ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΙΚΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Λογισμικό Οργάνωσης Ιατρικών Ραντεβού & Επισκέψεων σε Περιβάλλον Υπολογιστικού Νέφους

Χασιαλής Βασίλειος

Εισηγητής: Βασίλειος Μάμαλης, Καθηγητής

ΑΘΗΝΑ

ΟΚΤΩΒΡΙΟΣ 2019

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Λογισμικό Οργάνωσης Ιατρικών Ραντεβού & Επισκέψεων σε Περιβάλλον Υπολογιστικού Νέφους

Χασιαλής Βασίλειος
A.M. cse41758

Εισηγητής:

Βασίλειος Μάμαλης, Καθηγητής

Ημερομηνία εξέτασης 14/10/2019

ΠΕΡΙΛΗΨΗ

Στόχος είναι να χρησιμοποιηθούν τεχνολογίες Cloud και να διερευνηθούν νέες τεχνολογίες αιχμής (C#, AJAX, NodeJS, Angular). Πρόκειται για λογισμικό το οποίο καλείτε να καλύψει τις ανάγκες οργάνωσης και αποθήκευσης στοιχείων αλλά και του ιατρικού φακέλου των ασθενών με τη ιεραρχημένη διαχείριση προνομίων, ακολουθώντας σύστημα ανάθεσης ρόλων και εξουσιοδοτήσεων. Επιπρόσθετα θα υπάρχει πλήρης ιχνηλασιμότητα των ενεργειών κάθε χρήστη. Επίσης θα έχει την δυνατότητα δημιουργίας και παρακολούθησης ραντεβού.

ABSTRACT

The purpose of current thesis is to make use of Cloud technologies and look into cutting edge technologies (C#, AJAX, NodeJS, Angular). It is a software that is called upon to meet the needs of data management and storage as well as the medical records of patients with hierarchical privilege management, following a role assignment and authorization system. In addition, there will be complete traceability of each user's actions. It will also be able to create and track appointments.

Πίνακας Περιεχομένων

ΚΑΤΑΛΟΓΟΣ ΣΧΗΜΑΤΩΝ

ΚΑΤΑΛΟΓΟΣ ΕΙΚΟΝΩΝ

ΣΥΝΤΟΜΟΓΡΑΦΙΕΣ

RAM: Random Access Memory

DOM: Document Object Model

HTML: HyperText Markup Language

API: Application Program Interface

ICD: Implantable Cardioverter-Defibrillator

Μέρος 1ο

Κεφάλαιο 1ο

Εισαγωγή

1.1 Σκοπός της εργασίας

Την τελευταία δεκαετία έχει παρατηρηθεί ραγδαία ανάπτυξη στις τεχνολογίες πληροφορικής και επικοινωνιών. Ένα από τα επιτεύγματα αυτής της ανάπτυξης αποτελούν και οι τεχνολογίες του υπολογιστικού νέφους. Οι τεχνολογίες αιχμής χρησιμοποιούνται από τις εταιρείες με κύριο σκοπό τη μείωση του κόστους για την αποθήκευση και την επεξεργασία των πληροφοριών που διαχειρίζονται. Υπό αυτό το πρίσμα, οι εταιρείες κάνουν χρήση των διαφόρων μοντέλων υπολογιστικού νέφους για τη βελτιστοποίηση του κόστους και της παρεχόμενης ποιότητας υπηρεσιών.
Η παρούσα εργασία αποσκοπεί στην ενδελεχή παρουσίαση των τεχνολογιών του υπολογιστικού νέφους (Cloud Computing) και την επιρροή τους στον κλάδο της ιατρικής. Πιο συγκεκριμένα, παρουσιάζεται η χρήση των τεχνολογιών του υπολογιστικού νέφους ως βασικό εργαλείο για τη διαχείριση του ιατρικού ηλεκτρονικού φακέλου (Electronic Medical Record - EMR).

1.2 Ιστορική αναδρομή

Η ανάλυση της έννοιας του υπολογιστικού νέφους καθορίζει και την ιστορία του. Ο ορισμός του υπολογιστικού νέφους υποδηλώνει την παροχή υπηρεσιών με γνώμονα τις οντότητες που αποθηκεύουν και επεξεργάζονται πληροφορίες, σε κάποια ασαφή γεωγραφική περιοχή. Επιπρόσθετα, το γεγονός της δυναμικής δέσμευσης πόρων, χωρικών και υπολογιστικών, για την παροχή υπηρεσιών, αποτελεί την ειδοποιό διαφορά με τα συστήματα που υπήρχαν και διαμόρφωσαν το κλίμα για την έννοια του υπολογιστικού νέφους.

Η βασική διαφορά από τις τεχνολογίες που υπήρχαν μέχρι τη στιγμή της εμφάνισης του υπολογιστικού νέφους είναι ότι οι τελευταίες προσφέρονται με τέτοιο τρόπο ώστε κανείς να μη γνωρίζει που ακριβώς είναι αποθηκευμένα τα δεδομένα, όπως επίσης και οι υπολογιστικοί πόροι που χρησιμοποιούνται για την επεξεργασία των δεδομένων.
Η ύπαρξη του υπολογιστικού νέφους αν και αρκετά πρόσφατη, τοποθετείται ως ιδέα χρονικά περίπου το 1950. Οι ανάγκες των χρηστών για την πρόσβαση σε υπολογιστικούς πόρους, μια εποχή όπου οι υπολογιστές καταλάμβαναν τεράστιο όγκο και ήταν ιδιαίτερα ακριβοί, οδήγησε τα σχολεία και τους οργανισμούς στη διαδικασία να εγκαταστήσουν μεγάλους εξυπηρετητές. Είναι προφανές ότι δεν θα μπορούσαν να υπάρχουν ατομικοί υπολογιστές, παρά μόνο τερματικά. Για την επίλυση αυτού του προβλήματος εγκαταστάθηκαν μεγάλοι εξυπηρετητές, στους οποίους συνδεόταν με τερματικά οι χρήστες. Με αυτό τον τρόπο οι εταιρείες κατάφεραν να εφαρμόσουν ένα μοντέλο στο οποίο διαμοιραζόταν η ανάγκη του υπολογιστικού φόρτου και γινόταν η αναγωγή σε ένα και μόνο εξυπηρετητή.
Αργότερα, περί το 1970, σχεδόν μετά από 20 χρόνια, παρουσιάστηκε η έννοια της εικονικής μηχανής (Virtual Machine) από την εταιρεία IBM. Χάρη στην τεχνολογία των εικονικών μηχανών έγινε δυνατή η δημιουργία διακριτών υπολογιστικών μηχανών οι οποίες παρασιτούν στην ίδιο φυσικό περιβάλλον – μηχανή.
Το επόμενο λογικό βήμα της εικονοποίσης έγινε με τη διασύνδεση του τελευταίου στο διαδίκτυο. Με αυτή την ενέργεια κατέστη δυνατή η παροχή των υπηρεσιών εικονοποίησης σε όλους τους χρήστες. Έτσι, ενώ μέχρι πρότινος ήταν ακατόρθωτο για μια εταιρεία να αγοράσει τους δικούς της ανεξάρτητους εξυπηρετητές, ήταν πολύ πιο εύκολο να τους νοικιάσει. Με αυτό τον τρόπο λοιπόν μια εταιρεία που θα ήθελε να προσφέρει 10 διαφορετικές υπηρεσίες, μπορούσε πολύ εύκολα να νοικιάσει έναν εξυπηρετητή και να τον χωρίσει με τέτοιο τρόπο ώστε να είναι δυνατή η προσφορά των αντίστοιχων 10 υπηρεσιών.

Όσο το κόστος του υλικού μειωνόταν, τόσοι περισσότεροι χρήστες είχαν πρόσβαση σε υπηρεσίες εικονικοποίησης. Τελικά η μείωση του κόστους οδήγησε στην ανάγκη για διαμοιρασμό των μηχανημάτων που προσφερόταν στους διάφορους εξυπηρετητές. Με αυτό τον τρόπο αναπτύχθηκε και εξελίχθηκε το υπολογιστικό νέφος.
Σημαντική συμβολή στην εξέλιξη και τη δημιουργία του υπολογιστικού νέφους

αποτέλεσε και η δημιουργία λογισμικού με το οποίο ήταν δυνατή η παρακολούθηση των πόρων του συστήματος καθώς και των εικονικών μηχανών το οποίο αυτό φιλοξενούσε. Αυτό το λογισμικό ονομάστηκε επόπτης (hypervisor). Το πλεονέκτημα αυτού του λογισμικού ήταν ότι μπορούσε να εγκατασταθεί σε διαφορετικά φυσικά μηχανήματα και να παρουσιάζει τους διαθέσιμους πόρους σαν να ήταν ένα μηχάνημα μόνο του. Ουσιαστικά λοιπόν αυτό το λογισμικό εξομοιώνει όλες τις λειτουργίες ενός φυσικού μηχανήματος, του επεξεργαστή του, καθώς και ενός εικονικού μηχανήματος. Το πλεονέκτημα του είναι ότι μπορεί να διαχειρίζεται και να λειτουργεί πολλά εικονικά μηχανήματα παράλληλα. Έτσι λοιπόν δημιουργήθηκε η λογική της προσθήκης φυσικών μηχανημάτων στο ένα εικονικό μεγαλύτερο μηχάνημα και ο διαμοιρασμός των πόρων αυτών στους χρήστες.

Τελειώνοντας αυτή την ενότητα θα μπορούσαμε να πούμε ότι το υπολογιστικό σύννεφο είναι ένα μοντέλο για την δυναμική παροχή πόρων από ένα ενιαίο διαθέσιμο σύνολο, οι οποία μπορούν να διατίθενται με ελάχιστη παρέμβαση του παρόχου.

1.3 Τεχνολογίες αιχμής

Οι τεχνολογίες που αξιοποιήθηκαν στα πλαίσια της παρούσας εργασίας και θα αναλυθούν εκτενώς στο επόμενο κεφάλαιο, βασίζονται σε ένα σύνολο εργαλείων τα οποία δίνουν τη δυνατότητα για ανάπτυξη εφαρμογών σε διαδικτυακό περιβάλλον. Πιο συγκεκριμένα χρησιμοποιούνται τόσο για την ανάπτυξη του front-end μέρους της εφαρμογής, που αποτελεί και τη διεπαφή με το χρήστη, όσο και την ανάπτυξη του back-end, που αποτελεί ουσιαστικά την κινητήριο δύναμη της εφαρμογής, αλλά και την ίδια την υποδομή του υπολογιστικού σύννεφου η οποία ήταν απαραίτητη για την διεκπεραίωση της παρούσας εργασίας.

Η ανάπτυξη της διεπαφής με τον χρήστη σε ένα διαδικτυακό περιβάλλον, απαιτεί την χρήση μιας γλώσσας η οποία να μπορεί να εκτελεστεί στον ίδιο τον πελάτη. Συνήθως τέτοιου είδους γλώσσες όπως η javascript, είναι κατάλληλες για την επίτευξη αυτού του σκοπού. Παρόλαυτά, η διεπαφή του χρήστη δεν είναι τίποτε περισσότερο από τον γραφική απεικόνιση της επεξεργασίας που γίνεται στα κύρια αρθρώματα του λογισμικού, τα οποία λειτουργούν στο backend κομμάτι της εφαρμογής. Για την πραγματοποίηση αυτού του κομματιού απαιτούνται κατά κύριο λόγο γλώσσες, οι

οποίες εκτελούνται στον εξυπηρετητή και έχουν ως στόχο την ουσιαστική επεξεργασία των πληροφοριών από το διαδίκτυο και τον ίδιο τον χρήστη. Τέτοιες γλώσσες είναι για παράδειγμα η C++, JAVA, .NET κλπ. Φυσικά η επιλογή της κατάλληλης γλώσσας εξαρτάται από τις απαιτήσεις της ίδιας της εφαρμογής αναφορικά με το πλήθος των χρηστών και φυσικά την ταχύτητα εκτέλεσης. Έτσι για παράδειγμα μια εφαρμογή η οποία εκτελεί κάποια πραγματικού χρόνου λειτουργία, απαιτεί συνήθως ταχύτερα πλαίσια λειτουργίας, συγκριτικά με κάποια άλλη εφαρμογή η οποία δεν απαιτεί πραγματικού χρόνου επεξεργασία δεδομένων.

Τέλος, η σημαντικότερη τεχνολογία που χρησιμοποιείται στα πλαίσια της παρούσας εργασίας αφορά τη διασύνδεση των προαναφερθέντων τεχνολογιών με το υπολογιστικό νέφος.

Επί της ουσίας αξιοποιείται η χρήση μια πλατφόρμας ειδικά σχεδιασμένης για το υπολογιστικό νέφος, η οποία διατίθεται τόσο για δημόσια όσο και για ιδιωτική χρήση παρέχοντας στους ενδιαφερόμενους να δημιουργούν εικονικές μηχανές τις οποίες μπορούν να συνδέσουν με διάφορες εφαρμογές.

Κεφάλαιο 2ο

Το υπολογιστικό Νέφος

Το υπολογιστικό νέφος είναι η αποθήκευση, η επεξεργασία και η χρήση δεδομένων από απομακρυσμένους υπολογιστές στους οποίους εξασφαλίζεται πρόσβαση μέσω του διαδικτύου. Πολλοί άνθρωποι χρησιμοποιούν σήμερα το υπολογιστικό νέφος δίχως καν να το συνειδητοποιούν. Υπηρεσίες όπως το διαδικτυακό ηλεκτρονικό ταχυδρομείο ή τα κοινωνικά δίκτυα συχνά βασίζονται στην τεχνολογία του υπολογιστικού νέφους. Για τους επαγγελματίες χρήστες της τεχνολογίας των πληροφοριών το υπολογιστικός νέφος σημαίνει μεγάλη ευελιξία όσον αφορά τις ανάγκες υπολογιστικής ισχύος. Για παράδειγμα όποτε διαπιστώνεται αυξημένη χρήση μιας υπηρεσίας, μέσω του υπολογιστικού νέφους είναι πολύ απλό να προστεθεί επιπλέον δυναμικό σε αυτή, κάτι για το οποίο θα απαιτείτο πολύ περισσότερος χρόνος εάν μια εταιρία υποχρεωνόταν να εγκαταστήσει νέες μηχανές στο δικό της κέντρο δεδομένων.

2.1 Μορφές Υπολογιστικού Νέφους

Οι λύσεις του υπολογιστικού νέφους μπορούν να διαχωριστούν βάσει 2 μεγάλων κριτηρίων. Το πρώτο κριτήριο αφορά την οντότητα που κατέχει τις υποδομές και το δεύτερο κριτήριο έχει σχέση με το μοντέλο της υπηρεσίας που προσφέρεται.
Σχετικά **με την πρώτη κατηγορία**, οι λύσεις του υπολογιστικού νέφους μπορούν να διαχωριστούν σε:
Δημόσιο Σύννεφο: Αναφέρεται σε λύσεις οι οποίες προσφέρονται στους πελάτες, δημόσια, μέσω του διαδικτύου. Το δημόσιο σύννεφο έχει το πλεονέκτημα ότι επιτρέπει την επεκτασιμότητα και το διαμοιρασμό των πόρων, πράγμα το οποίο είναι πολύ δύσκολο να επιτευχθούν από ένα οργανισμό.
Ιδιωτικό Σύννεφο: Στη συγκεκριμένη κατηγορία συγκαταλέγονται λύσεις οι οποίες έχουν περιορισμό στην πρόσβαση. Αυτό επιτυγχάνεται, είτε παρέχοντας την υπηρεσία μέσω του διαδικτύου αλλά επιτρέποντας την πρόσβαση σε εξουσιοδοτημένους

χρήστες, είτε χρησιμοποιώντας κάποιο ιδιωτικό δίκτυο. Αξίζει να σημειωθεί ότι το ιδιωτικό σύννεφο παρέχεται κυρίως από συγκεκριμένες εταιρείες οι οποίες κατέχουν και τον εξοπλισμό.

Το εικονικό ιδιωτικό σύννεφο: είναι η τομή των δύο προηγούμενων κατηγοριών. Συνήθως ένα εικονικό ιδιωτικό σύννεφο αποτελείται από εξοπλισμό ο οποίος ανήκει σε κάποια εταιρεία, αλλά βρίσκεται σε δημόσιες ή διαμοιραζόμενες υποδομές. Σε αυτή την κατηγορία είναι καλό να αναφερθεί ότι οι διάφοροι χρήστες έχουν πλήρως απομονωμένους πόρους χρησιμοποιώντας ιδιωτικά υποδίκτυα και είναι διασυνδεδεμένοι χρησιμοποιώντας εικονικά ιδιωτικά δίκτυα ή κρυπτογραφημένα κανάλια επικοινωνίας.

