρακτηριστικά εφαρμογής (π.χ., αν η εφαρμογή εκτελείται σε server, desktop ή άλλη συσκευή).

4.Διακομιστής εξουσιοδότησης(Authorization Server)

Ο διακομιστής που εκδίδει access tokens στον client μετά από επιτυχία επαλήθευσης της ταυτότητας του ιδιοκτήτη του πόρου και παροχή άδειας. Η αλληλεπίδραση μεταξύ του διακομιστή εξουσιοδότησης και του διακομιστή πόρων είναι πέρα από το πεδίο εφαρμογής της προδιαγραφής Oauth. Ο διακομιστής εξουσιοδότησης μπορεί να είναι ο ίδιος διακομιστής με τον διακομιστή πόρων ή με μια ξεχωριστή οντότητα. Ένας μοναδικός διακομιστής εξουσιοδότησης μπορεί να εκδίδει access tokens αποδεκτά από πολλούς διακομιστές πόρων.

3.6.5. Χορήγηση εξουσιοδότησης (Access Grant)

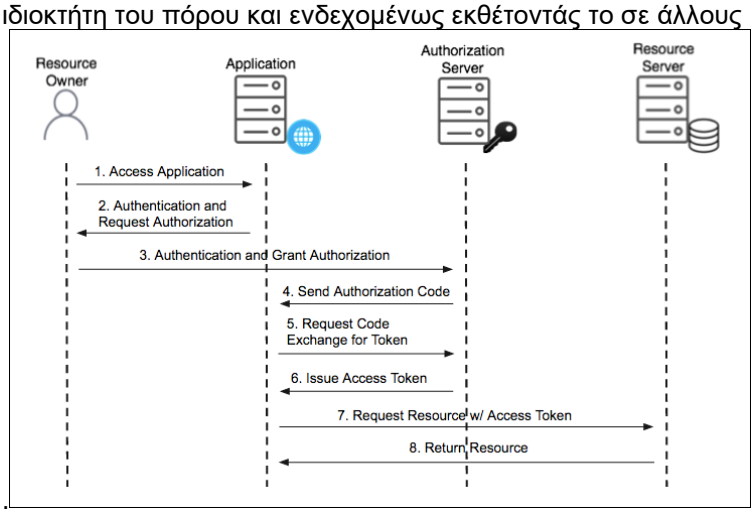
Μια χορήγηση εξουσιοδότησης είναι μια πιστοποίηση που αντιπροσωπεύει την εξουσιοδότηση του ιδιοκτήτη (για πρόσβαση στους προστατευμένους πόρους του) που χρησιμοποιείται από το πελάτη για να αποκτήσει ένα access token.Η προδιαγραφή OAuth 2.0 ορίζει τέσσερεις τύπους χορήγησης οι οποίοι και θα περιγράφουν παρακάτω.

1.Κωδικός εξουσιοδότησης (Authorization code)

Ο κωδικός εξουσιοδότησης αποκτάται χρησιμοποιώντας έναν διακομιστή εξουσιοδότησης ως διαμεσολαβητή μεταξύ του πελάτη και του ιδιοκτήτη του πόρου. Αντί να ζητηθεί εξουσιοδότηση απευθείας από τον ιδιοκτήτη του πόρου, ο client κατευθύνει τον ιδιοκτήτη του πόρου σε έναν διακομιστή εξουσιοδότησης (μέσω του user-agent) , ο οποίος με τη σειρά του επιστρέφει τον ιδιοκτήτη πόρων πίσω στον client μαζί με έναν κωδικό εξουσιοδότησης. Προτού επιστρέψει ο ιδιοκτήτης στον client με τον κωδικό εξουσιοδότησης, ο διακομιστής εξουσιοδότησης πιστοποιεί τον ιδιοκτήτη και λαμβάνει εξουσιοδότηση.

Επειδή ο ιδιοκτήτης επαληθεύεται μόνο στον διακομιστή εξουσιοδότησης, ο πόρος και τα διαπιστευτήρια του ιδιοκτήτη δεν μοιράζονται και δεν γίνονται ποτέ γνωστά στον client.Εν συνεχεία ο client με την χρήση του κωδικού εξουσιοδότησης λαμβάνει ένα access token από τον διακομιστή

εξουσιοδότησης το οποίο τέλος χρησιμοποιεί για το αίτημα απόκτησης του προστατευόμενου πόρου από τον διακομιστή πόρων. Ο κωδικός εξουσιοδότησης παρέχει μερικά σημαντικά οφέλη ασφάλειας, όπως η δυνατότητα επαλήθευσης της ταυτότητας του client, καθώς και της μετάδοσης του access token απευθείας στον client χωρίς την μεταβίβαση του μέσω του user agent του ιδιοκτήτη του πόρου και ενδεχομένως εκθέτοντάς το σε άλλους

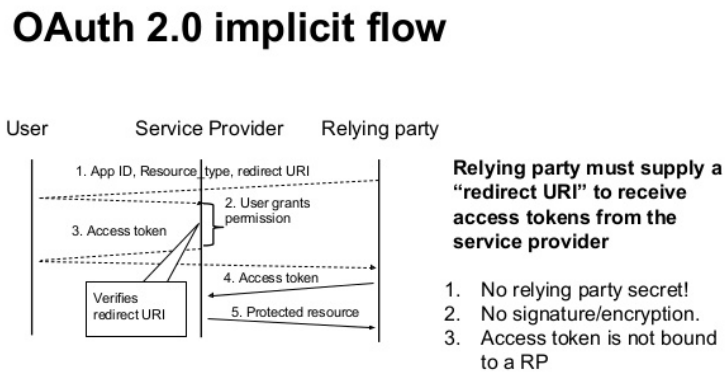


Εικόνα 3.17. Authorization code flow

2.Ανεμπόδιστη Ροή (Implicit Flow Grant)

Η ανεμπόδιστη επιχορήγηση είναι μια απλοποιημένη ροή authorization code βελτιστοποιημένη για clients που υλοποιούνται σε ένα πρόγραμμα περιήγησης χρησιμοποιώντας μια γλώσσα όπως η JavaScript. Στη ανεμπόδιστη ροή, αντί να εκδίδεται στον client ένας κωδικός εξουσιοδότησης, εκδίδεται απευθείας το access token (ως αποτέλεσμα της εξουσιοδότησης του ιδιοκτήτη πόρων). Ο τύπος χορήγησης είναι ανεμπόδιστος, καθώς δεν υπάρχουν ενδιάμεσα διαπιστευτήρια (authorization code). Όταν εκδίδετε ένα access token κατά τη διάρκεια της ανεμπόδιστης ροής ο διακομιστής εξουσιοδότησης δεν επαληθεύει τον client. Σε ορισμένες περιπτώσεις, η ταυτότητα του client μπορεί να επαληθευτεί μέσω του URI ανακατεύθυνσης που χρησιμοποιείται για την παράδοση του access token.

Οι ανεμπόδιστες επιχορηγήσεις βελτιώνουν την ανταπόκριση και την αποτελεσματικότητα ορισμένων clients δεδομένου ότι μειώνουν τον αριθμό των αιτημάτων και ανακατευθύνσεων που απαιτούνται για την απόκτηση ενός access token παρόλα αυτά το access token μπορεί να εκτεθεί σε άλλες εφαρμογές με πρόσβαση στον user agent του ιδιοκτήτη του πόρου.

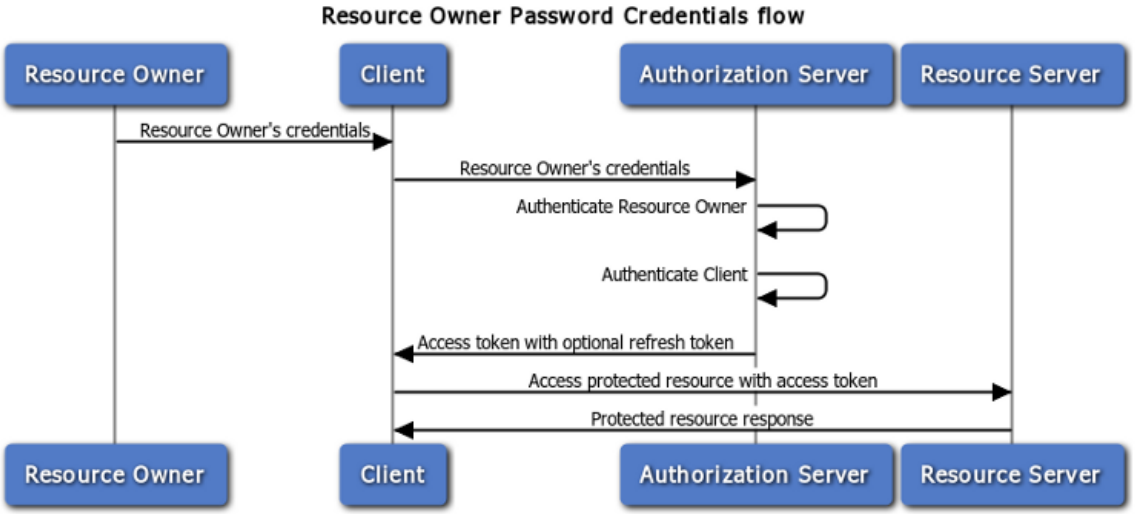


Εικόνα 3.18. Implicit flow

3.Διαπιστευτήρια ιδιοκτήτη πόρων (Resource Owner Password Credentials)

Τα διαπιστευτήρια του ιδιοκτήτη πόρων (δηλ. Όνομα χρήστη και κωδικός πρόσβασης) μπορούν να χρησιμοποιηθούν απευθείας ως επιχορήγηση εξουσιοδότησης . Τα διαπιστευτήρια πρέπει να χρησιμοποιούνται μόνο όταν υπάρχει υψηλός βαθμός εμπιστοσύνης μεταξύ του ιδιοκτήτη του πόρου και του client (π.χ. Ο client είναι μέρος του λειτουργικού συστήματος μιας συσκευής).

Παρόλο που ο συγκεκριμένος τύπος επιχορήγησης απαιτεί άμεση πρόσβαση του client στα διαπιστευτήρια ιδιοκτήτη πόρων, χρησιμοποιούνται τα διαπιστευτήρια ιδιοκτήτη πόρων για ένα μόνο αίτημα και ανταλλάσσονται με ένα access token. Αυτό ο τύπος επιχορήγησης μπορεί να εξαλείψει την ανάγκη αποθήκευσης των διαπιστευτηρίων του ιδιοκτήτη για μελλοντική χρήση, ανταλλάσσοντας το τα διαπιστευτήρια με ένα access token η ένα refresh token μακράς διαρκείας.



Εικόνα 3.19. Resource owner password credentials flow

4.Διαπιστευτήρια client

Τα διαπιστευτήρια client χρησιμοποιούνται ως επιχορήγηση εξουσιοδότησης συνήθως όταν ο client ενεργεί για λογαριασμό του (ο πελάτης είναι επίσης ο ιδιοκτήτης του πόρου) ή ζητά πρόσβαση σε συγκεκριμένους προστατευμένους πόρους που βασίζονται σε εξουσιοδότηση προκαθορισμένη με τον διακομιστή εξουσιοδότησης.