Το κοινοτικό σύννεφο: Περιλαμβάνει όλες εκείνες τις λύσεις, οι οποίες έχουν ως στόχο να χρησιμοποιηθούν όπως το ιδιωτικό σύννεφο, αλλά εμπλέκονται πολλοί διαφορετικοί πελάτες οι οποίοι ανήκουν στην ίδια κοινότητα. Για παράδειγμα, οι πελάτες οι οποίοι βρίσκονται μέσα σε μια βιομηχανία ή σε έναν οργανισμό, αλλά οι υπηρεσίες και τα τμήματα που εξυπηρετούνται είναι διαφορετικά. Αυτό το μοντέλο σύννεφου εφαρμόζεται κατά κύριο λόγο όταν υπάρχουν ευαίσθητα δεδομένα και φυσικά οι πελάτες πρέπει να δεχθούν ένα μικρό ρίσκο για τη χρήση της συγκεκριμένης υπηρεσίας.

Αναφορικά **με τη δεύτερη κατηγορία**, τα μοντέλα του υπολογιστικού νέφους μπορούν να διαχωριστούν ως εξής:

Υποδομή ως υπηρεσία (Infrastructure as a service - IAAS) η οποία αναφέρεται στις υπηρεσίες υπολογιστικού νέφους στις οποίες ο πάροχος παρέχει την υπολογιστική υποδομή, όπως φερειπείν οι εικονικές μηχανές και ο αποθηκευτικός χώρος τα οποία διαθέτει στους πελάτες μέσω του διαδικτύου. Σε αυτή τη μορφή υπολογιστικού νέφους ο πάροχος συνήθως διαθέτει τα δικά του κέντρα δεδομένων (data centers), τα οποία περιέχουν το υλικό όπου αποθηκεύεται και συντηρείται.

Ο σκοπός αυτού του μοντέλου είναι να δώσει την ελευθερία στον πελάτη να επιλέξει μια πλήρως επεκτάσιμη αρχιτεκτονική μέσω του υπολογιστικού σύννεφου η οποία να εξυπηρετεί και να καλύπτει πλήρως τις ανάγκες του. Έτσι για παράδειγμα ο κάθε πελάτης έχει να διαλέξει μεταξύ ενός μεγάλου πλήθους και διαφόρων ειδών

δρομολογητών, αναχωμάτων ασφαλείας και εξυπηρετητών. Επιπλέον, είναι σε θέση να καθορίσει πλήρως το είδος των πόρων αναφορικά με το είδος της απόδοσης τους σε ταχύτητα επεξεργαστή, αποθηκευτικό χώρο και μνήμη RAM.
Η ενοικίαση των προαναφερθέντων στοιχείων δίνουν την ευελιξία στον πελάτη να αυξομειώσει τους πόρους που χρειάζεται ανάλογα με τις ανάγκες, δίνοντας του πλήρη ελευθερία και εκτοξεύοντας τη σχέση ποιότητας υπηρεσιών και κόστους για την απόκτηση τους.

Πλατφόρμα ως υπηρεσία (Platform as a service - PAAS) το οποίο αναφέρεται στην περίπτωση όπου ο πάροχος προσφέρει το υλικό και τα εργαλεία ώστε να γίνει η ανάπτυξη λογισμικού που απαιτείται από τον πελάτη. Εδώ θα πρέπει να σημειωθεί πως και σε αυτή την περίπτωση ο πάροχος διαθέτει το σύνολο του υλικού και του λογισμικού το οποίο βρίσκεται στις εγκαταστάσεις του. Στην προκειμένη περίπτωση ο πελάτης ωφελείται γλυτώνοντας φόρτο για την εγκατάσταση των απαραίτητων εργαλείων. Επιπλέον, σε αυτό το μοντέλο συνήθως ο πελάτης διαθέτει τις δικές του εγκαταστάσεις υλικού, αλλά αξιοποιεί το μοντέλο της πλατφόρμας για συγκεκριμένες υπηρεσίες και την ανάπτυξη λογισμικού. Είναι σημαντικό να τονιστεί ότι σε αυτή την περίπτωση ο πάροχος συντηρεί και διατηρεί ένα πλήθος εργαλείων τα οποία μπορεί να καταναλώσει ο πελάτης, απλώς με την εγκατάσταση τους, και να τα χρησιμοποιήσει. Έτσι, ο πελάτης δεν ασχολείται καθόλου με την υποδομή και το υλικό, αλλά εστιάζει στο είδος της υπηρεσίας και της πλατφόρμας που αξιοποιεί, από τον πάροχο. Παραδείγματα για αυτό το είδος υπολογιστικού σύννεφου αποτελούν τα λειτουργικά συστήματα,το μεσισμικό κ.α όπως για παράδειγμα οι βάσεις δεδομένων καθώς και πλατφόρμες οι οποίες αξιοποιούνται για την ανάπτυξη λογισμικού.

Λογισμικό ως υπηρεσία (Software as a service - SAAS) το οποίο δίνει μεγάλη έμφαση στην ίδια την εφαρμογή. Η βασική διαφορά με το προηγούμενο είδος μοντέλου είναι ακριβώς ότι το προηγούμενο εστίαζε στην έννοια της πλατφόρμας, ενώ αυτό εστιάζει απευθείας στο λογισμικό. Αυτό το είδος υπολογιστικού σύννεφου είναι άρρηκτα συνδεδεμένο με την έννοια των μοντέλων λογισμικού τα οποία διατίθενται δυναμικά κατά βούληση. Το συγκεκριμένο είδος μοντέλου προσιδιάζει με τον πάροχο υπηρεσιών εφαρμογής ο οποίος προσφέρει μέσω διαδικτύου στους πελάτες

πρόσβαση σε εφαρμογές και σχετικές υπηρεσίες οι οποίες διαφορετικά θα έπρεπε να βρίσκονται στον πελάτη. Έτσι λοιπόν οι πελάτες αποκτούν πρόσβαση μέσω διαδικτύου σε μια εφαρμογή η οποία έχει εγκατασταθεί με σκοπό να γίνει η παροχή της μέσω του μοντέλου SAAS. Αντιστοίχως όταν η εφαρμογή ενημερώνεται, τότε γίνεται καθολική ενημέρωση σε όλους τους πελάτες.

2.2 Εργαλεία δημιουργίας & διαχείρισης υπολογιστικού νέφους

Όπως ήδη έχει αναφερθεί υπάρχει ποικιλία οργανισμών που παρέχουν υπηρεσίες υπολογιστικού νέφους, ωστόσο για αυτούς που θέλουν πραγματικά ένα ιδιωτικό υπολογιστικό νέφος υπάρχουν λύσεις όπως το **openstack** και το **cloudstack** με τις οποίες κάποιος μπορεί να δημιουργήσει και να διαχειριστεί κατά το δοκούν το δικό του ιδιωτικό ή μη νέφος.

2.2.1 Openstack

Η ανάγκη για διαχείριση των διαφορετικών πόρων στο σύννεφο οδήγησε στη δημιουργία κάποιων συστημάτων για τη διαχείριση τους. Μία από αυτές τις πλατφόρμες είναι το openstack.

Το openstack είναι μία ανοιχτού λογισμικού πλατφόρμα διαχείρισης για το σύννεφο. Επιπρόσθετα υπάρχει η τάση να σχετίζεται βιβλιογραφικά το openstack με λειτουργικό σύστημα.

Η δημιουργία του openstack προέκυψε από την ανάγκη να ενοποιηθούν οι διαφορετικές τεχνολογίες για τη διαχείριση των πόρων στο σύννεφο και αρχικά ήταν συνδεδεμένη με το πρόγραμμα nova, το οποίο ονομαζόταν nova-networking. Δυστυχώς η έλλειψη ευελιξίας και χαρακτηριστικών στο προηγούμενο πρόγραμμα (project), οδήγησε βαθμηδόν στην υιοθέτηση ενός αυτόνομου προγράμματος (project).

Η αυτονομία του openstack networking ή διαφορετικά neutron, οδήγησε στην παροχή των απαραίτητων διεπαφών για τον προγραμματισμό εφαρμογών (API). Οι προαναφερθείσες διεπαφές προσφέρουν ένα μεγάλο φάσμα επιλογών αναφορικά με

τη χρήση τους. Έτσι για παράδειγμα ο χρήστης είναι σε θέση να δημιουργήσει πόρτες, διεπαφές, δρομολογητές, ομάδες ασφαλείας κλπ.

Η διαχείριση και η ενσωμάτωση του openstack μπορεί να γίνει πολύ εύκολα με τη χρήση κατάλληλων πρόσθετων (plugin). Τα τελευταία βοηθούν για την ενσωμάτωση της οποιασδήποτε τεχνολογίας δικτύου με το openstack για την υλοποίηση οποιουδήποτε αριθμού πορτών, δρομολογητών και διεπαφών. Το σημαντικότερο στοιχείο σε αυτή την περίπτωση δεν είναι ο τρόπος που θα δημιουργηθεί η υποδομή, αλλά η δυνατότητα που θα παράσχει ο εκάστοτε πάροχος για την ενοποίηση με το openstack. Η τελευταία είναι δυνατή μόνο με την προϋπόθεση της υποστήριξης του Neutron API.

Τέλος, αξίζει να αναφερθεί ότι ο κάθε πάροχος υλοποιεί με διαφορετικό τρόπο τα πρόσθετα (plugins) με τα οποία είναι δυνατή η ενσωμάτωση με το openstack. Έτσι για παράδειγμα ο πάροχος Arista παρέχει ολοκλήρωση με σκοπό την παραμετροποίηση των εικονικών switch για την υποστήριξη του openstack. Στον αντίποδα, η VMWare παρέχει ολοκλήρωση με το openstack μέσω εικονικών switch. Επιπλέον, πολλά από αυτά τα πρόσθετα προσφέρουν ενσωμάτωση με το SDN, καθώς και με λύσεις οπτικοποίησης των δεδομένων

2.2.2 CloudStack

Το cloudstack αποτελεί μια ανοιχτού λογισμικού πλατφόρμα η οποία έχει τις βάσεις της στη γλώσσα προγραμματισμού JAVA. Η πλατφόρμα στοχεύει στην ανάπτυξη και τη διαχείριση της υποδομής στο σύννεφο ώς υπηρεσία (IAAS). Η πλατφόρμα δίνει τη δυνατότητα για την ανάπτυξη λύσεων στο δημόσιο, το ιδιωτικό και το υβριδικό σύννεφο.

Η απαρχή του cloudstack έγινε το 2008 μέσω του cloud.com. Η Citrix αγόρασε το cloudstack τον ιούλιο του 2011. Αργότερα, το 2012 η Citrix παραχώρησε το project στην Apache Software Foundation (ASF), έχοντας την άδεια χρήσης Apache 2.0. Από τον Μάρτιο του 2013 το συγκεκριμένο project έγινε ένα από τα κύρια project της Apache. Υπό αυτό το πρίσμα, πολλές εταιρείες υιοθέτησαν το cloudstack για την ανάπτυξη και τη διαχείριση λύσεων στο σύννεφο, με σκοπό τη διαχείριση των υποδομών.

Αναφορικά με την αρχιτεκτονική του cloudstack, οι φυσικοί πόροι είναι οργανωμένοι

και διαχειρίσιμοι ακολουθώντας μια ιεραρχική δομή. Στο κατώτατο μέρος αυτής της δομής παρασιτούν υπολογιστικές συσκευές όπως εξυπηρετητές και αποθηκευτικός χώρος. Στο κατώτατο στρώμα ακολουθείται η ακολουθία της συστοιχίας, η οποία δημιουργείται με τη συνένωση των διαφορετικών εξυπηρετητών. Στην προαναφερθείσα αρχιτεκτονική, ο αποθηκευτικός χώρος είναι διαμοιραζόμενος μεταξύ των εξυπηρετητών. Το επόμενο επίπεδο δημιουργεί ένα Pod χρησιμοποιώντας πολλαπλές συστοιχίες οι οποίες συνδυάζονται με ένα switch επιπέδου 2. Ανεβαίνοντας ακόμα ένα επίπεδο, τα διαφορετικά
pods συνδυάζονται με ένα switch επιπέδου 3 για να σχηματίσουν μια ζώνη (zone). Στο τελευταίο επίπεδο οι διαφορετικές ζώνες συνδυάζονται με σκοπό τη δημιουργία μιας περιοχής (region). Είναι σημαντικό να αναφερθεί ότι όλοι οι προηγούμενοι πόροι είναι εύκολα διαχειρίσιμοι μέσω ενός αντίστοιχου εξυπηρετητή.

Αναφορικά με τα διαφορετικά συστατικά που αναφέραμε στην προηγούμενη παράγραφο, ένας εξυπηρετητής αποτελεί ένα φυσικό υπολογιστικό μηχάνημα το οποίο έχει δικό του αποθηκευτικό χώρο. Τα φυσικά μηχανήματα είναι διαχειρίσιμα μέσω ενός hypervisor για την επίτευξη της διαδικασίας της εικονικοποίησης. Έπειτα η συστοιχία αποτελείται από εξυπηρετητές οι οποίοι ανήκουν σε ένα κοινό υπολογιστικό pool, κατέχοντας ένα κοινό αποθηκευτικό χώρο και παρασιτώντας στο ίδιο υποδίκτυο. Ακολούθως τα pods αποτελούνται από διαφορετικές συστοιχίες οι οποίες συνδυάζονται με switch τύπου 2. Είναι καλό να αναφερθεί πώς τα pods δεν είναι ορατά στον τελικό χρήστη. Ακολούθως ο διαχωρισμός των ζωνών ωφελεί στη απομόνωση των πόρων αλλά και στην επίτευξη του πλεονασμού τους. Επιπλέον οι διάφορες ζώνες οριοθετούνται γεωγραφικά με τη χρήση της έννοιας της περιοχής. Τέλος ο εξυπηρετητής διαχείρισης χρησιμοποιείται για τη διαχείριση του συνόλου των πόρων στην υποδομή του σύννεφου. Ένας εξυπηρετητής διαχείρισης μπορεί να υποστηρίξει περίπου 10.000 εξυπηρετητές.

Κεφάλαιο 3ο

Τεχνολογίες Αιχμής

3.1 Το .NET core

Το .NET core είναι ένα ελεύθερο ανοιχτού κώδικα πλαίσιο ανάπτυξης λογισμικού το οποίο μπορεί να χρησιμοποιηθεί σε περιβάλλοντα Windows, Linux και MacOS και αποτελεί επί της ουσίας τον διάδοχο του πλαισίου .NET. Το πλαίσιο αυτό , δημιουργήθηκε με γνώμονα την επίλυση βασικών προβλημάτων που είχαν εισαχθεί μέχρι εκείνη τη στιγμή. Τέτοια προβλήματα αφορούσαν την κεντρικοποίηση του πλαισίου και την εξάρτηση του από το εκάστοτε μηχάνημα. Έτσι λοιπόν δημιουργήθηκε ο διάδοχος του πλαισίου .NET με σκοπό τη φορητότητα και την ανεξαρτησία από την πλατφόρμα στην οποία εκτελούνται οι διεργασίες. Χρησιμοποιώντας τέτοιου είδους πλαίσια, οι προγραμματιστές είναι σε θέση να αναπτύξουν εφαρμογές για διάφορες πλατφόρμες όπως για παράδειγμα τα Windows, το Android, το iOS, καθώς επίσης και για συσκευές IoT.

Το .NET core τοποθετείται χρονικά πίσω στο 2001 όταν το Shared Source Common Language Infrastructure (SSCLI) διατέθηκε σε κοινή χρήση υπό την ονομασία Rotor. Επί της ουσίας αυτή αποτελούσε και την υλοποίηση του Common Language Infrastructure (CLI) του συγκεκριμένου πλαισίου. Το λογισμικό Rotor μπορούσε να χτιστεί σε πλατφόρμα που φιλοξενούσε FREEBSD έκδοση μεγαλύτερη της 4.7 και σε macOS X 10.2. Η αρχιτεκτονική και ο σχεδιασμός του ήταν τέτοια που επέτρεπε τη μεταφορά του CLI σε οποιαδήποτε πλατφόρμα αξιοποιώντας μόνο ένα πλατφορμικό επίπεδο αφαιρετικότητας κατάλληλο για τη συγκεκριμένη εργασία.