3.6.6. OpenID Connect

Το OpenID Connect είναι ένα απλό επίπεδο ταυτοποίησης που υλοποιείτε πάνω από το πρωτόκολλο OAuth 2.0, το οποίο επιτρέπει στους clients να επαληθεύουν την ταυτότητα ενός τελικού χρήστη με βάση τον έλεγχο ταυτότητας που εκτελείται από έναν server εξουσιοδότησης (Authorization Server) ή έναν πάροχο ταυτότητας (IdP) και δίνει την δυνατότητα παροχής πληροφοριών σχετικά με τον τελικό χρήστη κάνοντας χρήση του πρωτόκολλου REST.

Το OpenID Connect καθορίζει ένα RESTful HTTP API, χρησιμοποιώντας την μορφή δεδομένων JSON .Πλέον το OpenID Connect είναι το πιο διαδεδομένο πρωτόκολλο ελέγχου ταυτότητας: όταν μια εφαρμογή ζητά την πιστοποίηση της ταυτότητάς του χρήστη χρησιμοποιώντας τα διαπιστευτήρια του στο Facebook ή στο Google+, η εφαρμογή πιθανώς χρησιμοποιεί το OpenID Connect.Το OpenID Connect επιτρέπει σε web, mobile και javascript clients, να ζητούν και να λαμβάνουν πληροφορίες σχετικά με τις περιόδους σύνδεσης και τους τελικούς χρήστες.
Τα τρία βασικά πλεονεκτήματα του OpenID Connect είναι τα παρακάτω :

1. **Εύχρηστα Id Tokens**

Οι clients λαμβάνουν την ταυτότητα του χρήστη κωδικοποιημένη σε ένα ασφαλές JSON Web Token (JWT) το επονομαζόμενο Id Token. Τα JWT είναι ευανάγνωστα και φορητά και υποστηρίζουν μια μεγάλη γκάμα αλγορίθμων υπογραφής και κρυπτογράφησης.

2. **Το πρωτόκολλο OAuth 2.0**

Οι clients χρησιμοποιούν ροές OAuth 2.0 για να αποκτήσουν τα Id Tokens τα οποία μπορούν να καταναλωθούν εύκολα από web και native mobile εφαρμογές. Η χρήση του OAuth 2.0 σημαίνει επίσης ότι υπάρχει ένα ενιαίο πρωτόκολλο για τον έλεγχο ταυτότητας (authentication) και την εξουσιοδότηση (authorization) με την χρήση των access tokens.

3. **Απλό και ισχυρό**

Το OpenID Connect είναι αρκετά απλό να ενσωματωθεί σε βασικές εφαρμογές, προσφέροντας χαρακτηριστικά και επιλογές ασφαλείας που πέραν των βασικών εφαρμογών μπορούν να καλύψουν ικανοποιητικά και αρκετά απαιτητικές επιχειρηματικές εφαρμογές.

3.6.7, Id Token

Το Id token είναι μια τυποποιημένη μορφή JWT, που υπογράφεται από τον πάροχο OpenID (OP). Για να αποκτήσει ο client ένα Id Token πρέπει να μεταφέρει τον χρήστη στο OP με ένα αίτημα ελέγχου ταυτότητας.

Τα βασικά χαρακτηριστικά του Id Token είναι τα παρακάτω :

- Περιέχει την ταυτότητα του χρήστη, που ονομάζεται θέμα (Subject) (sub).
- Ορίζει την αρχή έκδοσης (ISS).
- Δημιουργείται για ένα συγκεκριμένο client (aud).
- Μπορεί να περιέχει ένα nonce (nonce).
- Μπορεί να καθορίσει πότε (auth_time) και πώς (acr) ο χρήστης πιστοποιήθηκε.
- Περιέχει τον χρόνο έκδοσης (iat) και το χρόνο λήξης (exp).
- Μπορεί να περιλαμβάνει επιπλέον ζητούμενες λεπτομέρειες σχετικά με την ταυτότητα του χρήστη, όπως όνομα και διεύθυνση ηλεκτρονικού ταχυδρομείου.
- Υπογράφεται ψηφιακά, επομένως μπορεί να επαληθευτεί από τους παραλήπτες που προορίζεται.
- Μπορεί προαιρετικά να κρυπτογραφηθεί.

Το Id Token αντιπροσωπεύεται από ένα json object

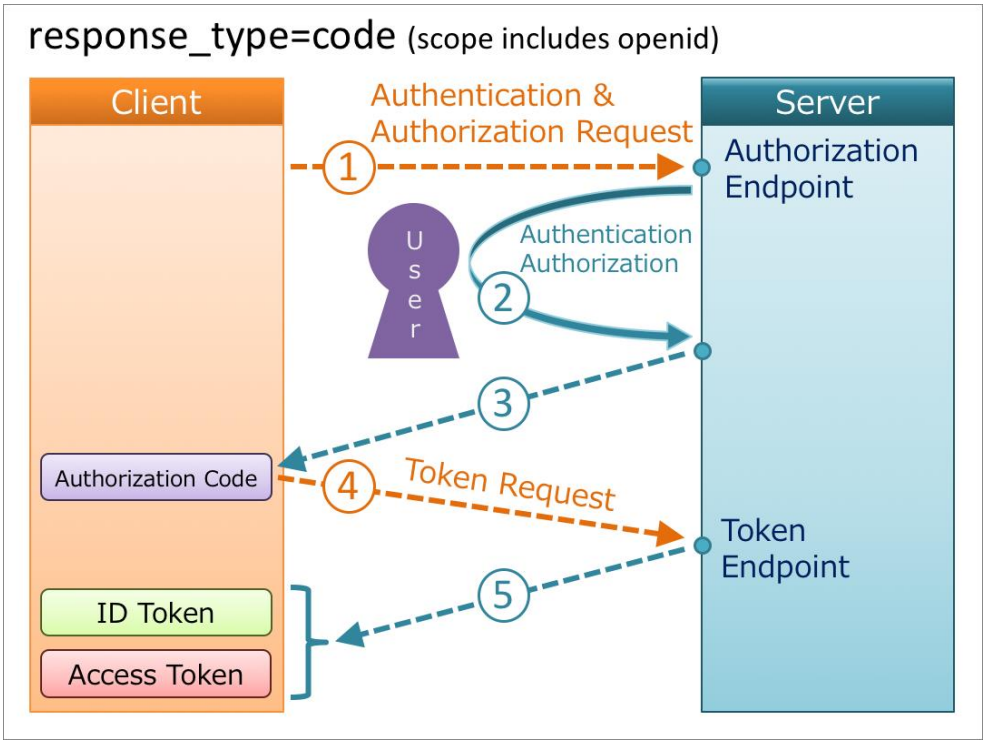
```
{
  "sub"       : "alice",
  "iss"       : "https://openid.c2id.com",
  "aud"       : "client-12345",
  "nonce"     : "n-0S6_WzA2Mj",
  "auth_time" : 1311280969,
  "acr"       : "c2id.loa.hisec",
  "iat"       : 1311280970,
  "exp"       : 1311281970
}
```

Η κεφαλίδα του Id Token, που αποτελείται από το JSON, τα claims και την υπογραφή, κωδικοποιείτε σε ένα base 64 URL-safe string, με σκοπό τον εύκολο διαμοιρασμό του, για παράδειγμα ως παράμετρος διεύθυνσης URL.

```
eyJhbGciOiJSUzI1NiIsImtpZCI6IjFlOWdkazcifQ.ewogImlzcyI6ICJodHRw
Oi8vc2VydmVyLmV4YW1wbGUuY29tIiwKICJzdWIiOiAiMjQ4Mjg5NzYxMDAxIiw
KICJhdWQiOiAiczZCaGRSa3F0MyIsCiAibm9uY2UiOiAibi0wUzZfV3pBMk1qIi
wKICJleHAiOiAxMzExMjgxOTcwLAogImlhdCI6IDEzMTEyODA5NzAKfQ.ggW8hZ
1EuVLuxNuuIJKX_V8a_OMXzR0EHR9R6jgdqrOOF4daGU96Sr_P6qJp6IcmD3HP9
9Obi1PRs-cwh3LO-p146waJ8IhehcwL7F09JdijmBqkvPeB2T9CJNqeGpe-gccM
g4vfKjkM8FcGvnzZUN4_KSP0aAp1tOJ1zZwgjxqGByKHiOtX7TpdQyHE5lcMiKP
XfEIQILVq0pc_E2DzL7emopWoaoZTF_m0_N0YzFC6g6EJbOEoRoSK5hoDalrcvR
YLSrQAZZKflyuVCyixEoV9GfNQC3_osjzw2PAithfubEEBLuVVk4XUVrWOLrLl0
nx7RkKU8NXNHq-rvKMzqg
```

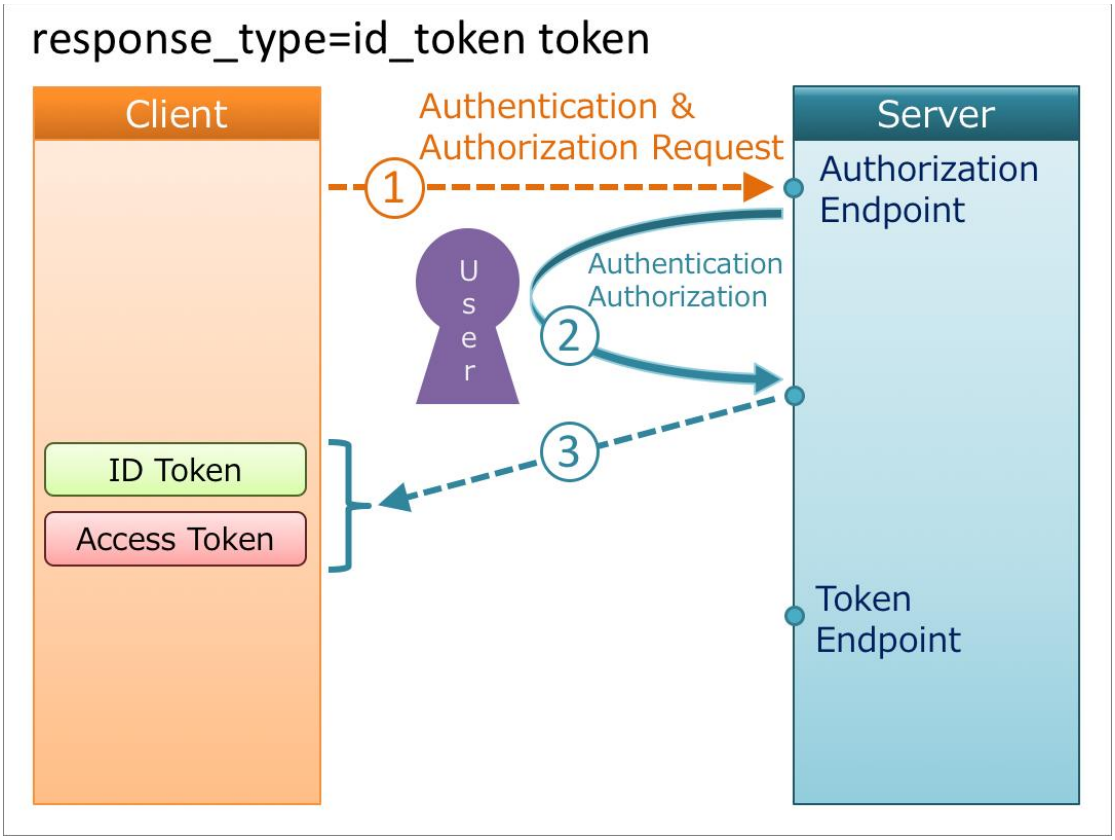
Το OpenId Connect υποστηρίζει παραλλαγμένες ροές (flows) του OAuth2.0. Παρακάτω παρουσιάζονται οι κυριότερες :

Authorization code flow – Χρησιμοποιείται κυρίως από εφαρμογές με web-server logic που επιτρέπουν επικοινωνία από το back-end. Λειτουργεί όπως μια παραδοσιακή ροή OAuth και έχει ως αποτέλεσμα την έκδοση ενός access token που επιστρέφεται στην εφαρμογή μέσω κλήσεων που πραγματοποιούνται στο back channel. Στη ροή αυτή, αντί να μεταδίδονται απευθείας τα στοιχεία του χρήστη, ο πάροχος στέλνει ένα ειδικό κωδικό μιας χρήσης (code) από το Authorization endpoint, που μπορεί να ανταλλάσσεται στο back-end της εφαρμογής (client) με ένα access token και ένα Id Token από το Token endpoint.