Η πρώτη έκδοση του πλαισίου .NET core 1.0 διατέθηκε τον Ιούνιο του 2016 μαζί με το Visual Studio 2015. Ακολούθησαν οι εκδόσεις 1.1 το Νοέμβριο του ίδιου χρόνου,
ενώ η έκδοση 2.0 δόθηκε τον Αύγουστο του επόμενου χρόνου. Ακολούθησαν οι εκδόσεις 2.1 και 2.2 μέσα στο έτος 2018, ενώ αυτή τη στιγμή (2019) το πλαίσιο βρίσκεται στην έκδοση 3.0.

3.1.1 Ανάπτυξη εφαρμογών με το .NET core και το Xamarin

Οι εφαρμογές για το .NET core μπορούν να αναπτυχθούν με το Visual Studio στο λειτουργικό σύστημα Windows, κα το αντίστοιχο λογισμικό για το λειτουργικό σύστημα macOS. Επιπλέον το Visual Studio Code και το Rider παρέχουν τη δυνατότητα για υλοποίηση στις προαναφερθείσες πλατφόρμες και στο Linux.
Το Xamarin αποτελεί το μοντέλο για την υλοποίηση εφαρμογών υπό την σκέπη του πλαισίου .NET core. Ως μέρος της διαπλατφορμικής φιλοσοφίας του πλαισίου, το Xamarin αξιοποιεί το runtime Mono. Το τελευταίο χρονολογείται πίσω στο 2001 όπου διατέθηκε για πρώτη φορά ως έργο ανοιχτού λογισμικού. Η χρησιμότητα του έγκειται στη δυνατότητα για μεταφορά μερών του πλαισίου .NET στο σύστημα Linux, iOS (MonoTouch), και στο Android (MonoDroid). Τέλος η ενοποίηση του με το πλαίσιο .NET οδήγησε στη μετονομασία του σε Xamarin.
Το Xamarin αξιοποιείται με σκοπό την παροχή ενός επιπέδου αφαιρετικότητας και προσαρμογής για το .NET και τις βιβλιοθήκες που το τελευταίο παρέχει. Μέσω της δυνατότητας για αφαιρετικότητα που προσφέρει το Mono, οι δυνατότητες του Xamarin επεκτείνονται σε κινητές πλατφόρμες όπως είναι το iOS και το Android.

3.1.2 Τα βασικά χαρακτηριστικά του .NET core 3.0

Η βασική διαφορά του πλαισίου .NET core 3.0 έκδοση αφορά κατά κύριο λόγο τις εφαρμογές desktop. Οι αλλαγές εστιάζονται κυρίως στην απόδοση της εφαρμογής και στην ταχύτητα των updates που αφορούν το runtime περιβάλλον. Επιπρόσθετα, η φύση του ανοιχτού λογισμικού πλαισίου προσφέρει εξαιρετική συμβατότητα και αυτό οφείλεται κατά κύριο λόγο στα εξελιγμένα BCL και API που προσφέρει. Έπειτα, επιτρέπει την αποσφαλμάτωση της τελευταίας έκδοσης του .NET core μόνο για μια εφαρμογή σε μια συσκευή. Έχει σχεδιαστεί με κύριο στόχο την τοπικότητα των εφαρμογών αλλά και την καθολικότητα των συσκευών στις οποίες μπορούν να εγκατασταθούν οι εφαρμογές. Επίσης, παρέχει πλήρη υποστήριξη για τα εργαλεία CLI καθώς και το αντίχτοιχο SDK. Τέλος, παρέχει βελτιώσεις αναφορικά με την πρόσβαση σε API των Windows 10, καθώς και τη δυνατότητα για φιλοξενία UMP XAML ελέγχων

σε φόρμες Windows και εφαρμογές WPF.

3.2 ASP

Η τεχνολογία Active Server Pages (ASP) εισήχθη το 1996 από τη Microsoft. Η πρώτη διαθέσιμη έκδοση δόθηκε ουσιαστικά για την απόλυτη και σύγχρονη λειτουργία με τον εξυπηρετητή Internet Information Server (IIS) της ίδιας εταιρείας. Στον πυρήνα της η τεχνολογία ASP μεταχειρίζεται είτε VBScript ή Jscript. Επί της ουσίας, η τεχνολογία ASP λειτουργεί ως ο συνδετικός κρίκος για την ενσωμάτωση της στις τεχνολογίες των ιστοσελίδων και εξαιτίας αυτού όταν γίνεται λόγος για ASP, ο μέσος χρήστης αναφέρεται στην VBScript ή στην Javascript.

Η τεχνολογία ASP είναι εξαρτώμενη από τον εξυπηρετητή. Αυτό σημαίνει ότι ο κώδικας εκτελείται στον εξυπηρετητή και έτσι ο κάθε πελάτης είναι σε θέση να προβάλλει τη σελίδα ανεξάρτητα από τον φυλλομετρητή που έχει εγκατεστημένο στον υπολογιστή του.

Η ASP βασίζεται κατά κύριο λόγο στην HTML. Κάποια από τα tags που βρίσκονται στις σελίδες ASP δεν διερμηνεύονται από τον εξυπηρετητή. Αντίθετα, στέλνονται στον φυλλομετρητή μαζί με την HTML που παράγεται από τον κώδικα ASP που είναι ενσωματωμένος. Ακολούθως ο φυλλομετρητής προβάλλει την ιστοσελίδα διερμηνεύοντας τα συγκεκριμένα HTML tags. Γενικά η ASP μπορεί να περιέχει script τα οποία εκτελούνται στον εξυπηρετητή. Αυτό έχει ως αποτέλεσμα τη δημιουργία ιστοσελίδων οι οποίες είναι απόλυτα δυναμικές.

Η συγκεκριμένη τεχνολογία δίνει τη δυνατότητα για διασύνδεση με βάσεις δεδομένων όπως η SQL Server, Oracle, Access, χρησιμοποιώντας ένα συγκεκριμένο σύνολο αντικειμένων τα οποία ονομάζονται ActiveX Data Objects (ADO).

3.2.1 Η δυναμική μεταγλώττιση

Κατά τη δημιουργία μιας σελίδας ASP .NET, δημιουργείται ο πηγαίος κώδικας μιας κλάσης τύπου .NET. Επί της ουσίας δημιουργείται ένα νέο στιγμιότυπο της κλάσης

System.Web.UI.Page ενώ τα πλήρη περιεχόμενα μιας σελίδας ASP .NET, συμπεριλαμβανομένων όλων των script και του περιεχομένου HTML μεταγλωττίζονται σε μια κλάση τύπου .NET.

Κατά τη διαδικασία της αίτησης μιας σελίδας ASP .NET, το πλαίσο .NET ελέγχει για κάποια σχετική κλάση .NET η οποία να ανταποκρίνεται στη συγκεκριμένη σελίδα. Αν μια τέτοια κλάση δεν υπάρχει, τότε το πλαίσιο .NET δημιουργεί μια τέτοια κλάση και την αποθηκεύει σε ένα συγκεκριμένο μονοπάτι στο σκληρό δίσκο. Αν η συγκεκριμένη σελίδα ζητηθεί ξανά στο μέλλον, τότε δεν πραγματοποιείται εκ νέου μεταγλώττιση της σελίδας. Αντίθετα, η ήδη μεταγλωττισμένη κλάση εκτελείται και τα αποτελέσματα επιστρέφονται στον φυλλομετρητή.

Αφού η κλάση τοποθετηθεί στο συγκεκριμένο μονοπάτι, δημιουργείται αυτόματα μια συσχέτιση μεταξύ της ιστοσελίδας και του συγκεκριμένου αρχείου της κλάσης. Αν η συγκεκριμένη ιστοσελίδα υποστεί επεξεργασία με οποιονδήποτε τρόπο, τότε αυτόματα το αρχείο της κλάσης διαγράφεται. Την επόμενη φορά που κάποιος πελάτης αιτηθεί για τη συγκεκριμένη ιστοσελίδα, τότε αυτή θα μεταγλωττιστεί αυτόματα κα θα παραχθεί ένα νέο αρχείο κλάσης .NET.

Η περιγραφή της προηγούμενης διαδικασίας αποτελεί τη λειτουργία της δυναμικής μεταγλώττισης. Αυτή ακριβώς η διαδικασία επιτρέπει στις εφαρμογές ASP .NET να υποστηρίζουν χιλιάδες διαφορετικών χρηστών. Η μεγάλη διαφορά με την παραδοσιακή τεχνολογία ASP έγκειται στο γεγονός ότι η σελίδα ASP .NET δεν χρειάζεται να υποστεί επεξεργασία και μεταγλώττιση κάθε φορά που αιτείται για αυτή κάποιος χρήστης. Αντίθετα, μια σελίδα ASP .NET μεταγλωττίζεται μόνο όταν μια εφαρμογή τροποποιείται.

3.3 AngularJs

Η αρχική μορφή των διαδικτυακών εφαρμογών τις καθιστούσε αρκετά δύστροπες εξαιτίας του γεγονότος ότι μια ιστοσελίδα θα έπρεπε να σταλεί πρώτα στον διαδικτυακό εξυπηρετητή για επεξεργασία και ακολούθως να επαναφορτωθεί ως μια νέα ιστοσελίδα. Ο συγκεκριμένος μηχανισμός της κλήσης και ανανέωσης του περιεχομένου δημιουργούσε μια περίπλοκη λειτουργικότητα, και ως εκ τούτου ο

τελικός χρήστης είχε πολύ κακή εμπειρία χρήσης.
Οι διαδικτυακές εφαρμογές αναβαθμίστηκαν με τέτοιο τρόπο ώστε να είναι κατά το δυνατόν καλύτερη η εμπειρία χρήσης. Για αυτό το λόγο εισήχθη η έννοια του XMLHttpRequest. Χάρις σε αυτό το αντικείμενο και στη λογική της ασύγχρονης επικοινωνίας, κατέστη δυνατή η επικοινωνία του πελάτη με τον εξυπηρετητή χωρίς να είναι απαραίτητη η λογική της ανανέωσης της ιστοσελίδας. Αυτή ακριβώς ήταν και η βάση επί της οποίας λειτούργησαν τα πρώτα πλαίσια της Javascript.
Από τα ποικίλα πλαίσια javascript τα οποία έκαναν την εμφάνιση τους, το JQuery θεωρήθηκε κυρίαρχο. Αυτό συνέβη γιατί το πλαίσιο jQuery προσέφερε ένα εξαιρετικό επίπεδο αφαιρετικότητας αναφορικά με τους διαφορετικούς φυλλομετρητές. Επιπλέον, έδωσε τη δυνατότητα στους προγραμματιστές να χρησιμοποιούν ένα κοινό API για τη δημιουργία εφαρμογών. Ένα από τα μεγάλα πλεονεκτήματα του συγκεκριμένου πλαισίου αφορούσε την επεξεργασία του DOM. Αντίθετα, ένα από τα μεγάλα μειονεκτήματα του πλαισίου αφορούσε την έλλειψη οργάνωσης του κώδικα με γνώμονα την ίδια την εφαρμογή, κυρίως από τη σκοπιά της διαχείρισης και της επεκτασιμότητας.
Έχοντας υπόψη όλα τα ανωτέρω, έκαναν την εμφάνιση τους ένα πλήθος πλαισίων για τη δημιουργία μεγάλης κλίμακας projects. Επίσης, μεγάλη έμφαση δόθηκε στη συντήρηση, στην επέκταση και στον έλεγχο των εφαρμογών. Επομένως τα πλαίσια που δημιουργήθηκαν έπρεπε να πληρούν τα συγκεκριμένα κριτήρια. Το πλαίσιο AngularJS της Google καθιερώθηκε ως ένα από τα πιο δημοφιλή.

Η AngularJS αποτελεί ένα ανοιχτού λογισμικού πλαίσιο διαδικτυακών εφαρμογών το οποίο προσφέρει λίγο πολύ σε ένα προγραμματιστή μια πλούσια βάση για να χτίσει τις εφαρμογές του. Ένα σύνολο πλεονεκτημάτων του συγκεκριμένου πλαισίου είναι τα ακόλουθα:
Οργάνωση κώδικα: Μια βασική αρχή για όλα τα μεγάλα project αφορά στην οργάνωση του κώδικα. Αυτή η ανάγκη εδράζεται κυρίως στη συντήρηση, τη συνεργασία, την αναγνωσιμότητα και την επεκτασιμότητα του κώδικα.
Η AngularJS προσφέρει αρχιτεκτονικά τη δυνατότητα για την αναπροσαρμογή (refactor) του κώδικα. Επιπλέον, το πλαίσιο προσφέρει τη δυνατότητα για επεξεργασία του DOM προγραμματιστικά.

Έλεγχος ποιότητα κώδικα: Ο έλεγχος κώδικα αποτελεί ένα από τα πιο κρίσιμα χαρακτηριστικά για ένα project, ακόμα και αν δεν υπάρχει άμεση σχέση με κάποιο χαρακτηριστικό (feature) ή με τον τελικό πελάτη.

Το πλαίσιο AngularJS βοηθάει σε αυτό τον τομέα προσφέροντας μια βάση για τη δημιουργία εφαρμογών οι οποίες έχουν ελεγκτή.

Διπλής διαδρομής σύνδεση: Στις πρώιμες εφαρμογές που χρησιμοποιούσαν JQuery, έπρεπε να χρησιμοποιηθεί το ίδιο το πλαίσιο για την αναζήτηση ενός συγκεκριμένου στοιχείου του DOM.

Χρησιμοποιώντας το πλαίσιο της AngularJS η ίδια εργασία είναι δυνατόν να επιτευχθεί δηλώνοντας μια ιδιότητα Javascript, και ακολούθως η σύνδεση του με την HTML. Χάρις σε αυτή τη λειτουργία δίνεται η δυνατότητα να αποφευχθεί ο κώδικας που απαιτείται για το συγχρονισμό μεταξύ της HTML και της Javascript.

Πρότυπα HTML: Η HTML είναι μια εγγενώς αδύναμη γλώσσα η οποία δημιουργήθηκε με σκοπό το σχεδιασμό layouts και της δομής μιας ιστοσελίδας, αλλά όχι για την αλληλεπίδραση με άλλες εφαρμογές. Εν ολίγοις, η HTML δεν σχεδιάστηκε με σκοπό να αποτελεί τη βάση μιας σύγχρονης διαδικτυακής εφαρμογής. Ορισμένα πλαίσια προσφέρουν τέτοιες υπηρεσίες με τις οποίες να καθίσταται πιο ομαλή η χρήση της HTML, δημιουργώντας κατάλληλες διεπαφές με τις οποίες επιτυγχάνεται η αφαιρετικότητα. Υπό αυτό το πρίσμα, το πλαίσιο της AngularJS προσφέρει τη δυνατότητα για αλληλεπίδραση με την HTML με σκοπό την επέκταση της σε οτιδήποτε χρειαστεί ώστε να διευκολυνθεί η δημιουργία μιας διαδικτυακής εφαρμογής.

Οι δομές δεδομένων της Javascript καθιστούν την ολοκλήρωση με άλλες εφαρμογές μια απλή διαδικασία: Η δυνατότητα της εργασίας με Plain Old Javascript Objects (Pojos) καθιστά την ολοκλήρωση με άλλες τεχνολογίες εξαιρετικά απλή διαδικασία. Η κατανάλωση και η διάθεση της γλώσσας Javascript χωρίς την ανάγκη για τη χρήση μεθόδων wrapping και unwrapping, αποτελούν ένα από τα ζωτικά χαρακτηριστικά ενός πλαισίου.

Επιπλέον, η δυνατότητα για ενσωμάτωση μοντέλων JSON από τον εξυπηρετητή και η άμεση κατανάλωση τους από το πλαίσιο της AngularJS, είναι ένα από τα μεγαλύτερα πλεονεκτήματα.

3.3.1 Τα δομικά στοιχεία της AngularJS

Τα βασικά δομικά στοιχεία της AngularJS είναι τα εξής:

- **Module**: Ένα module εξυπηρετεί την καλύτερη οργάνωση και διάθεση του κώδικα. Κάθε ένα module μπορεί να περιλαμβάνει άλλα εσωτερικά modules τα οποία δομούν τη συνολική λειτουργικότητα.
- **Config:** Ένα μπλοκ config σε μια εφαρμογή AngularJS, επιτρέπει την εφαρμογή της παραμετροποίησης πριν την εκκίνηση της εφαρμογής. Αυτή η δυνατότητα επιτρέπει τη δυναμική παραμετροποίηση υπηρεσιών, τον καθορισμό καταστάσεων (routes) κλπ.
- **Routes:** Τα routes επιτρέπουν τον καθορισμό συγκεκριμένων τρόπων για την περιήγηση σε συγκεκριμένες καταστάσεις μέσα σε μια εφαρμογή. Επιπρόσθετα, επιτρέπουν τον ορισμό παραμετροποίησης για κάθε route, με τέτοιο τρόπο ώστε να καθίσταται δυνατή η επιλογή του προτύπου και του controller που θα χρησιμοποιηθεί.
- **Views:** Ένα view στην AngularJS είναι επί της ουσίας ότι παράγεται μετά τη μεταγλώττιση της AngularJS, και της ενσωμάτωσης του DOM, λαμβάνοντας υπόψη επίσης και όλα τα στοιχεία της Javascript τα οποία παρασιτούν στην εφαρμογή.
- **Scope:** Το scope αποτελεί επί της ουσίας τη διασύνδεση (glue) μεταξύ ενός view και του controller. Επιπλέον, με την εισαγωγή του συντακτικού controller-as, η ανάγκη για τη ρητή εφαρμογή της εντολής scope μειώθηκε δραματικά.
- **Controller (ελεγκτής):** Ένας controller είναι υπεύθυνος για τον καθορισμό των μεθόδων και των ιδιοτήτων όπου ένα view μπορεί να ενσωματωθεί και να αλληλεπιδράσει. Θεωρείται καλή πρακτική οι controllers να δημιουργούνται με τέτοιο τρόπο ώστε να επιτυγχάνεται η κατά το δυνατόν μικρότερη προσθήκη φόρτου στην εφαρμογή (lightweight), καθώς επίσης και η επικέντρωση τους στο συγκεκριμένο view το οποίο ελέγχουν.

- **Directive:** Ένα directive αποτελεί στην ουσία μια επέκταση ενός view στην AngularJS, με το σκεπτικό ότι επιτρέπει τη δημιουργία συγκεκριμένων και επαναχρησιμοποιήσιμων στοιχείων τα οποία ενσωματώνουν τη συγκεκριμένη συμπεριφορά. Μια απλοποίηση της έννοιας ενός directive είναι ότι μπορούν να χρησιμοποιηθούν ως στοιχεία και διακοσμητικά για την ίδια την HTML. Με αυτό τον τρόπο είναι δυνατή η επέκταση ενός view, καθώς και η διάθεση τους σε περισσότερα από ένα σημεία μέσα στην εφαρμογή.
- **Service:** Με τη χρήση των services επιτυγχάνεται η κοινή διάθεση δεδομένων μέσα σε μια εφαρμογή AngularJS. Για παράδειγμα, αν πάνω από ένας controller απαιτούν τη χρήση συγκεκριμένων δεδομένων, τότε είναι δυνατή η προώθηση αυτών των δεδομένων σε ένα service και ακολούθως η διάθεση τους στους διάφορους controllers. Ουσιαστικά ένα service επεκτείνει τη λειτουργικότητα ενός controller και καθιστά εφικτή την πρόσβαση σε αυτή τη λειτουργικότητα από οποιοδήποτε σημείο.

3.3.2 Η συνολική εικόνα της AngularJS

Ένα view (DOM) στην AngularJS είναι επί της ουσίας ο κώδικας HTML που έχει παραχθεί από τον μεταγλωττιστή. Φυσικά αυτός ο κώδικας αποτελεί και τη διεπαφή με
τον τελικό χρήστη. Στη βάση του, ένας controller είναι ένα αντικείμενο της javascript το οποίο περιέχει μεθόδους και ιδιότητες. Έτσι, ένα view περιέχει τη δηλωτική μορφή της λειτουργικότητας, ενώ ο controller αποτελείται από κώδικα ο οποίος επιτρέπει την εκτέλεση. Λαμβάνοντας όλα τα προηγούμενα υπόψη, καταλαβαίνουμε πως ένα view και ένας controller είναι δύο οντότητες οι οποίες είναι πλήρως διαχωρισμένες.
Το ενωτικό στοιχείο που υπάρχει μεταξύ ενός controller και ενός view είναι το scope. Ο controller είναι υπεύθυνος για να παρέχει μεθόδους και ιδιότητες σε ένα view. Αυτό επιτυγχάνεται με την προσκόλληση τους στο scope. Πιο συγκεκριμένα, όταν μια μέθοδος ή μια ιδιότητα δηλωθεί σε ένα scope, τότε είναι διαθέσιμο να αλληλεπιδράσει με το view.

Ο κύκλος της μεταγλώττισης στην AngularJS γίνεται σε δύο φάσεις. Η πρώτη είναι αυτή της μεταγλώττισης και η δεύτερη αυτή της σύνδεσης. Όταν ο κώδικας HTML φορτωθεί ολοκληρωτικά, η AngularJS επεξεργάζεται το DOM και μεταγλωττίζει τη λίστα με όλα τα AngularJS directives. Αυτη η διαδικασία είναι γνωστή με τον όρο μεταγλώττιση. Αφού ολοκληρωθεί η διαδικασία για την HTML, η AngularJS εκκινεί τη φάση της σύνδεσης. Σε αυτή τη φάση γίνεται η σύνδεση των διαφόρων μερών της AngularJS με το κατάλληλο στιγμιότυπο scope.

3.3.3 Τα βασικά Events στην AngularJS

Όπως αναφέρθηκε νωρίτερα, το πλαίσιο της AngularJS βασίζεται σε ένα σύνολο από events τα οποία χρησιμοποιούνται με σκοπό να εκκινηθεί ένα event κάποια στιγμή, και να απαντηθεί οποιαδήποτε άλλη στιγμή μέσα στην εφαρμογή. Υπάρχουν δύο βασικοί τρόποι ώστε να εκκινηθεί ένα event. Αυτοί οι τρόποι είναι οι εντολές broadcast και emit. Η βασική διαφορά αυτών των εντολών έχει να κάνει με την κατεύθυνση στην οποία πηγαίνουν τα events. Επιπλέον, ο μηχανισμός που χρησιμοποιείται για να “αφουγκράζεται” η εφαρμογή ένα event αλλά και να απαντάται είναι η εντολή on.
Η χρήση των events προϋποθέτει την παρουσία ενός αντικειμένου scope. Αυτό επιπλέον δηλώνει ότι αν κάποιος επιθυμεί να κάνει broadcast ένα event από ένα

- service, θα πρέπει να εγχύσει ένα rootScope. Γενικότερα, οι κανόνες επιτάσσουν να αποφεύγεται η άμεση χρήση του rootScope, αλλά αντίθετα να χρησιμοποιείται ως δίαυλος μεταφοράς events. Έτσι το αντικείμενο broadcast στέλνει ένα event από τον γονέα στο παιδί και έτσι εγγυάται ότι όλα τα αντικείμενα scope θα έχουν την ευκαιρία να απαντήσουν στο event, καθώς όλα τα αντικείμενα scope τοποθετούνται ιεραρχικά κάτω από το rootScope.

3.3.4 NodeJS

Το nodejs έκανε την εμφάνιση του το 2009 από τον Ryan Dahl, ως μια πλατφόρμα ανάπτυξης λογισμικού. Το βασικό περιβάλλον γύρω από το οποίο δομήθηκε το nodejs ήταν η γλώσσα Javascript. Η βασική διαφορά του nodejs σε σχέση με τα υπόλοιπα πλαίσια ανάπτυξης εφαρμογών φυλλομετρητη έχει σχέση με το μοντέλο της

ασύγχρονης επικοινωνίας εισόδου/εξόδου.
Όταν ο δημιουργός του πλαισίου έκανε την ανάπτυξη του, δεν είχε ως γνώμονα τη γλώσσα Javascript. Αργότερα έψαχνε κάποια χαρακτηριστικά τα οποία ήταν κατάλληλα για το πλαίσιο που ήθελε να αναπτύξει. Αυτά τα χαρακτηριστικά ανέδειξαν τα ιδιώματα μιας γλώσσας προγραμματισμού η οποία θα μπορούσε να χαρακτηριστεί περισσότερο συστημική.

Τέτοια χαρακτηριστικά είναι τα ακόλουθα:

i. χειρισμός streams
ii. επεξεργασία αρχείων του συστήματος
iii. χειρισμός δυαδικών αντικειμένων
iv. διεργασίες
v. δικτυακές δυνατότητες.

Τα βασικά χαρακτηριστικά της γλώσσας nodejs είναι τα παρακάτω:

Ένα πρόγραμμα/διεργασία node εκτελείται αξιοποιώντας ένα μοναδικό νήμα, καθοδηγώντας την εκτέλεση των λειτουργιών μέσω γεγονότων. Οι διαδικτυακές εφαρμογές θεωρούνται κοστοβόρες αναφορικά με τις λειτουργίες εισόδου/εξόδου, επομένως το node θα έπρεπε να ενεργεί με τέτοιο τρόπο ώστε να εκτελεί αυτές τις λειτουργίες γρήγορα.
Η εκτέλεση ενός προγράμματος καθοδηγείται μέσω ασύγχρονων κλήσεων callback. Επιπρόσθετα, οι ενέργειες που εκτελούνται στον επεξεργαστή θα πρέπει να διαχωρίζονται σε ξεχωριστές παράλληλες διεργασίες οι οποίες εν τέλει να εκπέμπουν γεγονότα (events).
Τέλος, τα προγράμματα που θεωρούνται πολύπλοκα θα πρέπει να απαρτίζονται από απλούστερα προγράμματα.

Συμπερασματικά, η γενική αρχή του πλαισίου αφορά της διασφάλισης της συνεχούς

λειτουργίας. Υπό αυτό το πρίσμα, οι λειτουργίες δεν θα πρέπει ποτέ να μπλοκάρουν. Οι απαιτήσεις για βέλτιστη λειτουργία αναφορικά με την παράλληλη επεξεργασία και την απόδοση, απαιτεί τη μείωση των άχρηστων διεργασιών. Έτσι για παράδειγμα, αν μια διεργασία αναμένει για λειτουργίες εισόδου/εξόδου, τότε θεωρείται άχρηστη και το nodejs θα πρέπει να εκτελεί τις απαιτούμενες ενέργειες ώστε να απελευθερωθούν οι πόροι.

3.4 Microsoft Azure

Οι υπηρεσίες που παρέχονται από το Microsoft Azure αποτελούν ένα παράδειγμα του μοντέλου PaaS. Μια υπηρεσία στο σύννεφο μπορεί να σχεδιαστεί και να παρασχεθεί σε οποιοδήποτε κέντρο δεδομένων του Microsoft Azure ανά την υφήλιο. Μια υπηρεσία η οποία φιλοξενείται από το Microsoft Azure έχει όλα τα πλεονεκτήματα της επεκτασιμότητας και της ελάχιστης διαχείρισης τα οποία οφείλονται στο μοντέλο PaaS. Πιο συγκριμένα, χρησιμοποιώντας το μοντέλο PaaS, ουσιαστικά ο πελάτης εμπιστεύεται ένα μοντέλο τύπου black box. Εν ολίγοις, η είσοδος στο black box είναι ο προγραμματιστικός κώδικας και η έξοδος είναι η εφαρμογή η οποία εκτελείται σε εικονικά μηχανήματα τα οποία ανήκουν στη Microsoft.

Οι εφαρμογές-ιστοσελίδες της Azure αποτελούν μια διαχειρίσιμη υπηρεσία στο σύννεφο η οποία δίνει τη δυνατότητα για την ολοκληρωμένη παροχή διαδικτυακών υπηρεσιών καθώς και τη διάθεση τους στους πελάτες, μέσα σε ένα πολύ μικρό χρονικό διάστημα. Ένα μεγάλο πλεονέκτημα αυτής της τεχνολογίας είναι ότι δεν υπάρχει ανάγκη για τη διαχείριση των εκάστοτε εικονικών μηχανών, αλλά αυτή η ανάγκη καλύπτεται από την ίδια την πλατφόρμα.

Οι υποστηριζόμενες γλώσσες προγραμματισμού για την ανάπτυξη μιας εφαρμογής περιλαμβάνουν τη .NET, τη JAVA, την PHP, τη Node .js, καθώς και την Python. Επιπλέον, για τη δημιουργία μιας ιστοσελίδας υπάρχουν πολλές επιλογές εφαρμογών που επιτρέπουν την εύκολη και γρήγορη ανάπτυξη τους. Τέτοιες είναι για παράδειγμα, το Wordpress, το Umbraco, το Joomla, και το Drupal.

Μια βασική αρχή στο υπολογιστικό σύννεφο αφορά τον τύπο της επεκτασιμότητας. Το πρώτο είδος, η οριζόντια επεκτασιμότητα, κατά την οποία εισάγεται ένας αριθμός εξυπηρετητών, είναι άκρως αποτελεσματική σε αντίθεση με την κάθετη

επεκτασιμότητα. Η τελευταία επιτυγχάνεται με την αύξηση της ισχύος ενός εξυπηρετητή. Όπως και σε άλλες πλατφόρμες στο υπολογιστικό σύννεφο, έτσι και το Microsoft Azure εστιάζει στην οριζόντια επεκτασιμότητα. Η δυνατότητα που δίνεται για την αύξηση και τη μείωση του αριθμού των στιγμιοτύπων τα οποία τρέχουν στο υπολογιστικό σύννεφο περιγράφεται ως ελαστικότητα.

Το Microsoft Azure υποστηρίζει 2 τύπους ρόλων. Ο πρώτος αφορά τα web role τα οποία μπορούν να εξυπηρετήσουν ιστοσελίδες χρησιμοποιώντας το Internet Information Services (IIS). Επίσης, είναι σε θέση να φιλοξενήσουν πληθώρα ιστοσελίδων, αξιοποιώντας μόνο ένα endpoint και κάνοντας χρήση κεφαλίδων για το διαχωρισμό των σελίδων. Αυτό το μοντέλο βρίσκεται σε παρακμή καθώς έκανε την εμφάνιση του το μοντέλο Microsoft Azure το οποίο βασίζεται στο υπολογιστικό μοντέλο PaaS. Το στιγμιότυπο ενός web role υλοποιεί 2 διεργασίες. Η πρώτη αφορά τον κώδικα ο οποίος εκτελείται και αλληλεπιδρά με το Azure fabric και την διεργασία η οποία εκτελεί το IIS.

Στον αντίποδα, ένα worker role εξυπηρετεί μια μακρά χρονικά υπηρεσία, και επί της ουσίας αντικαθιστά τη λειτουργικότητα μιας υπηρεσίας στα Windows. Σε άλλη περίπτωση, η μοναδική διαφοροποίηση μεταξύ ενός web role και ενός worker role εντοπίζεται στον IIS. Επιπλέον, ένα worker role μπορεί να χρησιμοποιηθεί για να φιλοξενήσει διαδικτυακούς εξυπηρετητές διαφορετικούς από τον IIS. Στην πραγματικότητα τα worker roles χρησιμοποιούνται όταν κάποιος θέλει να χρησιμοποιήσει λογισμικό το οποίο δεν έχει σχεδιαστεί να λειτουργεί με το κλασικό stack της Microsoft. Για παράδειγμα, αν υπάρξει η ανάγκη να εκτελεστεί μια εφαρμογή JAVA, το worker role θα πρέπει να φορτώσει μια διεργασία του JEE Application Server και να φορτώσει τα σχετικά αρχεία τα οποία απαιτούνται για να εκτελεστεί μέσα σε αυτό.

3.4.1 Διαχείριση δεδομένων και αρχείων της Azure

Η υπηρεσία δεδομένων της Azure διαχειρίζεται ένα σύνολο αρχείων τα οποία είναι γνωστά ως blobs. Επίσης διαχειρίζεται ένα σύνολο ουρών και πινάκων. Καθοριστικής σημασίας προϋπόθεση είναι ή ασφαλής διατήρηση αυτών των αρχείων έτσι ώστε να

μην υπάρχει περίπτωση παράνομης πρόσβασης σε αυτά. Κάθε λογαριασμός που σχετίζεται με την αποθήκευση δεδομένων έχει ένα όνομα χρήστη και 2 κλειδιά πρόσβασης με τα οποία γίνεται η αυθεντικοποίηση στην υπηρεσία αποθήκευσης δεδομένων. Η συγκεκριμένη υπηρεσία παρέχει 2 διαφορετικά κλειδιά με σκοπό να αποφευχθεί η αναπαραγωγή των κλειδιών. Επιπλέον, η υπηρεσία αποθήκευσης δεδομένων υποστηρίζει τη συνάρτηση HMAC (Hash-based message authentication). Η βιβλιοθήκη της Azure για την αποθήκευση των δεδομένων, παρέχει ένα σύνολο κλάσεων οι οποίες υποστηρίζουν διαφορετικούς τρόπους με τους οποίους μπορεί να γίνει η παραγωγή του HMAC και έτσι προσφέρουν μια διεπαφή η οποία αποκρύπτει την πολυπλοκότητα παραγωγής του συγκεκριμένου κρυπτογραφήματος.