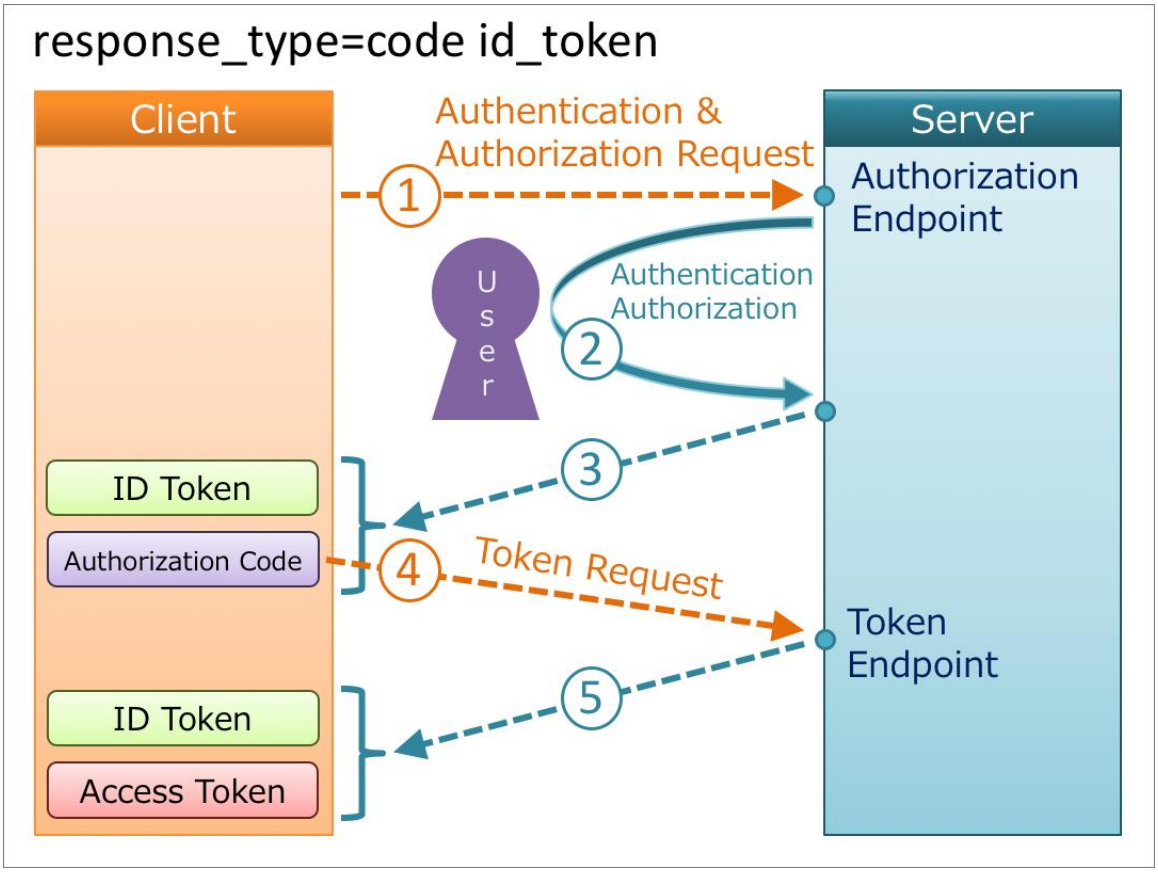
Εικόνα 3.20. Authorization code flow

Implicit flow – Χρησιμοποιείται κατεξοχήν από εφαρμογές browser . Όμοια με το αντίστοιχο flow του Oauth 2.0 η έκδοση των access και id token γίνεται απευθείας από το authorization endpoint, δηλαδή μέσω του user agent, κατόπιν ταυτοποίησης και εξουσιοδότησης. Σήμερα η χρήση του έχει αντικατασταθεί από το Authorization code flow η άλλες ασφαλέστερες επιλογές καθώς πλέον μέσω του πρωτόκολλου CORS επιτρέπονται τα request από browser εφαρμογές σε hosts εκτός του origin τους, εφόσον το επιτρέπουν, κάτι που καθιστά δυνατή την αξιοποίηση του token endpoint και κατά συνέπεια την χρήση του Authorization code flow από browser εφαρμογές.



Εικόνα 3.21. Implicit flow

Hybrid flow – Χρησιμοποιείται σπάνια, σε περιπτώσεις που μια εφαρμογή χρειάζεται πληροφορίες για τον τελικό χρήστη ενώ ακόμα δεν έχει αποκτηθεί το access token. Η λειτουργία του είναι παρόμοια με το Authorization code flow με την διαφορά ότι το Authorization endpoint δύναται να επιστρέψει μια ακόμα παράμετρο εκτός του authorization code, συνήθως ένα id token. Η δυνατότητα απόκτησης του id token μέσω και του authorization endpoint δίνει την δυνατότητα αντιμετώπισης επιθέσεων τύπου ‘code substitution’ αλλά μπορεί να εκθέσει ευαίσθητες πληροφορίες του τελικού χρήστη.



Εικόνα 3.22. Hybrid flow

3.6.8. Υλοποίηση σύνδεσης

Όπως αναφέρθηκε προηγουμένως για την σύνδεση σε ένα poll ο χρήστης (μαθητής) πρέπει πρώτα να τα ταυτοποιηθεί συμμετέχοντας σε μια διαδικασία ταυτοποίησης τριών παραγόντων (3-Factor Authentication). Σε αυτή την ενότητα θα αναλύσουμε πως υλοποιήθηκε αυτή η διαδικασία αναλύοντας τον κάθε παράγοντα ξεχωριστά.

3.6.9. Δαχτυλικό αποτύπωμα

Κατά την εκκίνηση της mobile εφαρμογής ο χρήστης θα δει μια οθόνη που του ζητά να δώσει το δαχτυλικό του αποτύπωμα ώστε να επαληθευθεί η ταυτότητα του .Για την υλοποίηση αυτής της διαδικασίας στο AndroidManifest.xml πρέπει αρχικά δηλώθηκε η άδεια για την χρήση του δαχτυλικού αποτυπώματος από την εφαρμογή.

```
<uses-permission android:name="android.permission.USE_FINGERPRINT" />
```

Παράδειγμα κώδικα 3.22. Android manifest permissions

Στην συνέχεια δημιουργήθηκε μια activity η οποία χρησιμοποιεί δύο System Services του Android (KeyGuardManager,FingerPrintManager) τα οποία παρέχουν όλες τις πληροφορίες που

σχετίζονται με τα δαχτυλικά αποτυπώματα της συσκευής που τρέχει η εφαρμογή αλλά και τους τρόπους με τους οποίους ξεκλειδώνει η συσκευή.

```
KeyguardManager keyguardManager = (KeyguardManager)GetSystemService(KeyguardSe
rvice);
FingerprintManager fingerprintManager = (FingerprintManager)GetSystemService(F
ingerprintService);
```

Παράδειγμα κώδικα 3.23. Fingerprint manager

Στο επόμενο βήμα ελέγχεται εάν έχει δοθεί η άδεια από τον χρήστη στην εφαρμογή να χρησιμοποιήσει το δαχτυλικό του αποτύπωμα άλλα και να τον ενημερώσει σε περίπτωση για τυχόν ρυθμίσεις που αποτρέπουν την χρήση του δαχτυλικού αποτυπώματος.

```
if (ActivityCompat.CheckSelfPermission(this, Manifest.Permission.UseFingerprin
t)
  != (int)Android.Content.PM.Permission.Granted)
  return;
  if (!fingerprintManager.IsHardwareDetected)
     Toast.MakeText(this, "FingerPrint authentication permission not enable", T
oastLength.Short).Show();
  else
  {
    if (!fingerprintManager.HasEnrolledFingerprints)
       Toast.MakeText(this, "Register at least one fingerprint in Settings", To
astLength.Short).Show();
      else
      {
        if (!keyguardManager.IsKeyguardSecure)
           Toast.MakeText(this, "Lock screen security not enable in Settings",
ToastLength.Short).Show();
        else
        {
          FingerprintHandler handler = new FingerprintHandler(this);
          handler.StartAuthentication(fingerprintManager, null);
        }
```

Παράδειγμα κώδικα 3.24. Έλεγχος άδειας

Εάν πληρούνται όλες οι προδιαγραφές καλείται η μέθοδος StartAuthentication της κλάσης FingerprintHandler οι οποία πρόκειται για τον listener που ‘διαβάζει’ τα δαχτυλικά αποτυπώματα.

```
internal class FingerprintHandler : FingerprintManager.AuthenticationCallback
{
  private Context AuthActivity;
  public FingerprintHandler(Context AuthActivity)
  {
    this.AuthActivity = AuthActivity;
  }
  internal void StartAuthentication(FingerprintManager fingerprintManager, Fin
gerprintManager.CryptoObject cryptoObject)
  {
    CancellationSignal cancellationSignal = new CancellationSignal();
    if (ActivityCompat.CheckSelfPermission(AuthActivity, Manifest.Permission.U
seFingerprint)
      != (int)Android.Content.PM.Permission.Granted)
```

```
      return;
    fingerprintManager.Authenticate(cryptoObject, cancellationSignal, 0, this,
 null);
   }
   public override void OnAuthenticationFailed()
   {
     Toast.MakeText(AuthActivity, "Fingerprint Authentication failed!",ToastLe
ngth.Long).Show();
   }
    public override void OnAuthenticationSucceeded(FingerprintManager.Authentic
ationResult result)
   {
     AuthActivity.StartActivity(new Intent(AuthActivity, typeof(MainActivity))
);
   }}
```

Παράδειγμα κώδικα 3.25. Fingerprint handler

3.6.10. One time password

Κατά την εγγραφή του χρήστη στον identity server δημιουργείται ένα Secret Key (HMACSHA256) το οποίο αποθηκεύεται στην βάση δεδομένων του identity server στα στοιχεία του χρήστη.

```
var hmac = new HMACSHA256();
var key = Convert.ToBase64String(hmac.Key);
var user = new ApplicationUser { UserName = Input.Email, Email = Input.Email ,
Key =key};
```

Παράδειγμα κώδικα 3.26. Δημιουργία secret key

Αυτόματα συνθέτετε ένα Json Object το οποίο αποθηκεύεται σε μορφή κειμένου σε ένα αντικείμενο ευρετηρίου TempData και περιέχει το email του χρήστη και το Secret Key.

```
//Create a JObject with Email and Shared Key
JObject job = new JObject();
job.Add("email", Input.Email);
job.Add("key", key);
//Save JObject to TempData
TempData["Key"] = job.ToString(Formatting.None);
await _signInManager.SignInAsync(user, isPersistent: false);
return Page();
```

Παράδειγμα κώδικα 3.27, Json Object

Κατά την ολοκλήρωση της εγγραφής στο frontend του identity server δημιουργείται ένα QR που περιέχει το Json κείμενο με τον κωδικό και το email.

```
<script language="JavaScript">
  $(document).ready(function () {
    new QRCode(document.getElementById("qrcode"), "@TempData["Key"]");});
</script>
```

Παράδειγμα κώδικα 3.28. Δημιουργία QR

Το επόμενο βήμα είναι η αποθήκευση του κλειδιού και του email του χρήστη στην mobile εφαρμογή. Διαβάζοντας το QR η εφαρμογή παίρνει την απαραίτητη πληροφορία από το κείμενο json.

```
string email ="";
string key="";
txtResult.Post(() => {
Vibrator vibrator = (Vibrator)GetSystemService(Context.VibratorService);
#pragma warning disable CS0618 // Type or member is obsolete
vibrator.Vibrate(1000);
#pragma warning restore CS0618 // Type or member is obsolete
var data = (JObject)JsonConvert.DeserializeObject(WebUtility.HtmlDecode(((Barc
ode)qrcodes.ValueAt(0)).RawValue));
email = data["email"].Value<string>();
key = data["key"].Value<string>();
```

Παράδειγμα κώδικα 3.29. QR scanning

Τελικά το κλειδί και το email αποθηκεύονται σε μια κρυπτογραφημένη βάση sqlite.

```
_keyChainHelper = new KeyChain.Net.XamarinAndroid.KeyChainHelper(() => this.Ap
plicationContext, "lol");
var dbFilePath = System.IO.Path.Combine(System.Environment.GetFolderPath(Syste
m.Environment.SpecialFolder.MyDocuments), "mysequredb.db3");
var platform = new SQLite.Net.Platform.XamarinAndroid.SQLitePlatformAndroid();
ISecureDatabase database = new MyDatabase(platform, dbFilePath);
List<string> emails = new List<string>();
var keySeed = _keyChainHelper.GetKey("MasterKey");
foreach (var item in database.SecureGetAll<UserModel>(keySeed))
{
  emails.Add(item.Email);
}
var user = new UserModel() { Email = email, Key = key, Id = Guid.NewGuid().ToS
tring() };
if (!emails.Contains(email))
{
  var inserted = database.SecureInsert<UserModel>(user, keySeed);
}
```

Παράδειγμα κώδικα 3.30. Αποθήκευση κλειδιού και email

Αφού το κλειδί και το email έχουν αποθηκευτεί η εφαρμογή δημιουργεί ανά 30 δευτερόλεπτα one time passwords με βάση το κλειδί και ενημερώνει το UI της εφαρμογής.

```
private static void SetTimer()
{
   // Create a timer with a two second interval.
   timer1 = new System.Timers.Timer(30000);
   // Hook up the Elapsed event for the timer.
   MainActivity main = new MainActivity();
   timer1.Elapsed += main.CreateOtp;
   timer1.AutoReset = true;
   timer1.Enabled = true;
}
```

Παράδειγμα κώδικα 3.31. Περιοδική δημιουργία otp

```
private void CreateOtp(Object source, ElapsedEventArgs e)
{
  TotpAuthenticationService totpAuthenticationService = new TotpAuthentication
Service(6, 30);
   try
   {
     RunOnUiThread(() => emails.Clear());
     foreach (var item in database.SecureGetAll<UserModel>(keySeed))
     {
       if (item.Email != "")
       {
          emails.Add(item.Email + " " + totpAuthenticationService.GenerateCode(
Convert.FromBase64String(item.Key)));
       }
     }
      RunOnUiThread(() => adapter.Clear());
      RunOnUiThread(() => adapter.AddAll(emails));
      RunOnUiThread(() => adapter.NotifyDataSetChanged());
     }
     catch (Exception exe) {
       System.Diagnostics.Debug.WriteLine(exe.ToString());
      }
}
```

Παράδειγμα κώδικα 3.32. Μέθοδος δημιουργίας otp

Όπως φαίνεται από τον κώδικα για την δημιουργία των otp δημιουργείται ένα αντικείμενο της κλάσης TotpAuthenticationService με παραμέτρους το μήκος του κωδικού και το time step και καλείται η μέθοδος GenerateCode η οποία δέχεται σαν παράμετρο το κλειδί και υλοποιεί την διαδικασία που περιεγράφηκε στην αρχή του κεφαλαίου.

Αφού λοιπόν έχει ενημερωθεί το UI ο χρήστης επιλέγοντας το poll στο οποίο θέλει να συνδεθεί και πατώντας τον λογαριασμό δημιουργείται ένα json text το οποίο περιέχει το otp, το roomid (id του poll) και το email του λογαριασμού που έχει επιλεγεί . Εν συνεχεία ανοίγει ένας browser στο οποίο ανοίγει ο MVC Client με ένα link που οδηγεί στην μέθοδο GET Preconnect του MVC Client η οποία δέχεται σαν όρισμα το json text που έχει δημιουργηθεί.

```
//Create JObject including email roomId and One Time Password
JObject job2 = new JObject();
job2.Add("email", email);
job2.Add("roomId", roomidView.SelectedItem.ToString());
job2.Add("otp", code);
string encodedjob2 = Uri.EscapeDataString(job2.ToString());
//Open Mobile Browser
var uri = Android.Net.Uri.Parse("https://localhost:44314/Home/Preconnect?incjs
on=" + encodedjob2);
var intent = new Intent(Intent.ActionView, uri);
```

Παράδειγμα κώδικα 3.33. Κλήση μεθόδου Preconnect

Στην συνέχεια η μέθοδος Preconnect αφού διαβάσει το Json Text ορίζει στο Session State του Http Context δύο values που αντιπροσωπεύουν το otp και το email του χρήστη και τέλος καλεί την GET μέθοδο ConnectAsync που εν τέλη θα μεταφέρει τον χρήστη στο poll που έχει επιλέξει αφού πρώτα του δοθεί εξουσιοδότηση μέσω του identity server.

```
public async Task<IActionResult> Preconnect(string incjson)
{
```

```
    AppModel model = Newtonsoft.Json.JsonConvert.DeserializeObject<AppModel>(i
ncjson);
    HttpContext.Session.SetString("otp", model.otp);
    HttpContext.Session.SetString("email", model.email);
    return RedirectToAction("ConnectAsync", new { username = model.email ,pass
word = model.roomId});
}
```

Παράδειγμα κώδικα 3.34. Μέθοδος Preconnect

Όπως αναφέραμε πριν για να μεταφερθεί ο χρήστης στο poll πρέπει πρώτα να του δοθεί εξουσιοδότηση. Έτσι κατά την κλήση της ConnectAsync ο χρήστης πρώτα μεταφέρεται στον Identity server για να εισάγει τα login credentials που με τα οποία έχει εγγραφεί ώστε να προχωρήσει η διαδικασία εξουσιοδότησης η οποία θα αναλυθεί στην επόμενη ενότητα. Κατά το redirect στον identity server το otp και το email θέτονται σαν Header values.

```
options.Events.OnRedirectToIdentityProvider = async n =>
{
  //Send otp and email on Redirect to Authorization Server
  var headerValue = n.HttpContext.Session.GetString("otp");
  var headerValue2 = n.HttpContext.Session.GetString("email");
  n.ProtocolMessage.SetParameter("otp", headerValue);
  n.ProtocolMessage.SetParameter("email", headerValue2);
  n.ProtocolMessage.Prompt = "login";
  await Task.FromResult(0);
};
```

Παράδειγμα κώδικα 3.35. On redirect event

Τέλος κατά την διαδικασία της επιβεβαίωσης των διαπιστευτηρίων του χρήστη ο identity server παίρνει τις παραμέτρους email και otp από τον header και με την χρήση του κοινού κλειδιού που έχει αποθηκευτεί στην βάση δεδομένων του κατά την διαδικασία της εγγραφής του χρήστη επικυρώνει το otp. Σε περίπτωση που είναι έγκυρο προχωράει στην σελίδα εισαγωγής των διαπιστευτηρίων του χρήστη.

```
var vm = await BuildLoginViewModelAsync(returnUrl);
string param1 = HttpUtility.ParseQueryString(returnUrl).Get("otp");
var otp = param1;
string param2 = HttpUtility.ParseQueryString(returnUrl).Get("email");
var email = param2;
HttpContext.Session.SetString("email", email);
vm.Username = email;
if (vm.IsExternalLoginOnly)
 {
   return RedirectToAction("Challenge", "External", new { provider = vm.Extern
alLoginScheme, returnUrl });
 }
TotpAuthenticationService totpAuthentication = new TotpAuthenticationService(6
, 30);
string hmackey = _context.Users.Where(a => a.Email == HttpContext.Session.GetS
tring("email")).Select(b => b.Key).FirstOrDefault();
byte[] keybytes = Convert.FromBase64String(hmackey);
if (totpAuthentication.ValidateCode(keybytes, Int32.Parse(otp), 2) == true)
{
return View(vm);
```

```
}
else
{
  return Content("One Time Password is invalid" + otp);}
```

Παράδειγμα κώδικα 3.36. Επαλήθευση otp

3.6.11. Εξουσιοδότηση και ταυτοποίηση

Για την μεταφορά του χρήστη στο poll ο χρήστης πρέπει πρώτα να μεταφερθεί στον authorization server, να δώσει τα διαπιστευτήρια του και με την επαλήθευση των στοιχείων του χρήστη και του otp και αφού του δοθούν τα απαιτούμενα access grants να μεταφερθεί τελικώς στην σελίδα του poll. Ο authorization server έχει στηθεί με βάση το IdentityServer4 Core, ενός Open Id Connect και Oauth2 framework για την ASP.NET Core. Αυτό το Framework μπορεί να παραμετροποιηθεί για την εκάστοτε χρήση και υποστηρίζει τα επιμέρους flow του Oauth2 και του OIDC που αναλύθηκαν . Για το σύστημα αυτό επιλέχθηκε το Hybrid Flow του OIDC