Έχοντας ως γνώμονα τα δεδομένα, μια βασική υπηρεσία της Azure αφορά και τα διαγνωστικά τεστ τα οποία πρέπει να γίνονται σε μια υπηρεσία η οποία τρέχει στο σύννεφο. Μια τέτοια υπηρεσία της Azure μπορεί να αποτελείται από πολλά στιγμιότυπα και πολλαπλούς ρόλους. Όλα τα στιγμιότυπα τρέχουν σε ένα απομακρυσμένο κέντρο δεδομένων της Azure, το οποίο λειτουργεί 24/7. Τα διαγνωστικά δεδομένα μπορούν να χρησιμοποιηθούν ώστε να γίνουν γνωστά προβλήματα με μια υπηρεσία στο σύννεφο. Η ικανότητα για παρατήρηση των δεδομένων από πολλές διαφορετικές πηγές και μέσω διαφορετικών στιγμιοτύπων, διευκολύνει τη διαδικασία εύρεσης ενός προβλήματος. Η διαδικασία για την παραμετροποίηση των διαγνωστικών τεστ στο Azure εντοπίζεται στο επίπεδο των ρόλων, αλλά η παραμετροποίηση των διαγνωστικών γίνεται σε επίπεδο στιγμιότυπου. Για κάθε στιγμιότυπο, ένα ξεχωριστό αρχείο XML σε ένα συγκεκριμένο container που ονομάζεται wad-control-container και ο οποίος βρίσκεται στον λογαριασμό της υπηρεσίας αποθήκευσης που έχει παραμετροποιηθεί για τη συγκεκριμένη υπηρεσία.

Η υπηρεσία υποστηρίζει τα ακόλουθα δεδομένα:

Αρχεία λογιστικών καταγραφών (logs) επιπέδου εφαρμογής (application): Στα συγκεκριμένα δεδομένα περιλαμβάνουν πληροφορία η οποία καταγράφεται σε κάποιο listener που έχει δημιουργηθεί για την καταγραφή δεδομένων εφαρμογής.
Αρχεία λογιστικών καταγραφών γεγονότων (events): Αυτά τα δεδομένα

περιλαμβάνουν πληροφορία από οποιοδήποτε παραμετροποιήσιμο αρχείο λογιστικών καταγραφών που σχετίζεται με γεγονότα στα Windows.

Αρχεία λογιστικών καταγραφών υποδομών: Αυτά τα δεδομένα περιλαμβάνουν πληροφορίες που σχετίζονται με την ίδια τη διαγνωστική διεργασία.

Μετρήσεις απόδοση: Αυτά τα δεδομένα περιλαμβάνουν πληροφορία που σχετίζεται με οποιαδήποτε παραμετροποιήσιμη μέτρηση απόδοσης

Κεφάλαιο 4ο

Ηλεκτρονικός Ιατρικός Φάκελος

Ο ηλεκτρονικός ιατρικός φάκελος (EMR) είναι μια ψηφιακή έκδοση όλων των πληροφοριών που βρίσκονται συνήθως σε έναν φάκελο νοσοκομείου, όπως αυτές είναι: ιατρικό ιστορικό, διαγνώσεις, φάρμακα, ημερομηνίες ανοσοποίησης, αλλεργίες, εργαστηριακά αποτελέσματα και σημειώσεις γιατρού.

Τα EMR κοινώς, είναι online ιατρικά αρχεία των τυποποιημένων ιατρικών και κλινικών δεδομένων από το γραφείο ενός παρόχου υγείας, τα οποία χρησιμοποιούνται κυρίως από τους ίδιους για τη διάγνωση και τη θεραπεία των ασθενών.

Τα EMR είναι κάτι παραπάνω από απλώς αντικαταστάτης των χαρτογραφημάτων. Επιτρέπουν αποτελεσματικά την επικοινωνία και τον μεταξύ των μελών μιας ομάδας υγείας με σκοπό τη βέλτιστη φροντίδα των ασθενών ανεξαρτήτου χρονικής στιγμής.

4.1 Ορισμός Ηλεκτρονικού φακέλου υγείας

Το λογισμικό του ηλεκτρονικού φακέλου υγείας έχει δύο βασικά χαρακτηριστικά που το διαφοροποιούν πλήρως από τα υπόλοιπα πληροφοριακά συστήματα. Το πρώτο χαρακτηριστικό αφορά τους βασικούς του χρήστες οι οποίοι είναι οι πάροχοι των υπηρεσιών υγείας. Το δεύτερο χαρακτηριστικό αφορά το γεγονός ότι χρησιμοποιούνται με σκοπό την αποθήκευση και την παρουσίαση των δεδομένων που αφορούν την κλινική κατάσταση ενός ασθενούς. Πιο συγκεκριμένα, αποτυπώνονται τα συμπτώματα, τα δεδομένα των εξετάσεων, καθώς και το πλάνο των θεραπειών ενός ασθενούς. Ο ηλεκτρονικός φάκελος του ασθενούς αποτελεί ένα εργαλείο το οποίο βοηθά τον κλινικό επιστήμονα να διαχειριστεί το σύνολο των ενεργειών που απαιτούνται να γίνουν καθ΄ όλη τη διάρκεια της παροχής των υπηρεσιών υγείας. Οι μέθοδοι που αξιοποιούνται με σκοπό τη συλλογή των δεδομένων, βοηθούν στη διασφάλιση της ορθής καταγραφής των δεδομένων των ασθενών, τα οποία αφορούν

κάποια συγκεκριμένη ιατρική διάγνωση που έχει γίνει.

4.2 Ιστορική Αναδρομή

Οι πρώτες αναφορές για τη χρήση ηλεκτρονικών υπολογιστών με σκοπό την υποστήριξη δεδομένων υγείας, έκαναν την εμφάνιση τους για πρώτη φορά στο τέλος της δεκαετίας του 1950. Οι πρώτες προσπάθειες για να δημιουργηθεί ένα σύστημα ηλεκτρονικού φακέλου ασθενούς όπως είναι γνωστό σήμερα, ξεκίνησε στις αρχές του 1960. Το σύστημα είχε την ονομασία COSTAR και αναπτύχθηκε από τον Barnett στο εργαστήριο του τμήματος επιστήμης υπολογιστών του γενικού νοσοκομείου της Μασαχουσέτης. Την ίδια περίοδο έγιναν προσπάθειες από το Πανεπιστήμιο του Duke, καθώς και από το Ιατρικό κέντρο του Πανεπιστημίου της Ιντιάνα, οι οποίες οδήγησαν στη δημιουργία εύρωστων συστημάτων ηλεκτρονικών φακέλων ασθενών τα οποία μπορούν να υποστηρίξουν χιλιάδες ασθενείς. Επιπλέον, οι εμπορικές εφαρμογές των ηλεκτρονικών φακέλων ασθενών είναι πλέον διαθέσιμες από περισσότερες των 200 εταιρειών. Παρόλα αυτά, αν και υπήρξε αυτή η ανάπτυξη στον τομέα, η συγκεκριμένη τεχνολογία δεν έγινε ευρέως γνωστή. Εκτιμάται ότι μόνο το 3 - 5 % των ιατρών έχουν πρόσβαση ή χρησιμοποιούν τους ηλεκτρονικούς φακέλους ασθενών σε τακτική βάση. Δοθείσης της τεσσαρακονταετούς ανάπτυξης αυτών των συστημάτων είναι ιδιαίτερα παράδοξο το γεγονός ότι οι ηλεκτρονικοί φάκελοι ασθενών δεν αποτελούν ένα σύνηθες κλινικό εργαλείο. Η απάντηση σε αυτό το ερώτημα τεκμηριώνεται εξαιτίας της πολυπλοκότητας των θεμάτων που αφορούν αυτά τα συστήματα. Με το πέρασμα των ετών οι ηλεκτρονικοί υπολογιστές βελτιώθηκαν για να υποστηρίζουν ένα σύνολο από ενέργειες σχετικές με δεδομένα που αφορούν τον τομέα της υγείας. Μέχρι πρόσφατα τα ιδρύματα υγείας θεωρούνταν πρωτοπόρα στην ανάπτυξη των πληροφοριακών συστημάτων τα οποία σχετίζονται με τα δεδομένα υγείας.

4.3 Εφαρμογές ηλεκτρονικού φακέλου υγείας

Όπως προαναφέρθηκε, ένα σύστημα ηλεκτρονικού φακέλου υγείας μπορεί να μειώσει

σημαντικά τον φόρτο εργασίας και τα ιατρικά λάθη, ενώ την ίδια ώρα να οφελήσει στη μείωση του συνολικού κόστους ενός συστήματος υγείας.
Υπάρχουν ένα σύνολο συστημάτων ηλεκτρονικού φακέλου υγείας τα οποία διατίθενται ως έργα ανοιχτού λογισμικού.

Τέτοιες περιπτώσεις είναι οι ακόλουθες:

- **OPEN MRS:** Το συγκεκριμένο σύστημα είναι αρχιτεκτονικά δομημένο ως διαδικτυακή εφαρμογή, αλλά δεν απαιτεί την πρόσβαση στο διαδίκτυο. Οι απαιτήσεις που έχει αναφορικά με το υλικό είναι ελάχιστες. Αφορά τον τομέα της πρωτογενούς υγείας καθώς και πιο συγκεκριμένα στον ιό HIV, και διατίθεται στα Αγγλικά και τα Ισπανικά. Επιπλέον, προσφέρει την αυτόματη δημιουργία του κωδικού του ασθενούς, την αναζήτηση και την ανάκτηση πληροφοριών θέτοντας ένα σύνολο διαφορετικών κριτηρίων, καθώς και την εξαγωγή αναφοράς από τον χρήστη. Το συγκεκριμένο λογισμικό μπορεί να εγκατασταθεί σε λειτουργικό σύστημα Linux και χρησιμοποιεί τους
- εξυπηρετητές Apache και GlassFish για την εξυπηρέτηση των σελίδων που προσφέρει. Επίσης το σύστημα βάσεων δεδομένων που φιλοξενεί είναι η MySQL. Οι απαιτήσεις λογισμικού που έχει το συγκεκριμένο σύστημα εντοπίζονται στο πακέτο JDK της JAVA που θα πρέπει να είνα μεγαλύτερο της έκδοσης 1.6, καθώς και η έκδοση της PHP που θα πρέπει να είναι μεγαλύτερη από την 5.3.
- **Dream eSant EDIGIO:** Το συγκεκριμένο σύστημα λειτουργεί σε πλατφόρμα Windows και χρησιμοποιεί MS SQL Server ως σύστημα βάσεων δεδομένων, καθώς και MS ACCESS. Τα συγκεκριμένα λογισμικά έχουν απαιτήσεις αναφορικά με τη συντήρηση και τα κόστη τους. Έτσι, θεωρείται αναγκαία τόσο η εξέταση τους από κατάλληλο λογισμικό για την ανίχνευση ιών, αλλά επίσης και η ανανέωση των αδειών χρήσης που απαιτούν. Οι απαιτήσεις που έχει το λογισμικό αναφορικά με το υλικό είναι ελάχιστες. Το συγκεκριμένο λογισμικό διατίθεται δωρεάν, αλλά η άδεια χρήσης του δεν επιτρέπει την πρόσβαση στον κώδικα. Αυτό πρακτικά σημαίνει ότι περιορίζεται η δυνατότητα για την παροχή

εξειδικευμένων λύσεων όπου απαιτούνται. Επιπλέον, η εφαρμογή ακολουθεί το μοντέλο της δικτυακής αρχιτεκτονικής πελάτη εξυπηρετητή. Πιο συγκεκριμένα, οι χρήστες μπορούν να συνδέονται μέσω του τοπικού δικτύου, αλλά δεν είναι σε θέση να συνδεθούν μέσω διαδικτύου. Το συγκεκριμένο λογισμικό σχεδιάστηκε με γνώμονα τον ιό HIV και διατίθεται στις ακόλουθες γλώσσες: Αγγλικά, Γαλλικά, Πορτογαλικά, και Ιταλικά. Η συγκεκριμένη λύση λογισμικού αξιοποιεί την τεχνολογία των φορμών και διαθέτει ένα πάρα πολύ αναλυτικό σύστημα για την εγγραφή των ασθενών.

- **G.H.I.S.:** Το εν λόγω λογισμικό σχεδιάστηκε τόσο για τον πρωτογενή τομέα υγείας όσο και για τον ιό HIV. Το συγκεκριμένο σύστημα κάνει επίσης χρήση της δικτυακής αρχιτεκτονικής πελάτη εξυπηρετητή και έχει αναπτυχθεί με γνώμονα το λειτουργικό σύστημα WINDOWS. Επίσης, το σύστημα βάσεων δεδομένων που χρησιμοποιεί είναι αυτό της MS SQL SERVER. Πρέπει να αναφερθεί ότι όπως και στις προηγούμενες περιπτώσεις, οι απαιτήσεις του υλικού είναι ελάχιστες για τη λειτουργία του συγκεκριμένου λογισμικού. Η απλότητα της αρχιτεκτονικής και οι ελάχιστες απαιτήσεις σε υλικό είναι 2 παράγοντες που απογειώνουν την απόδοση του συγκεκριμένου συστήματος. Παρόλαυτα, το σύστημα υστερεί στην υποστήριξη απομακρυσμένων χρηστών. Το σύστημα διατίθεται στην Αγγλική γλώσσα και όπως το σύστημα Dream, η χρήση ιδιόκτητης πλατφόρμας όσο αυξάνεται ο αριθμός των χρηστών, θεωρείται οικονομικά ασύμφορη. Οι ιατρικές συνταγές μπορούν να εκτυπωθούν ή να σταλούν ηλεκτρονικά. Αυτό το χαρακτηριστικό επιτρέπει την καταγραφή και την οργάνωση των εμπορευμάτων. Το σύστημα μπορεί να παράξει αναφορές σχετικά με δημογραφικά στοιχεία, κωδικούς ICD, ιατρικές συνταγές κλπ.
- **ISANTE:** Το συγκεκριμένο σύστημα είναι βασισμένο και πάλι σε δικτυακή αρχιτεκτονική πελάτη εξυπηρετητή, αλλά δεν απαιτεί την πρόσβαση στο διαδίκτυο. Επίσης όπως και σε όλες τις προηγούμενες περιπτώσεις η πρόσβαση στο διαδίκτυο δεν είναι αναγκαία. Η συγκεκριμένη λύση λογισμικού μπορεί να τρέξει σε πλατφόρμες Linux με τη χρήση εξυπηρετητή Apache και σύστημα βάσεων δεδομένων My SQL, όσο και σε περιβάλλοντα WINDOWS.

Το σύστημα δημιουργήθηκε έχοντας ως γνώμονα τον ιό HIV και διατίθεται στα Αγγλικά και τα Γαλλικά. Όπως και τα προηγούμενα συστήματα έχει δυνατότητα για την εγγραφή των ασθενών. Αξιοποιεί την τεχνολογία των φορμών και είναι εύκολο να δημιουργηθούν κατά την πρώτη επίσκεψη, το ιατρικό ιστορικό ενός ασθενούς, καθώς και το ιστορικό της οικογενείας του. Παρόλαυτά δεν είναι εύκολο να γίνει επεξεργασία αυτών σε επόμενο στάδιο. Τα προβλήματα που παρουσιάζονται εμφανίζονται με συγκεκριμένο κωδικό ICD, αλλά σε μια πολύ μικρή λίστα. Το συγκεκριμένο λογισμικό έχει δημιουργηθεί για τη διαλειτουργικότητα με ένα φαρμακείο, αλλά δεν είναι σε θέση για την καταγραφή αλλεργιών και αλληλεπιδράσεων. Τέλος, το λογισμικό μπορεί να παράξει αναφορές βάσει δημογραφικών, κωδικών ICD και ιατρικών συνταγών.