3.6.12. Υλοποίηση στο σύστημα

Αρχικά στα αρχεία startup.cs των API,MVC Client πρέπει να προστεθούν στα services η αρχή εξουσιοδότησης και η χρήση της ταυτοποίησης με Open Id Connect αντίστοιχα καθώς και οι επιθυμητές ρυθμίσεις όπως φαίνεται παρακάτω :

```
services.AddAuthentication("Bearer") // it is a Bearer token
        .AddIdentityServerAuthentication(options =>
        {
          //Identity Server URL
          options.Authority = "https://localhost:5000/";
          // make it true since we are  using https
          options.RequireHttpsMetadata = true;
          //api name which should be registered in IdentityServer
          options.ApiName = "api1";
        });
```

Παράδειγμα κώδικα 3.37. Προσθήκη αρχής εξουσιοδότησης στο API

```
.AddOpenIdConnect("oidc", options =>
 {
   options.Events.OnRedirectToIdentityProvider = async n =>
   {
     //Send otp and email on Redirect to Authorization Server
     var headerValue = n.HttpContext.Session.GetString("otp");
     var headerValue2 = n.HttpContext.Session.GetString("email");
     n.ProtocolMessage.SetParameter("otp", headerValue);
     n.ProtocolMessage.SetParameter("email", headerValue2);
     n.ProtocolMessage.Prompt = "login";
     await Task.FromResult(0);
```

```
    };
    options.SignInScheme = "Cookies";
    options.Events.OnTicketReceived = async (context) =>
    {
      context.Properties.ExpiresUtc = DateTime.UtcNow.AddSeconds(30);
    };
    options.Authority = "http://localhost:5000/";
    options.RequireHttpsMetadata = false;

      options.ClientId = "mvc";
      options.ClientSecret = "secret";
      //Hybrid flow
      options.ResponseType = "code id_token";
      options.SaveTokens = true;
      options.GetClaimsFromUserInfoEndpoint = true;
      options.Scope.Add("api1");
      options.Scope.Add("offline_access");
      options.ClaimActions.MapJsonKey("website", "website");
    });
```

Παράδειγμα κώδικα 3.38. Χρήση OIDC στον MVC Client

Στην συνέχεια πρέπει να δημιουργηθεί ένας client ο οποίος να αντιπροσωπεύει τον MVC Client στο αρχείο config.cs του IdentityServer που να ορίζει τα επιτρεπόμενα scopes και το API σαν resource.

```
public static IEnumerable<ApiResource> GetApis()
        {
            return new List<ApiResource>
            {
                new ApiResource("api1", "My API")
            };
        }
```

Παράδειγμα κώδικα 3.39. Προσθήκη του API στα resources

```
new Client
  {
    ClientId = "mvc",
    ClientName = "MVC",
    AllowedGrantTypes = GrantTypes.Hybrid,
    ClientSecrets =
    {
      new Secret("secret".Sha256())
    },
    RedirectUris           ={https://localhost:44314/signin-oidc},
    PostLogoutRedirectUris ={"https://localhost:44314/signout-callback-oidc"},
    AllowedScopes =
    {
      IdentityServerConstants.StandardScopes.OpenId,
      IdentityServerConstants.StandardScopes.Profile,
      "api1"
    },
```

```
    AllowOfflineAccess = true
  }
```

Παράδειγμα κώδικα 3.40. Register του MVC client

Κεφάλαιο 4.

Εγκατάσταση σε IIS Server

4.1. Περίληψη κεφαλαίου

Σε αυτό το κεφάλαιο θα δείξουμε πως το σύστημα μπορεί να εγκατασταθεί σε έναν iis server. Ξεκινώντας θα δείξουμε την προετοιμασία που πρέπει να γίνει προτού εκδώσουμε τα επιμέρους projects και μετέπειτα θα προχωρήσουμε με την έκδοση τους. Συνεχίζοντας θα δούμε πως τα projects θα στηθούν στον iis server και πως θα δημιουργηθούν οι βάσεις σε αυτόν. Το μηχάνημα στο οποίο θα στηθεί το σύστημα έχει εγκατεστημένα τα Windows Server 2012 και SQL Server Express 2017

4.2. Προετοιμασία των projects

Ανοίγοντας σε Visual Studio 2017 το solution του συνολικού project (Audience Response API) προχωράμε στις παρακάτω αλλαγές σε καθένα από τα τρία projects που περιέχει(AudienceResponseApi, IdentityServerAspNetIdentity, MainApp). Στην συνέχεια ανοίγουμε το project App1 για να προχωρήσουμε και στις αλλαγές στην mobile εφαρμογή

4.2.1. AudienceResponseAPI

Στο αρχείο **Startup.cs** ορίζουμε το URL στο οποίο σκοπεύουμε να φιλοξενήσουμε τον identity server.

```
options.Authority =  "https://snf-875402.vm.okeanos.grnet.gr:8080/";
//Identity Server URL
```

Στο αρχείο **appsettings json** ορίζουμε το connection string της βάσης του API.

```
"ConnectionStrings": {
  "DefaultConnection": "Server=.\\SQLEXPRESS;Database=ARSAPI;Trusted_Connectio
n=True;MultipleActiveResultSets=true"
},
```

4.2.2. IdentityServerAspNetIdentity

Στο αρχείο **config.cs** στο οποίο έχουμε κάνει register τον MVC Client (MainApp) ορίζουμε τα redirection link του.

```
RedirectUris ={"https://snf-875402.vm.okeanos.grnet.gr:8000/signin-oidc" },
PostLogoutRedirectUris ={"https://snf-875402.vm.okeanos.grnet.gr:8000/signout-
callback-oidc" },
```

Στο αρχείο **appsettings json** ορίζουμε το connection string της βάσης του Identity server.

```
{
  "ConnectionStrings": {
    "DefaultConnection": "Server=.\\SQLEXPRESS;Database=IDENTITYUSERS;Trusted_
Connection=True;MultipleActiveResultSets=true"
  }
}
```

4.2.3. MainApp

Στο αρχείο **Startup.cs** ορίζουμε το URL στο οποίο σκοπεύουμε να φιλοξενήσουμε τον identity server.

```
options.Authority = "https://snf-875402.vm.okeanos.grnet.gr:8080/";
```

Στο αρχείο **Helper -> Helper.cs** ορίζουμε το URL στο οποίο σκοπεύουμε να φιλοξενήσουμε το API.

```
Client.BaseAddress = new Uri("https://snf-875402.vm.okeanos.grnet.gr:8700/");
```

Στο αρχείο **Views -> Shared -> Layout.cshtml** ορίζουμε το URL στο οποίο σκοπεύουμε να φιλοξενήσουμε τον identity server (υπογραμμισμένο).

```
<li> <a href="https://snf-
875402.vm.okeanos.grnet.gr:8080/Identity/Account/Register">Register
</a>
</li>
</ul>
```

Στο αρχείο **Views -> Question -> AddQuestion.cshtml** ορίζουμε το URL στο οποίο σκοπεύουμε να φιλοξενήσουμε το API (υπογραμμισμένο).

```
$('#questionTitle').autocomplete({
  source: 'https://snf-875402.vm.okeanos.grnet.gr:8700/api/Search/search'
});
```

4.2.4. App1

Ανοίγουμε το αρχείο MainActivity.cs και κάνουμε τις παρακάτω αλλαγές :

Γραμμή 132

Ορίζουμε το URL στο οποίο σκοπεύουμε να φιλοξενήσουμε το API (υπογραμμισμένο).

```
string sUrl = "                                                    https://snf-
875402.vm.okeanos.grnet.gr:8700/api/VoteSessions/openSessions";
```

Γραμμή 247

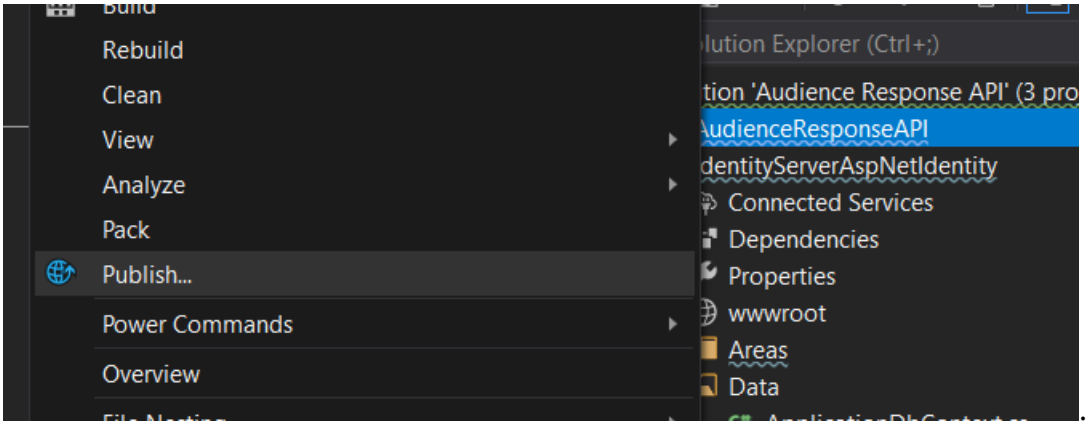
Ορίζουμε το URL στο οποίο σκοπεύουμε να φιλοξενήσουμε το API (υπογραμμισμένο).

```
var uri = Android.Net.Uri.Parse("https://snf-
875402.vm.okeanos.grnet.gr:8700/Preconnect?incjson=" + encodedjob2);
```

4.3. Έκδοση των projects

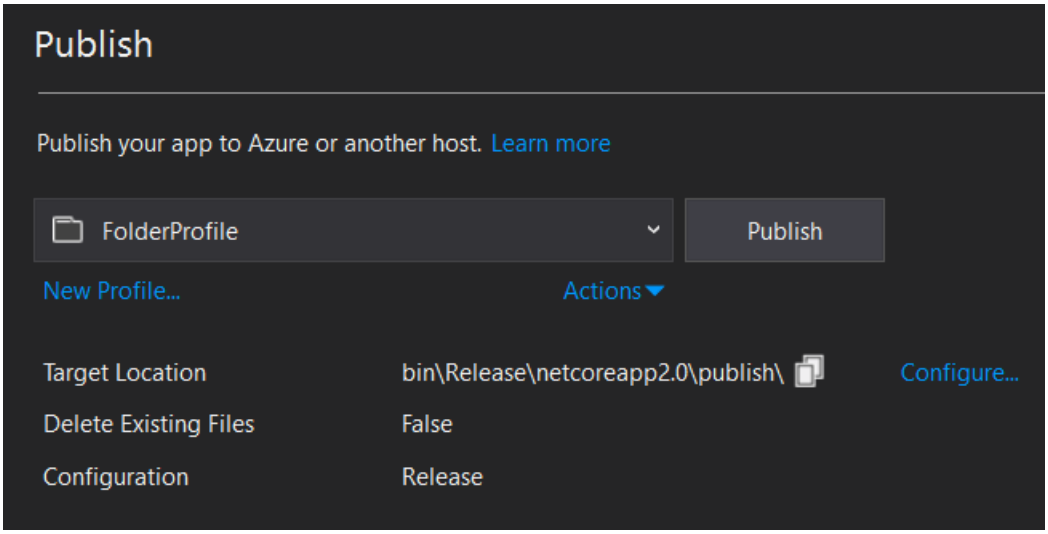
4.3.1. MVC Client, API, Identity server

Αφού έχουν γίνει οι παραπάνω αλλαγές ανοίγοντας σε Visual Studio 2017 το solution του συνολικού project (Audience Response API) κάνουμε publish σε τοπικό φάκελο όλα τα projects (AudienceResponseApi, IdentityServerAspNetIdentity, MainApp) πατώντας δεξί κλικ στο κάθε project και publish για να μεταβούμε στις επιλογές έκδοσης



Εικόνα 4.1. Μετάβαση στα publish options

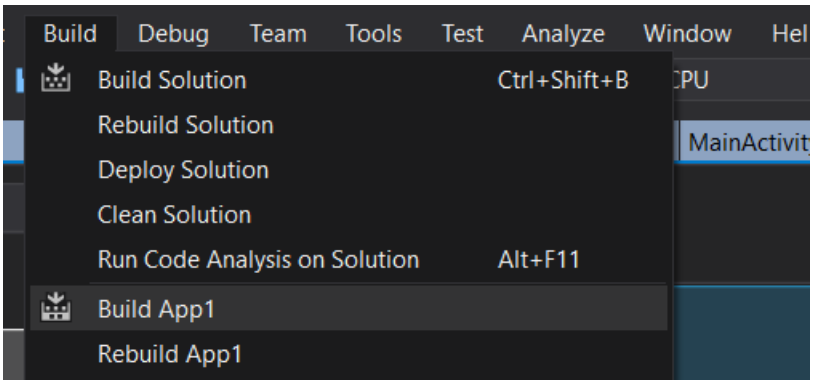
Αφού μεταβούμε στις επιλογές επιλέγουμε FolderProfile και τον φάκελο στον οποίο θέλουμε να αποθηκευτούν τα published αρχεία και τέλος πατάμε Publish . Αφού τελειώσει επιτυχώς η διαδικασία μπορούμε να βρούμε τα αρχεία στον φάκελο που επιλέξαμε.



Εικόνα 4.2. Publishing

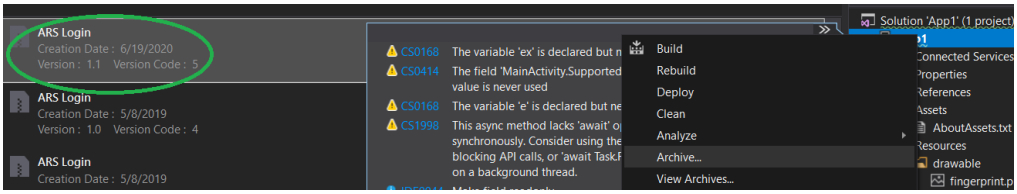
4.3.2. Mobile Application

Αφού επαναλάβουμε την διαδικασία για όλα τα projects του project Audience Response API,ανοίγουμε το Xamarin project App1 για να δημιουργήσουμε το apk της mobile εφαρμογής. Το πρώτο βήμα είναι να κάνουμε build την εφαρμογή με επιλεγμένη την σήμανση Release.



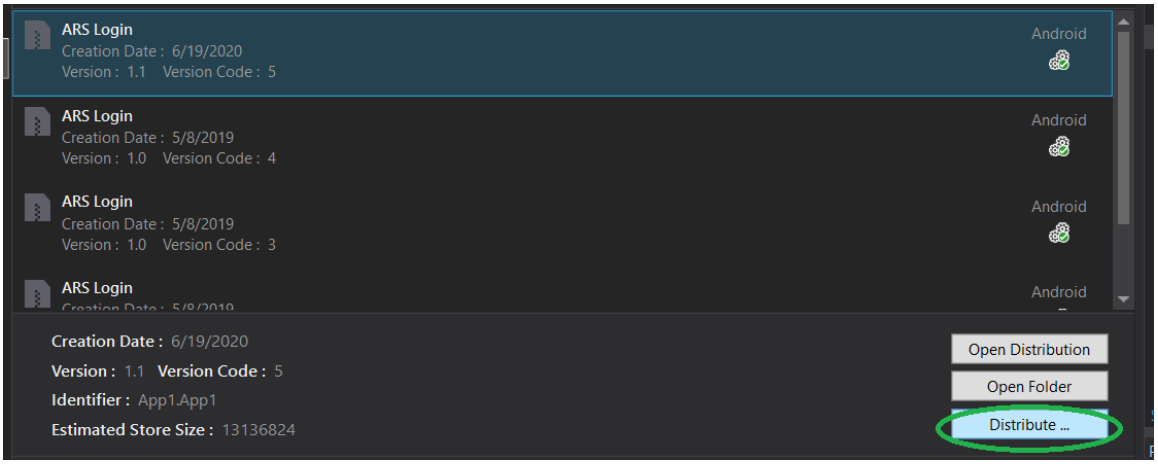
Εικόνα 4.3. Building App1

Αφού το build ολοκληρωθεί επιτυχώς κάνουμε δεξί κλικ στο project app1 και πατάμε Archive.. για να εμφανιστούν τα αρχειοθετημένα build της εφαρμογής.



Εικόνα 4.4. Archive App1

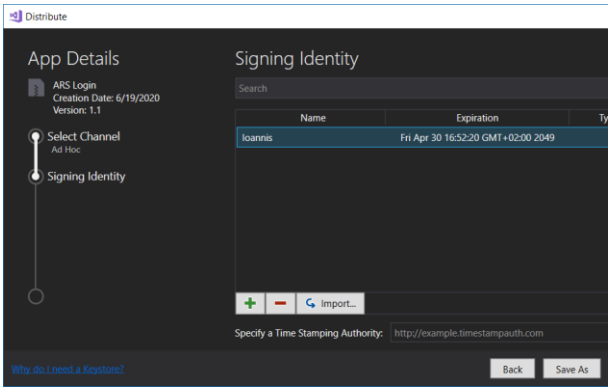
Επιλέγουμε το πιο πρόσφατο και πατάμε Distribute.



Εικόνα 4.5. Distribute App1

Στην οθόνη επιλογών που ανοίγει επιλέγουμε Ad Hoc. Στην συνέχεια επιλέγουμε sign in identity και τέλος πατάμε Save As για να επιλέγουμε τον φάκελο που θα δημιουργηθεί το apk.

Εικόνα 4.6, Επιλογή Ad Hoc

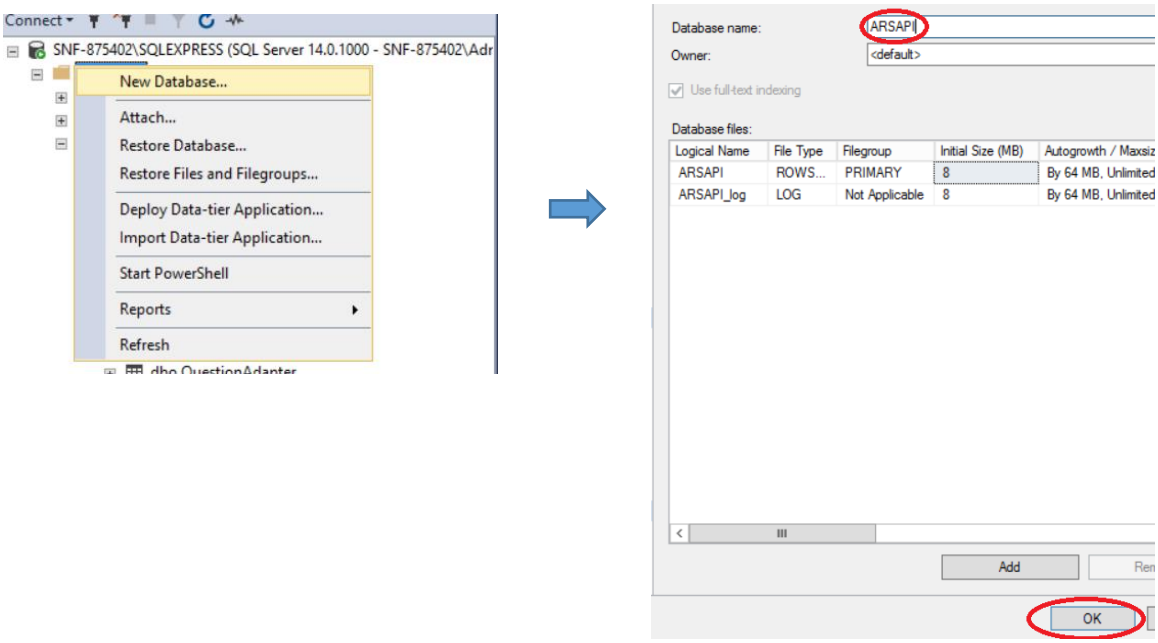


Εικόνα 4.7, Αποθήκευση apk

4.4. Έκδοση στον IIS server

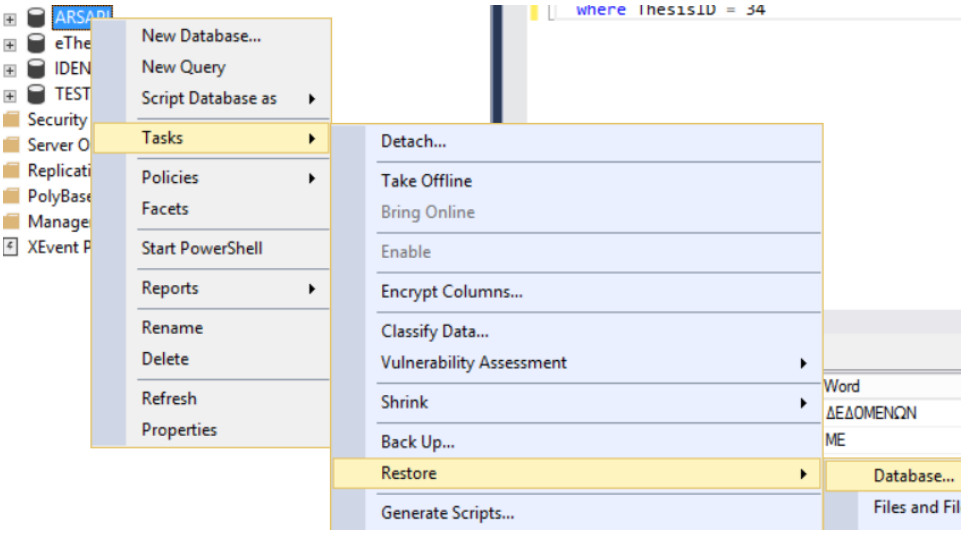
4.4.1. Δημιουργία βάσεων δεδομένων

Από τον τοπικό SQL Server έχουν εξαχθεί δύο bak αρχεία, για την βάση του API και την βάση του Identity server. Ανοίγοντας το SQL server management studio στο μηχάνημα που θα γίνει η εγκατάσταση του συστήματος πηγαίνουμε στο instance στο οποίο θέλουμε να δημιουργήσουμε τις βάσεις και στον φάκελο **Databases** πατάμε δεξί κλικ και **New Database...** . Ξεκινώντας με την βάση του API την ονομάζουμε αντίστοιχα με το όνομα που έχουμε δώσει στο connection string του API προηγουμένως και πατάμε OK.