- **WORLD VISTA:** Το συγκεκριμένο λογισμικό έχει αναπτυχθεί παρέχοντας τη δυνατότητα για πρόσβαση τόσο μέσω δικτυακής εφαρμογής, όσο και ακολουθώντας την δικτυακή αρχιτεκτονική πελάτη εξυπηρετητή. Ένα μεγάλο πλεονέκτημα της συγκεκριμένης εφαρμογής αφορά την οργάνωση της κοινότητας η οποία το υποστηρίζει. Παρόλα αυτά ένα μεγάλο μειονέκτημα
- αφορά τον προγραμματιστικό κώδικα ο οποίος δεν είναι εύκολο να τροποποιηθεί. Η συγκεκριμένη λύση χρησιμοποιείται στις Ηνωμένες Πολιτείες της Αμερικής, κυρίως σε περιβάλλοντα κλινικών και νοσοκομειακών ιδρυμάτων. Το σύστημα έχει δημιουργηθεί με γνώμονα την ιατρική φροντίδα, αλλά πρότυπα για εξειδικευμένη φροντίδα μπορούν να δημιουργηθούν από τον τελικό χρήστη. Η γλώσσα στην οποία διατίθεται το λογισμικό είναι η Αγγλική. Το παλιό ιστορικό του ασθενούς και της οικογενείας του, καθώς και παράγοντες κινδύνου, μπορούν να εισαχθούν στο σύστημα με τη μορφή μεταβλητών που κατέχουν συγκεκριμένους κωδικούς. Το μειονέκτημα των τελευταίων αφορά τη δυσκολία στην επεξεργασία τους μετά την πρώτη επίσκεψη. Τα προβλήματα παρουσιάζονται με τη μορφή λιστών. Παρέχεται η δυνατότητα για προβολή μικρών και αναλυτικών λιστών. Το μεγάλο πλεονέκτημα του συγκριμένου λογισμικού σε σχέση με τα προηγούμενα αφορά την ενσωματωμένη λίστα φαρμάκων (αυτά αφορούν φάρμακα που κυκλοφορούν στις Ηνωμένες Πολιτείες της Αμερικής), και επιτρέπει την ταυτοποίηση ιατρικών αλλεργιών και

αλληλεπιδράσεων. Τέλος είναι σε θέση να δημιουργεί φόρμες βάσει δημογραφικών στοιχείων, συγκεκριμένων προβλημάτων και φαρμακευτικής αγωγής.

- **OSCAR:** Το OSCAR αναπτύχθηκε στον Καναδά με σκοπό την κύρια φροντίδα υγείας. Όπως και τα προηγούμενα λογισμικά, το συγκεκριμένο σύστημα δεν έχει ιδιαίτερες απαιτήσεις σε υλικό και χρησιμοποιεί δικτυακή αρχιτεκτονική. Οι πλατφόρμες στις οποίες μπορεί να τρέξει το συγκεκριμένο λογισμικό καθώς και τα διάφορα εργαλεία τα οποία απαιτούνται για τη λειτουργία του είναι όλα ανοιχτού κώδικα. Επιπλέον, το συγκεκριμένο λογισμικό έχει μια ισχυρή κοινότητα υποστήριξης. Όπως και τα άλλα εργαλεία το συγκεκριμένο έχει εγγραφή ασθενών και χρησιμοποιεί πρότυπα τα οποία βασίζονται σε φόρμες. Σε αντίθεση με τα άλλα εργαλεία επιτρέπει την ενημέρωση των δεδομένων αναφορικά με παλιά ιατρικά δεδομένα τόσο του ίδιου του ασθενούς όσο και της οικογενείας του. Τα προβλήματα παρουσιάζονται με κωδικούς ICD τόσο σε σύντομες, όσο και σε αναλυτικές λίστες. Το προϊόν έχει ενσωματωμένη μια λίστα φαρμάκων τα οποία αντιστοιχούν μόνο σε αυτά που κυκλοφορούν στον Καναδά. Αυτό επιτρέπει τον έλεγχο αλλεργιών καθώς και άλλων αλληλεπιδράσεων. Τέλος, μπορεί να παράξει αναφορές βάσει δημογραφικών στοιχείων και κωδικών ICD.

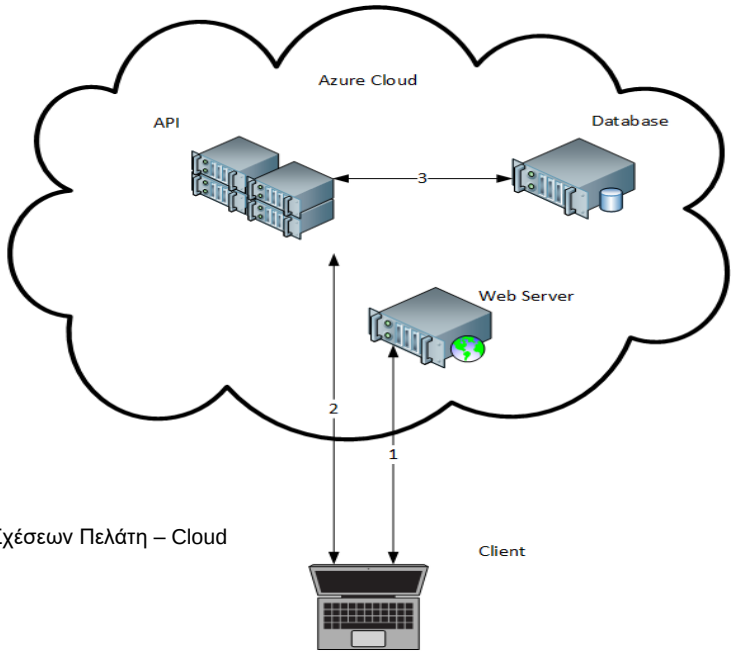
Κεφάλαιο 5ο

Πρακτική Εφαρμογή

Όπως ήδη έχει αναφερθεί στα πλαίσια της παρούσης πτυχιακής εργασίας έγινε μια προσπάθεια υλοποίησης ενός συστήματος οργάνωσης και διαχείρισης ιατρικών ηλεκτρονικών φακέλων, προσφέροντας συγχρόνως την δυνατότητα στους άμεσα ενδιαφερόμενους, να ορίζουν συναντήσεις επιλέγοντας βάση κάποιων κριτηρίων, όπως η ημέρα και η τοποθεσία.

Το σύστημα αυτό, όπως έχει κατασκευαστεί, απευθύνετε σε δυο ομάδες χρηστών, τους Ιατρούς και τους Ασθενείς. Κάθε μία από τις ομάδες αυτές, έχει διάφορες δυνατότητες που παρουσιάζονται στην ενότητα παρακάτω και αναλύονται, με σκοπό, να γίνουν κατανοητές αλλά συγχρόνως να τονιστούν τα πλεονεκτήματα που μπορεί να αποκτήσει η κάθε ομάδα από τις προαναφερθείσες, από την χρήση του συστήματος αυτού.

Τέλος, για την υλοποίηση της εφαρμογής αυτής χρησιμοποιήθηκε ASP και .net Core για την δημιουργία RESTful JSON API, με σκοπό ασφαλή μετάφραση και επικοινωνία του τελικού πελάτη / φυλλομετρητή (client / browser) με την βάση δεδομένων MariaDB στην οποία αποθηκεύονται όλα τα δεδομένα και αρχεία. NodeJS→ Angular 7 για την δημιουργία του περιβάλλοντος φυλλομετρητή. Microsoft Azure για την φιλοξενία όλων τον παραπάνω



Σχήμα 1 Διάγραμμα Σχέσεων Πελάτη – Cloud

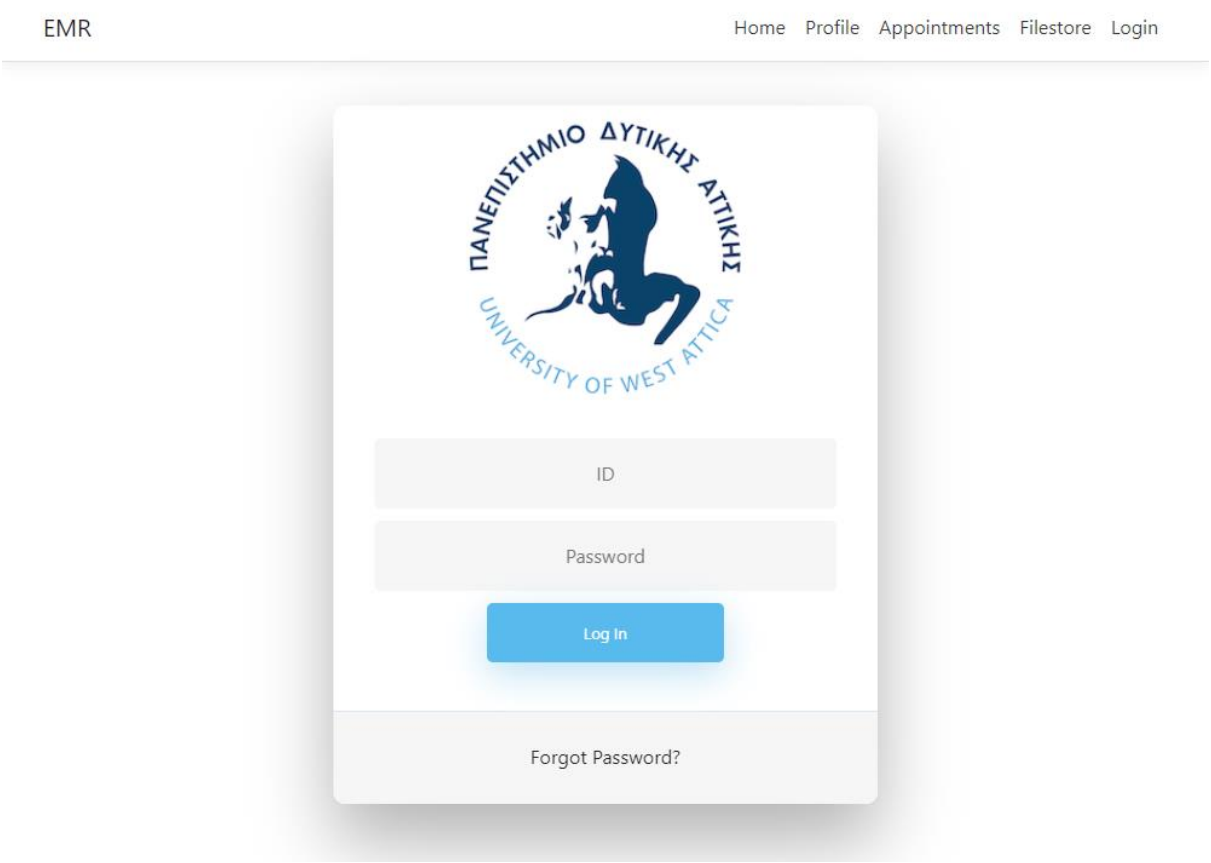
5.1 Ιατροί και σύστημα διαχείρισης ιατρικού ηλεκτρονικού φακέλου ασθενών & ραντεβού

Οι επαγγελματίες ιατροί εκμεταλλευόμενοι τις δυνατότητες που τους προσφέρει η εφαρμογή μπορούν να αποκτήσουν εύκολα και γρήγορα πρόσβαση σε ιατρικούς φακέλους των ασθενών τους αλλά και να διαχειριστούν τα ραντεβού που ενδεχομένως έχουν, εύκολα, χωρίς να είναι υποχρεωμένοι να βρίσκονται στο ιατρείο τους. Το μόνο που χρειάζονται είναι ένας υπολογιστής ή κινητό με σύνδεση στο ίντερνετ.

Παρακάτω παρουσιάζονται συνοπτικά τα προαναφερθέντα με την μορφή εικόνων.

Πιο συγκεκριμένα οι ιατροί έχουν τις εξής δυνατότητες:

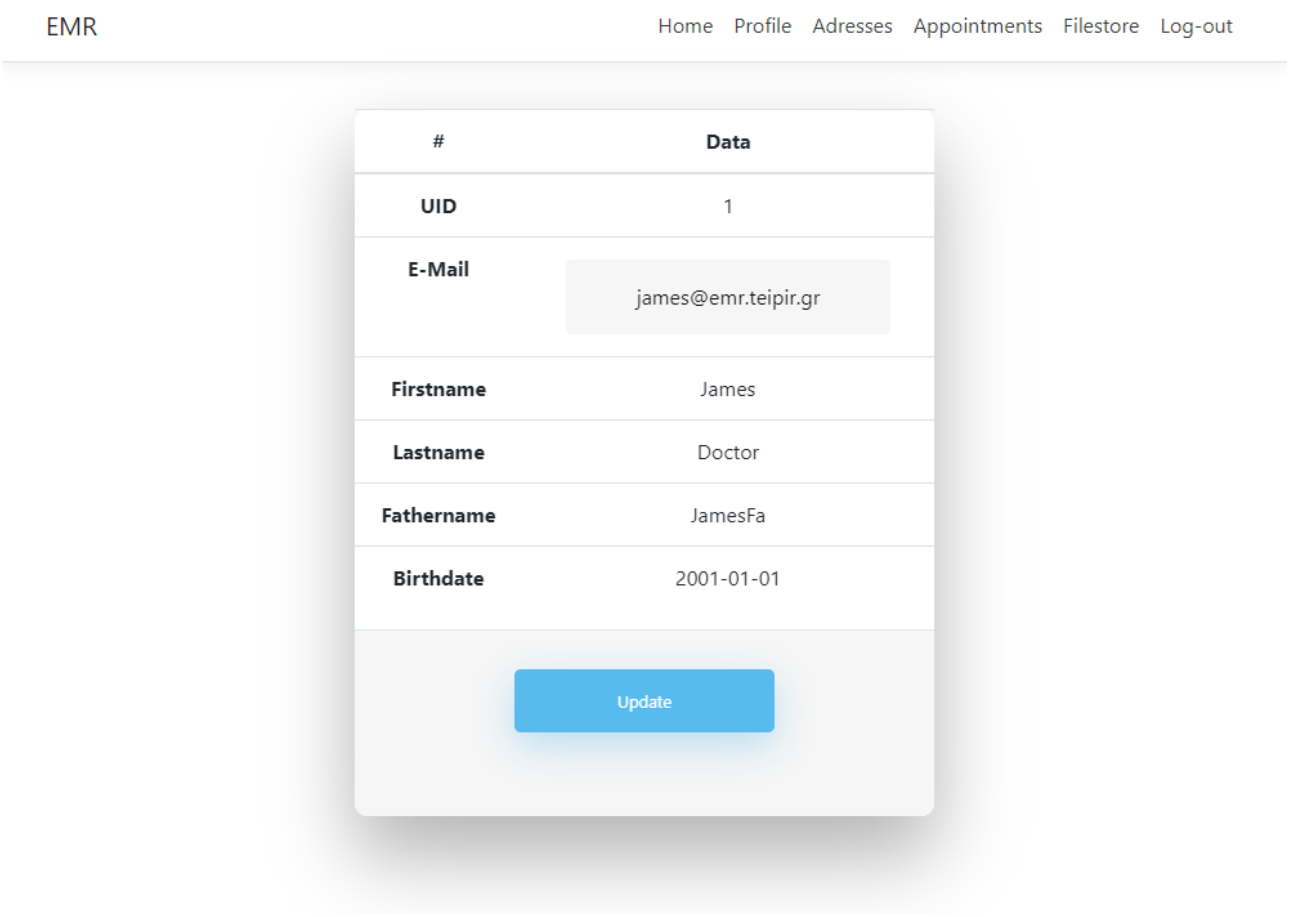
- Σύνδεσης (Login)



Εικόνα 1: Πλαίσιο εισαγωγής διαπιστευτηρίων ιατρών

Η σύνδεση επιτυγχάνεται αφού ο ιατρός εισάγει το όνομα χρήστη – id και τον κωδικό κατά την διαδικασία που έχει ορίσει κατά την ενεργοποίηση της υπηρεσίας, στο συγκεκριμένο πλαίσιο εισαγωγής διαπιστευτηρίων της εφαρμογής.

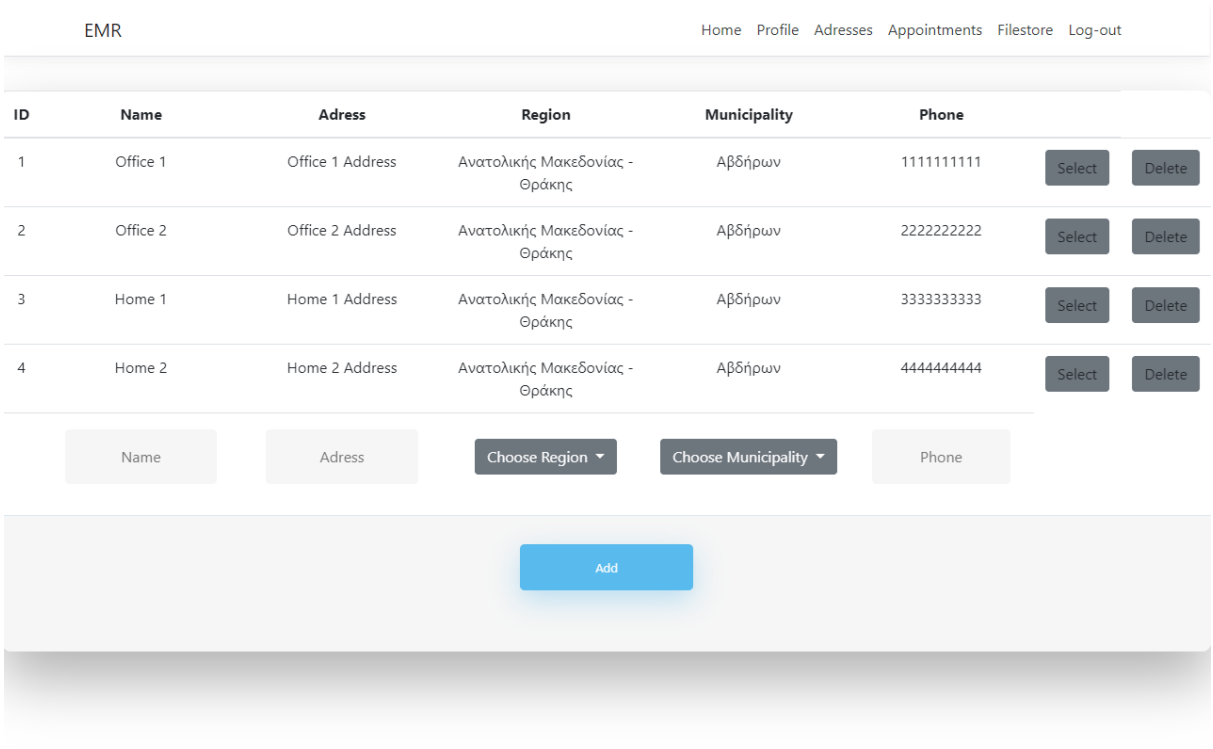
- Προβολής των στοιχείων τους και αλλαγή της διεύθυνσης ηλεκτρονικού ταχυδρομείου τους (Profile)



Εικόνα 2: Πλαίσιο ενημέρωσης ηλεκτρονικής διεύθυνσης ιατρών

Στο συγκεκριμένο πλαίσιο ο ενδιαφερόμενος ιατρός μπορεί να αλλάξει το email έχει ορίσει κατά την ενεργοποίηση της υπηρεσίας, ορίζοντας χωρίς κανέναν περιορισμό την διεύθυνση ηλεκτρονικού ταχυδρομείου της αρεσκείας του.

- Εισαγωγή, διαγραφή και επεξεργασία των διευθύνσεων τους – εξεταστικά κέντρα (Addresses)

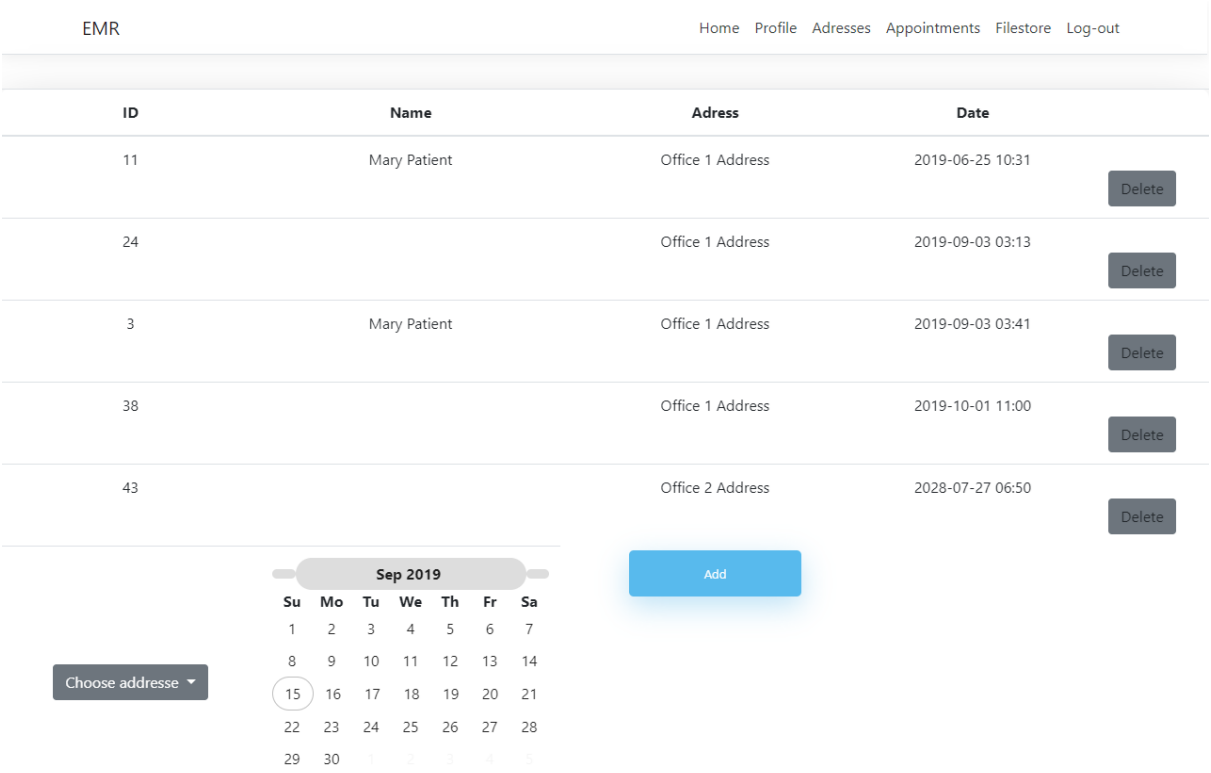


Εικόνα 3: Πλαίσιο εισαγωγής και επεξεργασίας διευθύνσεων εξεταστικών κέντρων

Δίνεται η δυνατότητα εισαγωγής νέων και διαγραφής ή επεξεργασίας ήδη υπαρχόντων εξεταστικών κέντρων – ιατρείων με αποτέλεσμα να είναι πάντα ενημερωμένη η λίστα με τα διαθέσιμα σημεία εξέτασης.

Αυτό έχει το πλεονέκτημα οι ιατροί με περισσότερα από ένα ιατρεία να μπορούν να δέχονται ασθενείς και στα δύο, συγκεκριμένες μέρες και ώρες που έχουν διαθέσιμο χρόνο για επισκέψεις. Από την άλλη, οι ασθενείς πάντα γνωρίζουν σε ποιο σημείο μπορούν να βρούν διαθέσιμο τον ιατρό που τους επιβλέπει αποφεύγοντας τις ημέρες που δεν είναι διαθέσιμος.

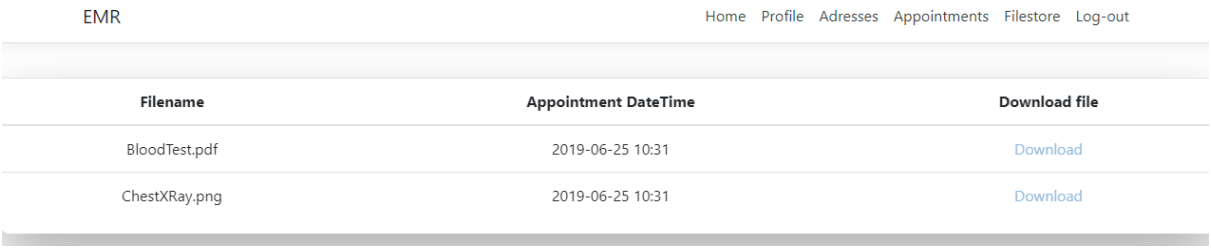
- Εισαγωγή, διαγραφή και επεξεργασία των διαθέσιμων ραντεβού



Εικόνα 4: Πλαίσιο επεξεργασίας ραντεβού – επισκέψεων

Ενημερώνεται εύκολα και γρήγορα των ημερολόγιο με τα διαθέσιμα ραντεβού με αποτέλεσμα να γνωρίζει πάντα και ο ιατρός αλλά και ο ασθενής πότε έχει διαθέσιμο χρόνο για επίσκεψη.

- Κατέβασμα των διαθέσιμων αρχείων ηλεκτρονικού ιατρικού φακέλου των ασθενών



Εικόνα 5: Πλαίσιο ελέγχου ιατρικών φακέλων ασθενών

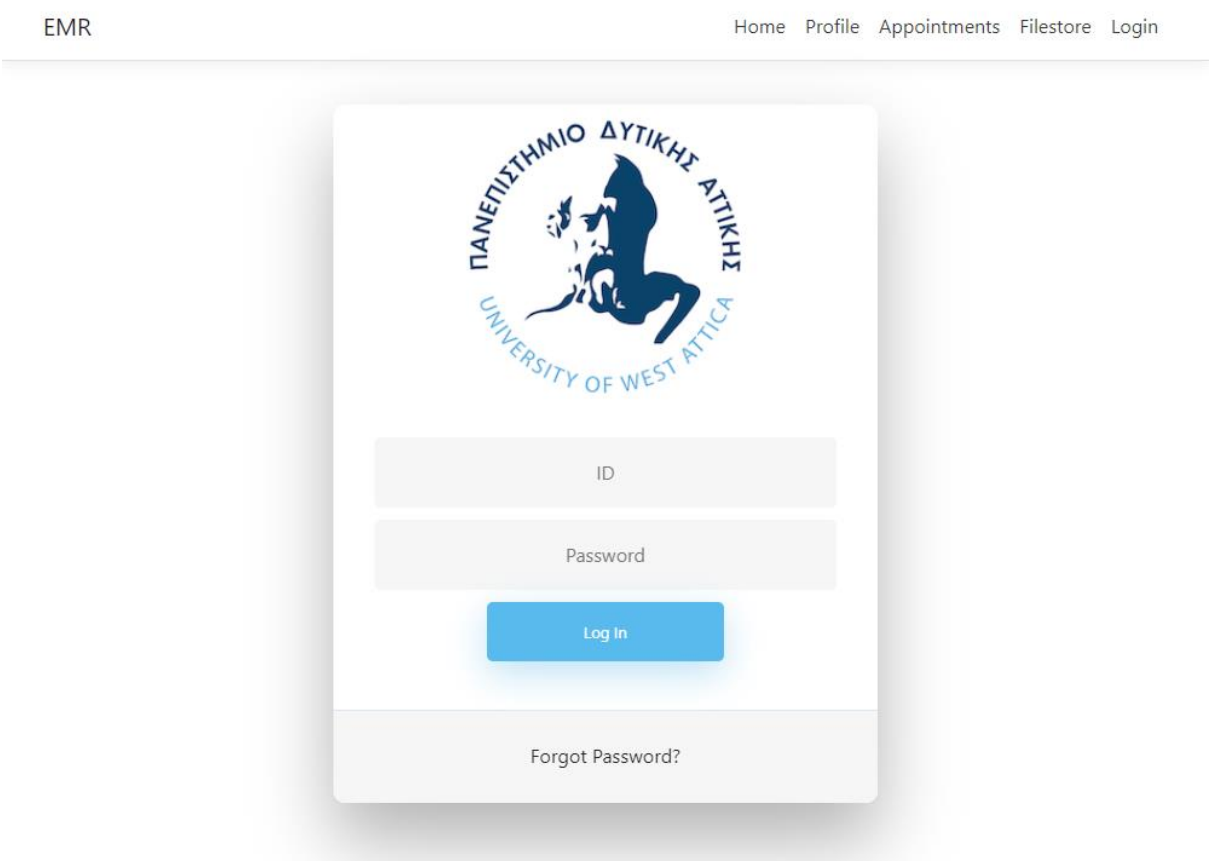
Δίνεται ανά πάσα ώρα και στιγμή η δυνατότητα ελέγχου του φακέλου των ασθενών χωρίς να χρειάζεται η μεταφορά του ιατρού στο χώρο οπού φυλάσσεται το αρχείο.

5.2 Ασθενείς & σύστημα διαχείρισης ιατρικού ηλεκτρονικού φακέλου & ραντεβού

Η δεύτερη ομάδα στην οποία αναφέρεται η συγκεκριμένη εφαρμογή είναι οι ασθενείς, οι οποίοι αποκομίζουν αρκετά πλεονεκτήματα σε σχέση με τον παραδοσιακό τρόπο επικοινωνίας τους με τους εκάστοτε θεράποντες ιατρούς.
Πιο συγκεκριμένα με την χρήση της εφαρμογής μπορεί ο κάθε ασθενής να κανονίζει μία ή περισσότερες επισκέψεις στον ιατρό με γνώμονα την τοποθεσία του μέσω του ελέγχου και της αναζήτησης διαθέσιμων ραντεβού αλλά και να ενημερώνει τον ιατρικό το φάκελο, ανά πάσα ώρα και στιγμή, με τις πιο πρόσφατες εξετάσεις.

Πιο συγκεκριμένα οι ασθενείς έχουν τις εξής δυνατότητες:

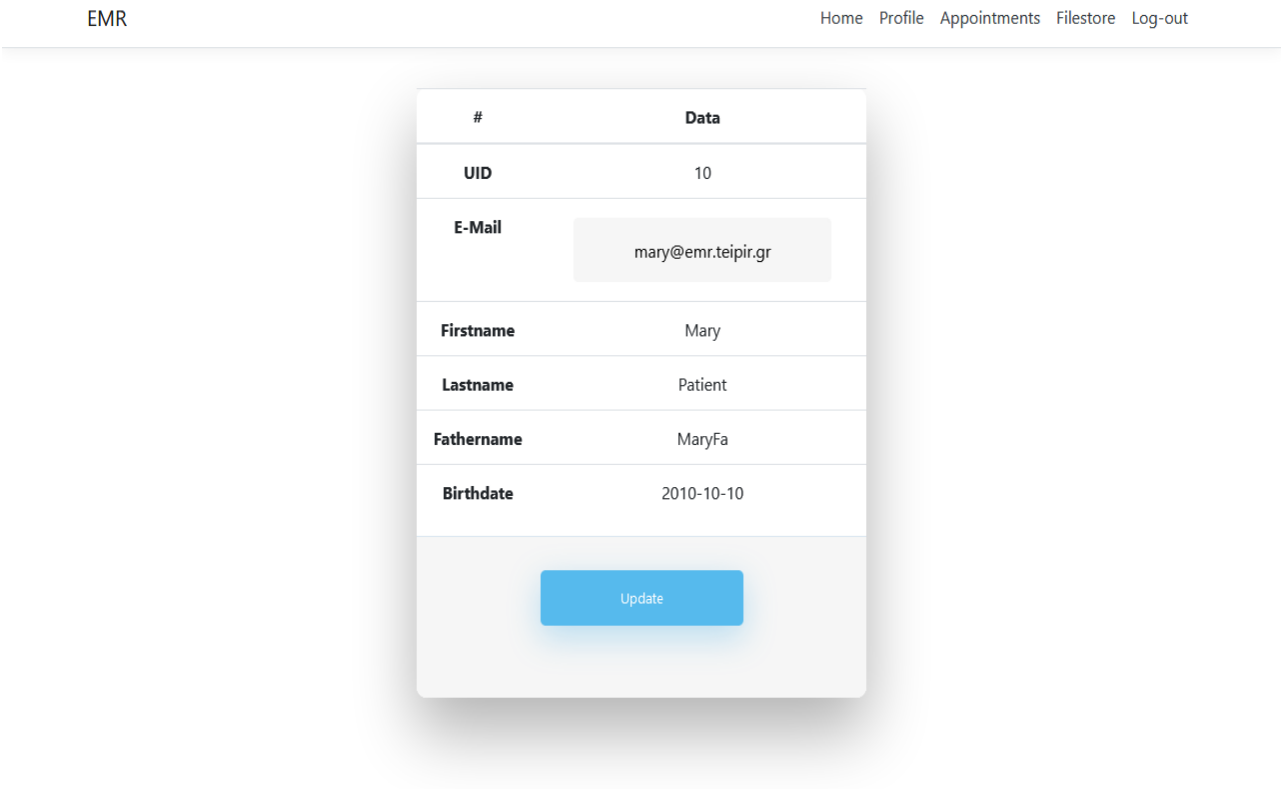
- Να συνδέονται στον λογαριασμό τους.



Εικόνα 6: Πλαίσιο εισαγωγής διαπιστευτηρίων ασθενών

Η σύνδεση επιτυγχάνεται αφού ο ασθενής εισάγει το όνομα χρήστη – id και τον κωδικό κατά την διαδικασία που έχει ορίσει κατά την ενεργοποίηση της υπηρεσίας, στο συγκεκριμένο πλαίσιο εισαγωγής διαπιστευτηρίων της εφαρμογής.