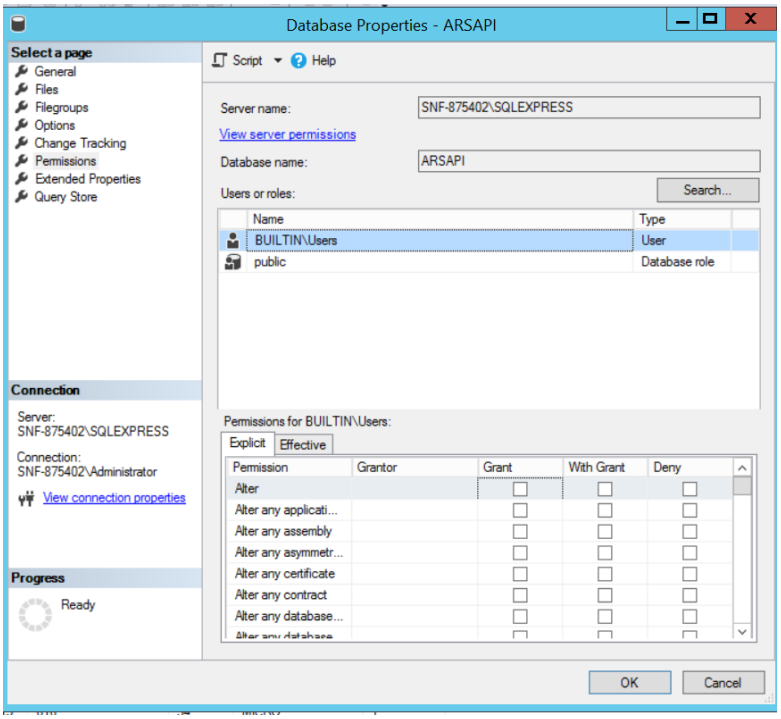
Εικόνα 4.8. Δημιουργία βάσης δεδομένων

Αφού η βάση δημιουργηθεί με δεξί κλικ στη νέα βάση επιλέγουμε **Tasks -> Restore -> Database.** Στην οθόνη επιλογών που ανοίγει στην ομάδα πεδίων **source** επιλέγουμε **device**. Προσθέτουμε το αρχείο bak που αφορά τη βάση του API πατάμε OK και περιμένουμε να ολοκληρωθεί η διαδικασία.



Εικόνα 4.9. Restore βάσης δεδομένων

Η βάση πλέον έχει δημιουργηθεί πλήρως. Για να είναι προσβάσιμη από το site που θα δημιουργήσουμε έπειτα, κάνουμε δεξί κλικ στη βάση και επιλέγουμε **properties**. Από εκεί μεταβαίνουμε στην καρτέλα **Permissions** και εισάγουμε την ομάδα χρηστών **BUILTIN/Users.** Την ίδια διαδικασία επαναλαμβάνουμε για την βάση δεδομένων του Identity server.

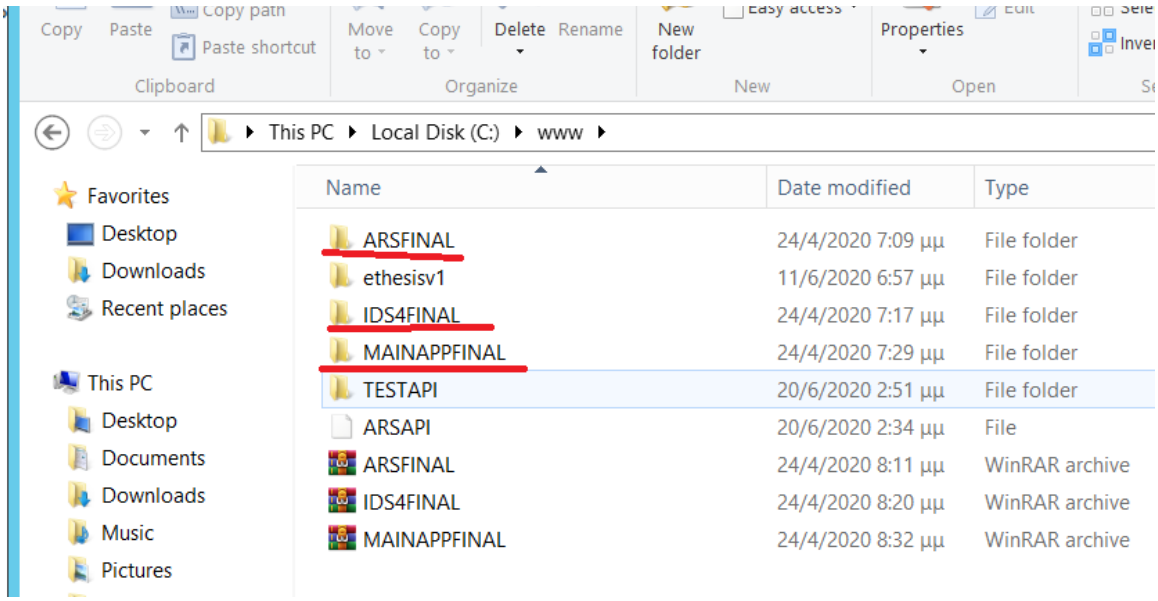


Εικόνα 4.10. Προσθήκη permissions

4.4.2. Δημιουργία των site

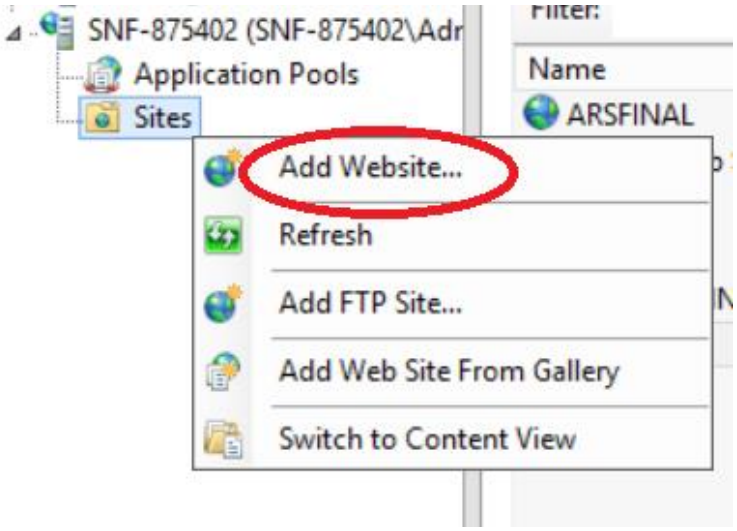
Μετά την επιτυχή δημιουργία των βάσεων προχωράμε στην δημιουργία των τριών site (MVC Client, API, Identity server). Στον server πηγαίνουμε στον C: και φτιάχνουμε έναν νέο φάκελο που τον

ονομάζουμε www. Μέσα σε αυτόν αντιγράφουμε τους τρεις φακέλους με τα published αρχεία που δημιουργήσαμε στην ενότητα 4.3.



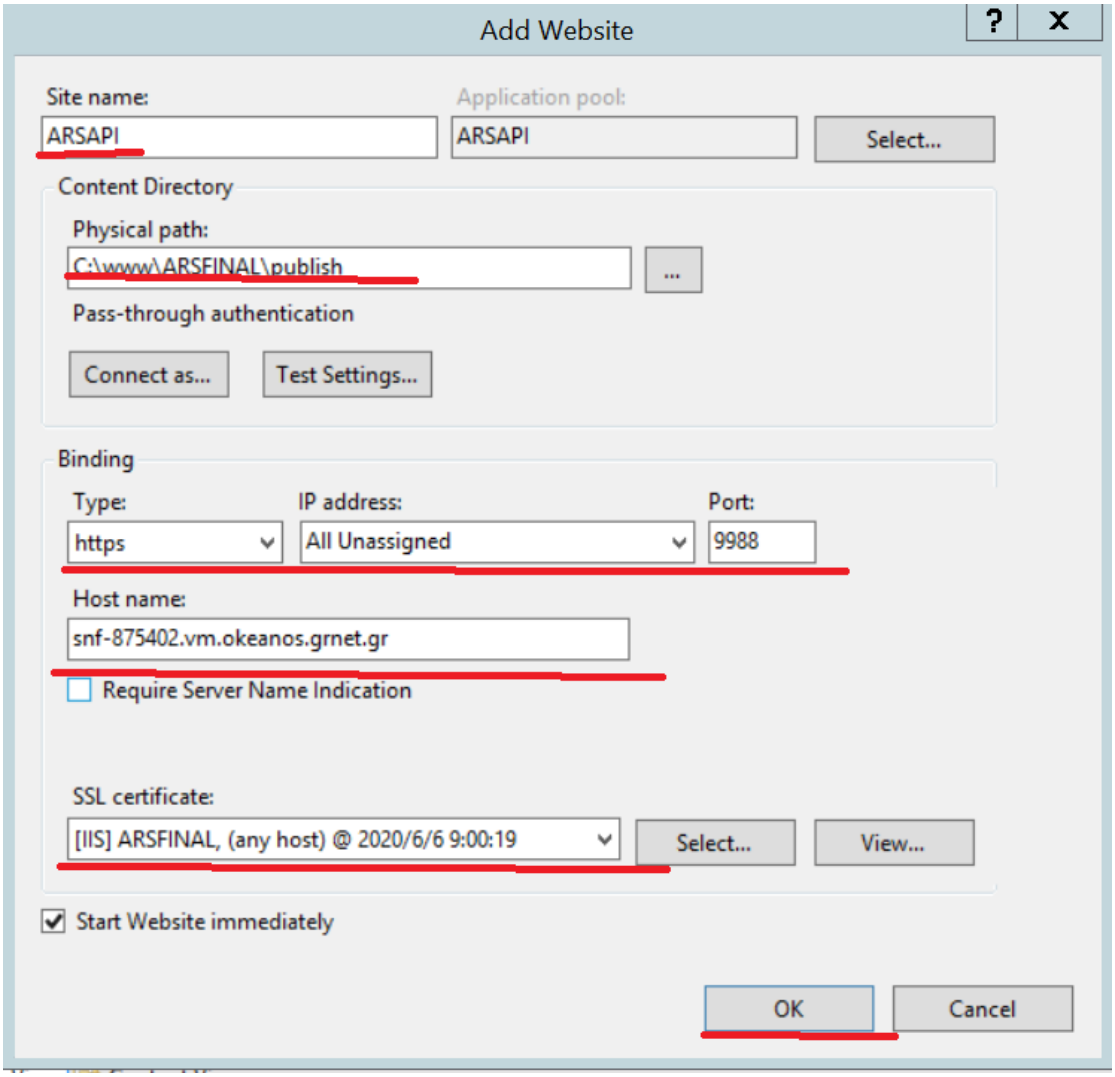
Εικόνα 4.11. Φάκελος www

Στη συνέχεια ανοίγουμε των IIS manager και στα **Sites** πατάμε δεξί κλικ και **Add Website**.



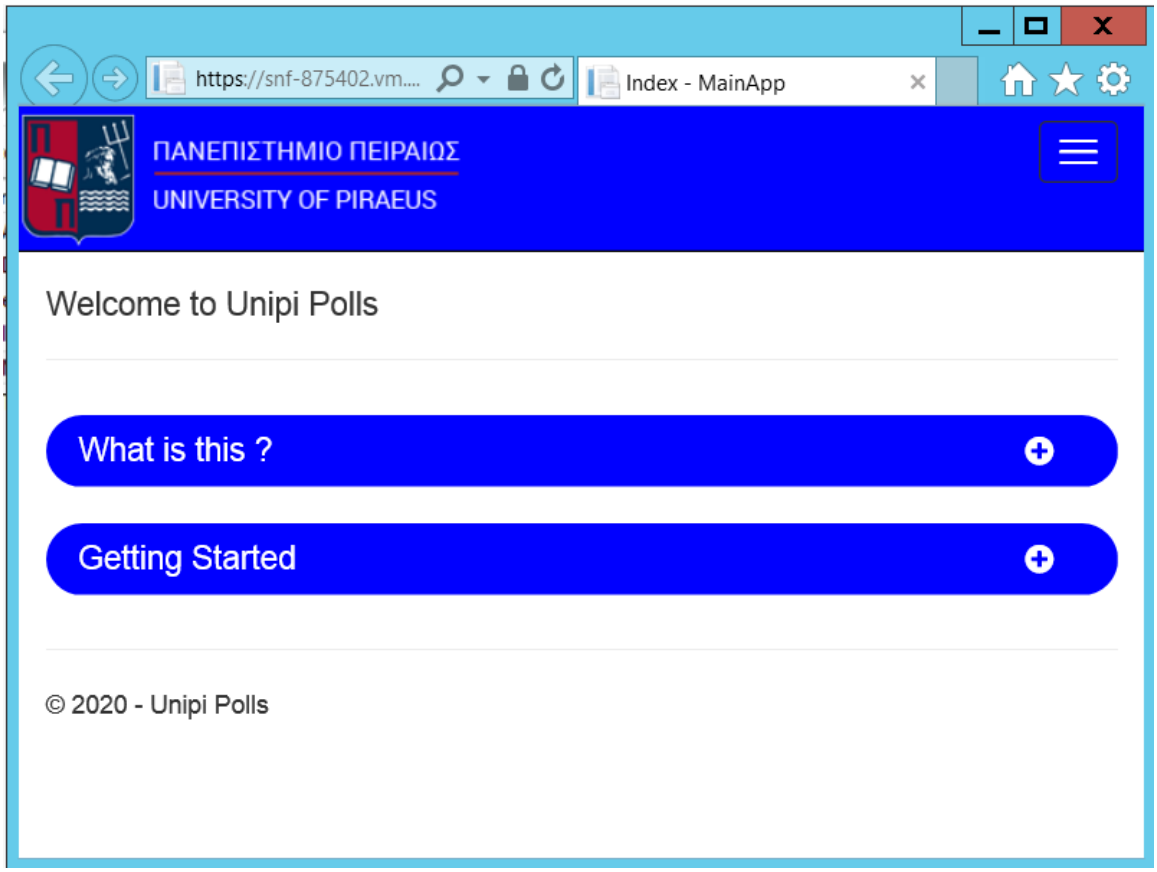
Εικόνα 4.12. Δημιουργία Website

Στην οθόνη που ανοίγει, ξεκινώντας με το API, συμπληρώνουμε το όνομα του site και ορίζουμε το path που βρίσκονται τα published αρχεία του API. Εν συνεχεία στην ομάδα πεδίων binding επιλέγουμε https και το port που έχουμε ορίσει κατά την προετοιμασία των project. Στο πεδίο Host name βάζουμε το όνομα του μηχανήματος και τέλος στο πεδίο SSL Certificate επιλέγουμε ένα πιστοποιητικό το οποίο είτε έχουμε δημιουργήσει είτε έχουμε προμηθευτεί.



Εικόνα 4.13. Επιλογές δημιουργίας

Επαναλαμβάνουμε την διαδικασία για τον MVC Client και τον Identity server. Αφού έχουμε δημιουργήσει τα τρία site πατώντας δεξί κλικ -> Manage Website -> Browse….. στο site του MVC Client μπορούμε να το δούμε να φορτώνει στον browser.



Εικόνα 4.14. MVC Client

Κεφάλαιο 5.

Συμπεράσματα και μελλοντικές επεκτάσεις

5.1. Συμπεράσματα

Η παρούσα διατριβή είχε σαν στόχο το να προτείνει ένα εκπαιδευτικό σύστημα απόκρισης κοινού το οποίο να ακολουθεί τις σύγχρονες απαιτήσεις ασφαλείας των BYOD πολιτικών σε συνδυασμό με ευέλικτη εγκατάσταση ανάλογα τις προτιμήσεις του εκάστοτε ιδρύματος. Συμπερασματικά δίνει απάντηση στα δύο παρακάτω καίρια ζητήματα :

1. **Μη εξουσιοδοτημένη πρόσβαση στο σύστημα.**

Η ταυτοποίηση τριών παραγόντων που προτείνεται στο παρόν έγγραφο καταφέρνει να θωρακίσει το σύστημα από ενέργειες που στοχεύουν στην μη εξουσιοδοτημένη απόκτηση πρόσβασης στο σύστημα και συνεπώς στο δίκτυο του ιδρύματος. Πιο συγκεκριμένα :

i. Η χρήση του δαχτυλικού αποτυπώματος στην mobile εφαρμογή αποτρέπει την πρόσβαση σε μη εξουσιοδοτημένα πρόσωπα ακόμα και σε περίπτωση κλοπής της συσκευής
ii. Η χρήση του one time password εμποδίζει τις επιθέσεις τύπου phising.
iii. Η Χρήση του API και του πρωτόκολλου Oauth2 ως access control framework δίνει μεγάλες δυνατότητες διαχείρισης και προστασίας των resources του συστήματος από τρίτα φυσικά η μη πρόσωπα.

2. **Χαμηλές απαιτήσεις hardware και ευέλικτη εγκατάσταση**

Το προτεινόμενο σύστημα μπορεί να εγκατασταθεί με την ίδια ευκολία είτε σε on premise υποδομές είτε στο azure.Σε περίπτωση on premise εγκατάστασης, λόγω της ανάπτυξης στο cross platform περιβάλλον της .net core, υπάρχει δυνατότητα επιλογής της πλατφόρμας (Linux, Mac, Windows) ανάλογα με τις ανάγκες χωρίς κανένα περιορισμό. Τέλος η χρήση του συστήματος μπορεί να γίνει αποκλειστικά από ιδιόκτητες κινητές συσκευές από καθηγητές και μαθητές με ασφάλεια χωρίς επιπλέον κόστος για hardware.

5.2. Μελλοντικές επεκτάσεις

Η παρούσα διατριβή αποτελεί ένα proof of concept το οποίο με συγκεκριμένες επεκτάσεις θα μπορούσε να γίνει ένα πλήρως λειτουργικό σύστημα το me σκοπό την χρήση από διάφορα εκπαιδευτικά ιδρύματα. Μερικές από αυτές τις επεκτάσεις παρουσιάζονται επιγραμματικά παρακάτω :

i. Δημιουργία mobile εφαρμογής για κινητές συσκευές ios.
ii. Διακριτοί ρόλοι καθηγητή και μαθητή για την πρόσβαση στις διάφορες λειτουργίες με την χρήση του API και του Identity server.
iii. Περισσότερες επιλογές παρουσίασης αποτελεσμάτων .
iv. Βελτίωση του user interface.
v. Χρήση του φορέα ταυτοποίησης του εκάστοτε εκπαιδευτικού ιδρύματος.

5.3. Γνώσεις που αποκτήθηκαν

Η ανάπτυξη του παρόντος συνέβαλε στην απόκτηση γνώσεων και δεξιοτήτων που αφορούν το σύγχρονο μοντέλο ανάπτυξης web εφαρμογών όπως αυτές που παρουσιάζονται επιγραμματικά παρακάτω :

i. Ανάπτυξη web εφαρμογών και REST API's στο σύγχρονο περιβάλλον .Net Core.
ii. Ανάπτυξη μιας real time web εφαρμογής και εμβάθυνση στις τεχνικές υλοποίησης της (Web sockets, SSE κλπ.) .
iii. Πρακτική εφαρμογή του πρωτόκολλου OAuth2 και παραμετροποίηση του Identity Server.
iv. Κατανόηση του αλγόριθμού TOTP και χρήση του για την δημιουργία one time passwords.

Βιβλιογραφία

Adrian Drury, R. A., χ.χ. *BYOD: an emerging market trend in more ways than one, Employee attitudes to work/life balance drive BYOD behavior.* s.l., Logicalis white paper, ovum.

Anon., 2013. Best Practices for Enabling BYOD in Education. *Palmer Research white Paper.*

Anon., 2013. Education Institutions at Forefront of BYOD. *Infosecurity magazine.*

Anon., χ.χ. BYOD in Education, The Cisco Bring-Your-Own Device Solution for Education: Getting Mobile Devices Simply and Securely Connected.

Avanade Research &Insights, G. S., χ.χ. Dispelling Six Myths of Consumerization of IT.

Boardman, D. E., 1968. *The use of immediate response systems in junior college,* University of California: PhD thesis.

Caldwell, J. E., 2007. *Clickers in the Large Classroom : Current Research and Best-Practice Tips.* s.l., In CBE Life Sci Educ. 2007.

Dahlstrom, E., 2012. Ecar study of undergraduate students and information technology.

Dufresne, R. J. G. W. J. L. W. J. M. J. P. a. W. L., 1996. Classtalk: A classroom communication system for active learning. *Journal of Computing in Higher Education,* 7(2), pp. 3-37.

Ebner, M., 2009. Introducing live microblogging: how single presentations can be enhanced by the mass. *Journal of research in innovative teaching,* 2(1), pp. 91-101.

Ebner, M. N. W. a. S. M., 2012. Have They Changed? Five Years of Survey on Academic Net-Generation. *World Conference on Educational Media and Technology,* Τόμος 8, p. 21–31.

Froehlich, H. P., 1963. What about classroom Communicators?. *Audiovisual communication review,* 11(3), pp. 19-16.

Judson, E. a. S. D., 2002. Learning from Past and Present: Electronic Response Systems in College Lecture Halls.. *Journal of Computers in Mathematics and Science Teaching,* 21(2), p. 167–18.

Liu, T.-c. L. J.-k. W. H.-y. a. C. T.-w., 2003. *The Features and Potential of Interactive Response System..* Hong Kong, In Proceedings of International Conference on Computers in Education (ICCE), pages 315–322.

Logicalis, 2012. *BYOD : an emerging market trend in more ways than one. Technical report,* s.l.: s.n.

Networks, B., χ.χ. The Impact of BYOD in Education.

Roschelle, J., 2003. Unlocking the learning value of wireless mobile devices. *Journal of Computer Assisted Learning,* p. 260–272.

surd, T. O., 2013. 10 Mobile Device Management Leaders That Help IT Control BYOD.

Thomson, G., 2012. FEATURE BYOD : enabling the chaos. *Network Security,* Τόμος 2, pp. 5-8.

whitehouse.gov, 2012. Bring Your Own Device A Toolkit to Support federal Agencies Implementing Bring Your Own Device (BYOD) Programs.