- Προβολής των στοιχείων τους και αλλαγή της διεύθυνσης ηλεκτρονικού ταχυδρομείου τους (Profile)

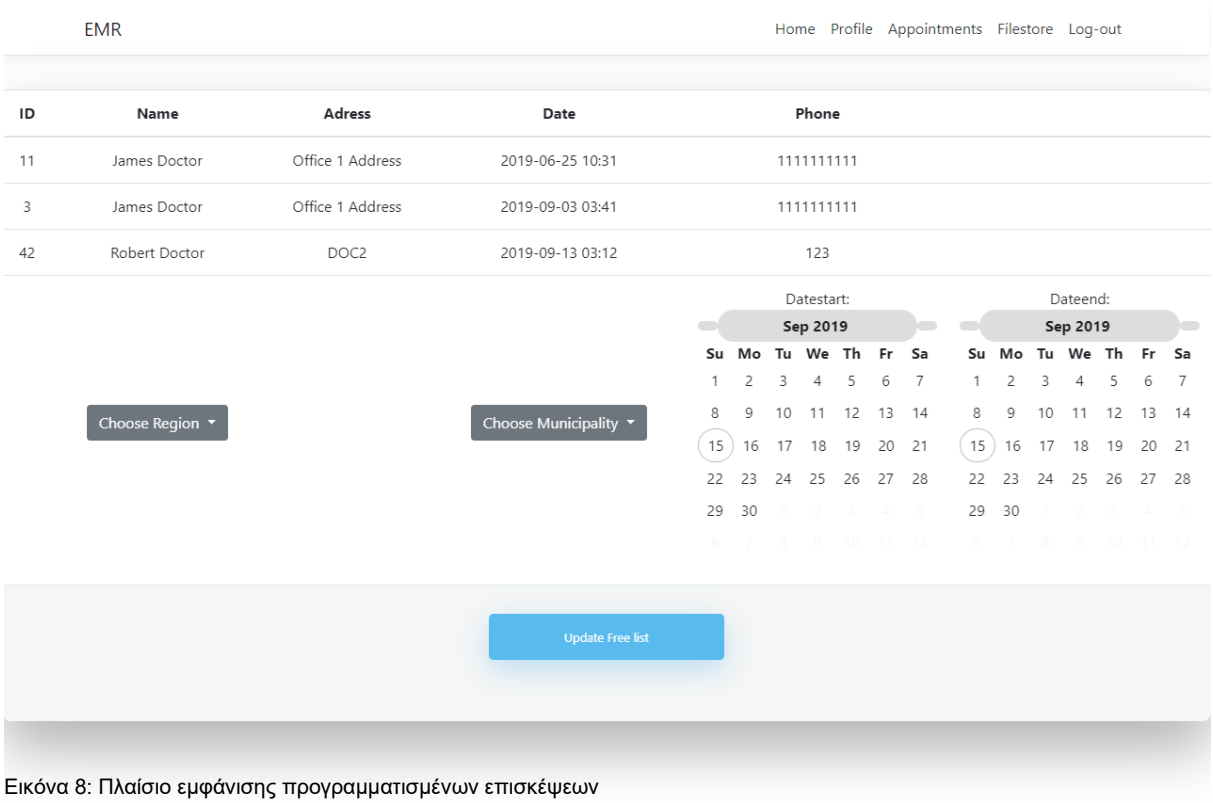


Εικόνα 7: Πλαίσιο επεξεργασίας διεύθυνσης ηλεκτρονικού ταχυδρομείου ασθενών

Στο συγκεκριμένο πλαίσιο ο ενδιαφερόμενος ασθενής μπορεί να αλλάξει το email το οποίο έχει ορίσει κατά την ενεργοποίηση της υπηρεσίας

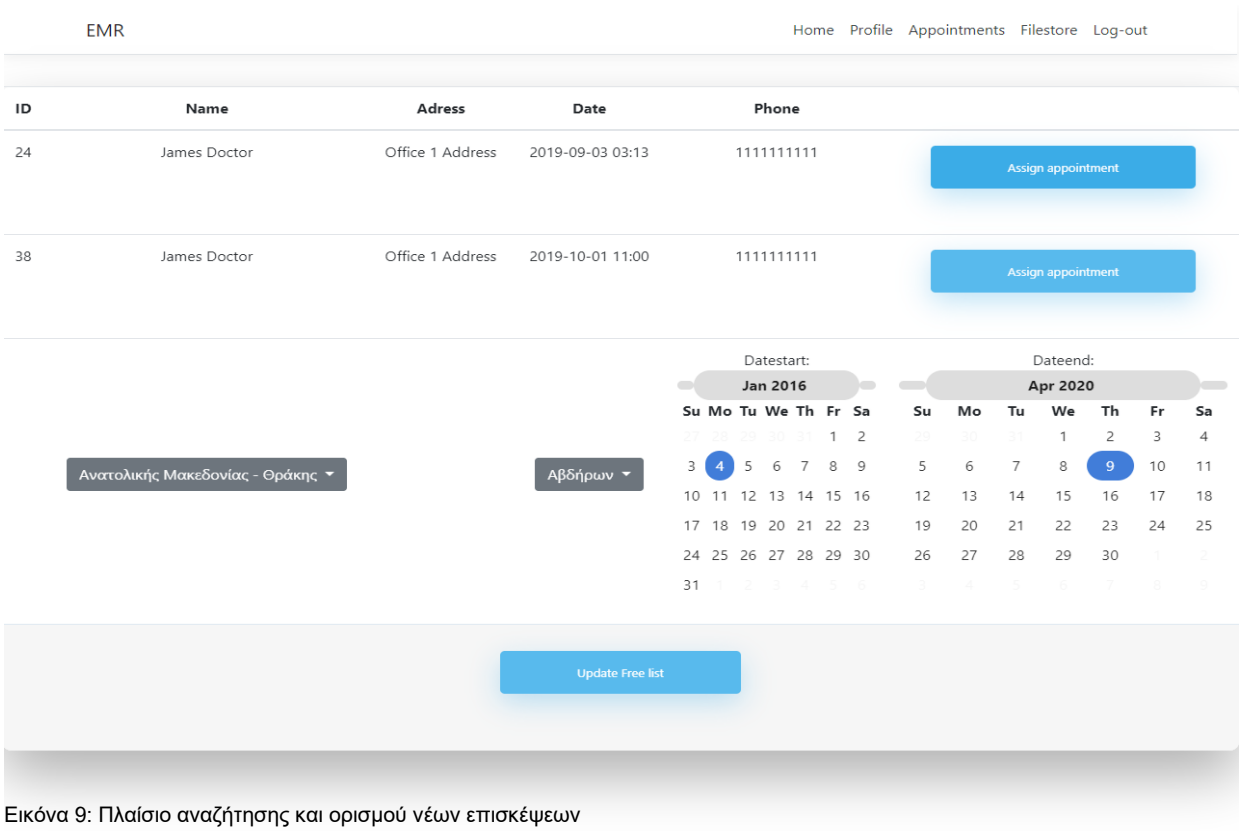
.

- Μπορεί να δει τις επισκέψεις που έχει προγραμματίσει



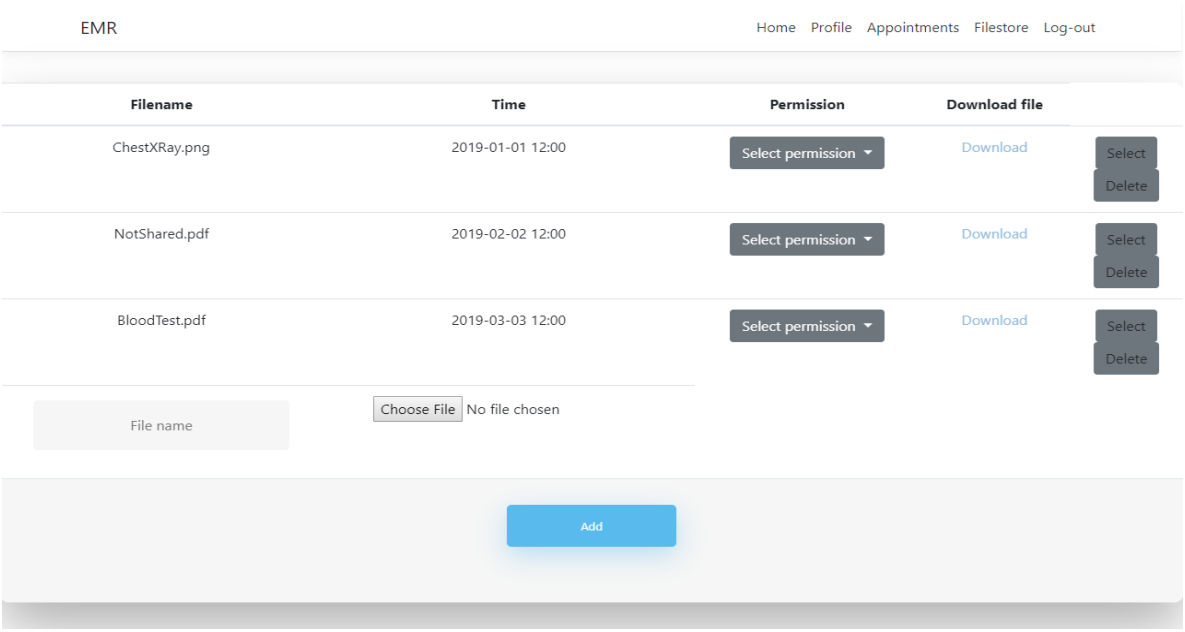
Εικόνα 8: Πλαίσιο εμφάνισης προγραμματισμένων επισκέψεων

- Μπορεί να προγραμματίσει νέες επισκέψεις



Εικόνα 9: Πλαίσιο αναζήτησης και ορισμού νέων επισκέψεων

- Μπορεί να ενημερώσει τον ιατρικό του φάκελο.



Εικόνα 10: Πλαίσιο ανεβάσματος ιατρικών εξετάσεων

-

Με ένα κλικ μπορεί να ανεβάσει τις πιο πρόσφατες εξετάσεις του, με σκοπό την ολοκληρωμένη και έγκυρη διάγνωση από τον θεράποντα ιατρό. Τέλος υπάρχει η δυνατότητα προσθήκης δικαιωμάτων στους ιατρούς, σχετικά με το τι μπορούν να κάνουν με τις εξετάσεις.

Κεφάλαιο 6ο

Αντί επιλόγου

6.1 Συμπεράσματα

Ο ηλεκτρονικός φάκελος υγείας αποτελεί ένα από τα πιο σημαντικά εργαλεία ενός συστήματος υγείας. Το συγκεκριμένο εργαλείο μπορεί να ωφελήσει και να συνδράμει στην ιατρική φροντίδα ενός ασθενούς, παρέχοντας μια ολοκληρωμένη εικόνα στο ιατρικό προσωπικό. Επιπλέον, όπως παρουσιάστηκε και στο τρίτο κεφάλαιο, τα λογισμικά που διατίθενται για αυτό το σκοπό παρέχουν ένα σύνολο λειτουργιών σχετικά με τις αναφορές και τον έλεγχο των αλληλεπιδράσεων που μπορεί να έχει μια ιατρική συνταγή με άλλα φάρμακα που λαμβάνει ο ασθενής. Ως εκ τούτου αντιλαμβάνεται κανείς την αναγκαιότητα της χρήσης αυτών των συστημάτων. Όπως έγινε γνωστό αποτα υπάρχοντα συστήματα λογισμικού που παρέχουν υπηρεσίες ηλεκτρονικού φακέλου υγείας, τα περισσότερα από αυτά ακολουθούν το μοντέλο πελάτη εξυπηρετητή και σε πολλές των περιπτώσεων παρέχουν διαδικτυακή διεπαφή με τον χρήστη. Ένα από τα κύρια προβλήματα που παρουσιάστηκαν αφορά την απομακρυσμένη χρήση τους. Το συγκεκριμένο χαρακτηριστικό αποτελεί τροχοπέδη για τα συγκεκριμένα συστήματα καθώς η αποδεκτή και πλήρως ολοκληρωμένη λύση θα επέτρεπε σε έναν ασθενή να μπορεί οπουδήποτε στον κόσμο να έχει πρόσβαση στον ηλεκτρονικό φάκελο υγείας του. Η διαθεσιμότητα των δεδομένων αποτελεί προϋπόθεση για την παροχή των βέλτιστων υπηρεσιών υγείας. Αυτό απαιτείται καθώς μια διάγνωση ενδεχομένως να απαιτεί το ιατρικό ιστορικό του ασθενούς ή της οικογενείας του. Αυτό βέβαια εγείρει διάφορα ερωτήματα αναφορικά με το επίπεδο της ασφάλειας και της ιδιωτικότητας των προσωπικών δεδομένων.

Όλα τα παραπάνω μας ωθούν στον εντοπισμό κάποιων βασικών χαρακτηριστικών που αφορούν την ανάπτυξη λογισμικών για την παροχή υπηρεσιών ηλεκτρονικού φακέλου ασθενούς. Αυτά είναι τα εξής: α. Η πρόσβαση στα δεδομένα από οπουδήποτε,β.Η διαθεσιμότητα τω δεδομένων, γ. Η ασφάλεια των πληροφοριών. Οι 3 προηγούμενες απαιτήσεις μας οδήγησαν στα πλαίσια της παρούσας εργασίας στη

δημιουργία ενός λογισμικού για την παροχή υπηρεσιών
ηλεκτρονικού φακέλου ασθενούς, αξιοποιώντας τεχνολογίες αιχμής για την παροχή μιας εύχρηστης εφαρμογής, αλλά και την αξιοποίηση του υπολογιστικού σύννεφου για την καθολική πρόσβαση στα δεδομένα από οπουδήποτε και αν απαιτείται. Τέλος θα πρέπει να αναφέρουμε ότι οι πλατφόρμες του υπολογιστικού σύννεφου διασφαλίζουν τόσο τη διαθεσιμότητα των δεδομένων όσο και την ασφάλεια τους. Θα πρέπει να αναφερθεί ότι αν κάποιος χρήστης απαιτεί τη διασφάλιση της εμπιστευτικότητας των δεδομένων ενώ αυτά είναι αποθηκευμένα στο υπολογιστικό σύννεφο, τότε θα πρέπει επιπλέον να κρυπτογραφήσει τα ίδια τα δεδομένα όσο αυτά είναι αποθηκευμένα στον εκάστοτε πάροχου.

6.2 Το μέλλον

Οι πληροφορίες που αφορούν τα δεδομένα υγείας ενός ασθενούς αποτελούν προσωπικά δεδομένα και πιο συγκεκριμένα ευαίσθητα δεδομένα. Παρόλαυτά γίνεται εύκολα αντιληπτή η αναγκαιότητα για παροχή υψηλού επιπέδου υπηρεσιών υγείας οπουδήποτε στον κόσμο. Για την επίτευξη αυτού του στόχου απαιτείται η εύκολη πρόσβαση στα δεδομένα υγείας ενός ασθενούς. Ας φανταστούμε την περίπτωση κατά την οποία ένας ασθενής θα ήταν σε θέση να ταξιδέψει οπουδήποτε στον κόσμο, γνωρίζοντας ότι ανεξάρτητα από την ασθένεια ή την πάθηση που θα είχε προσβληθεί ο ασθενής, οι πάροχοι των υπηρεσιών υγείας θα είχαν πλήρη πρόσβαση στα δεδομένα υγείας του ασθενούς. Η πληροφορία θα μπορούσε να διαμοιράζεται μέσω ενός διεθνούς δικτύου το οποίο θα διασφάλιζε την ασφάλεια των πληροφοριών. Σε μια διαφορετική περίπτωση θα ήταν δυνατή η χρήση έξυπνων καρτών (smart cards) ή ακόμα και η εμφύτευση ολοκληρωμένων κυκλωμάτων στο σώμα του ασθενούς. Σε κάθε περίπτωση τίθενται ζητήματα προσωπικών δεδομένων. Επομένως ο σχεδιασμός και η υλοποίηση αυτών των συστημάτων απαιτεί τη συνέργεια πολλών επιστημόνων για την πλήρη διασφάλιση της ιδιωτικότητας των χρηστών.

Βιβλιογραφία

[1] Othman, Mazliza, Sajjad Ahmad Madani, and Samee Ullah Khan. "A survey of mobile cloud computing application models." IEEE Communications Surveys & Tutorials 16.1 (2013): 393-413.

[2] Syed, Basarat. Beginning Node. js. Apress, 2014.

[3] Darwin, Peter Bacon, and Pawel Kozlowski. AngularJS web application development. Packt Publ., 2013.

[4] Freeman, Adam. Pro Asp. net Core Mvc. Apress, 2016.

[5] Copeland, Marshall, et al. Microsoft Azure. New York, NY, USA:: Apress, 2015.

[6] Wang, Samuel J., et al. "A cost-benefit analysis of electronic medical records in primary care." The American journal of medicine 114.5 (2003): 397-403.

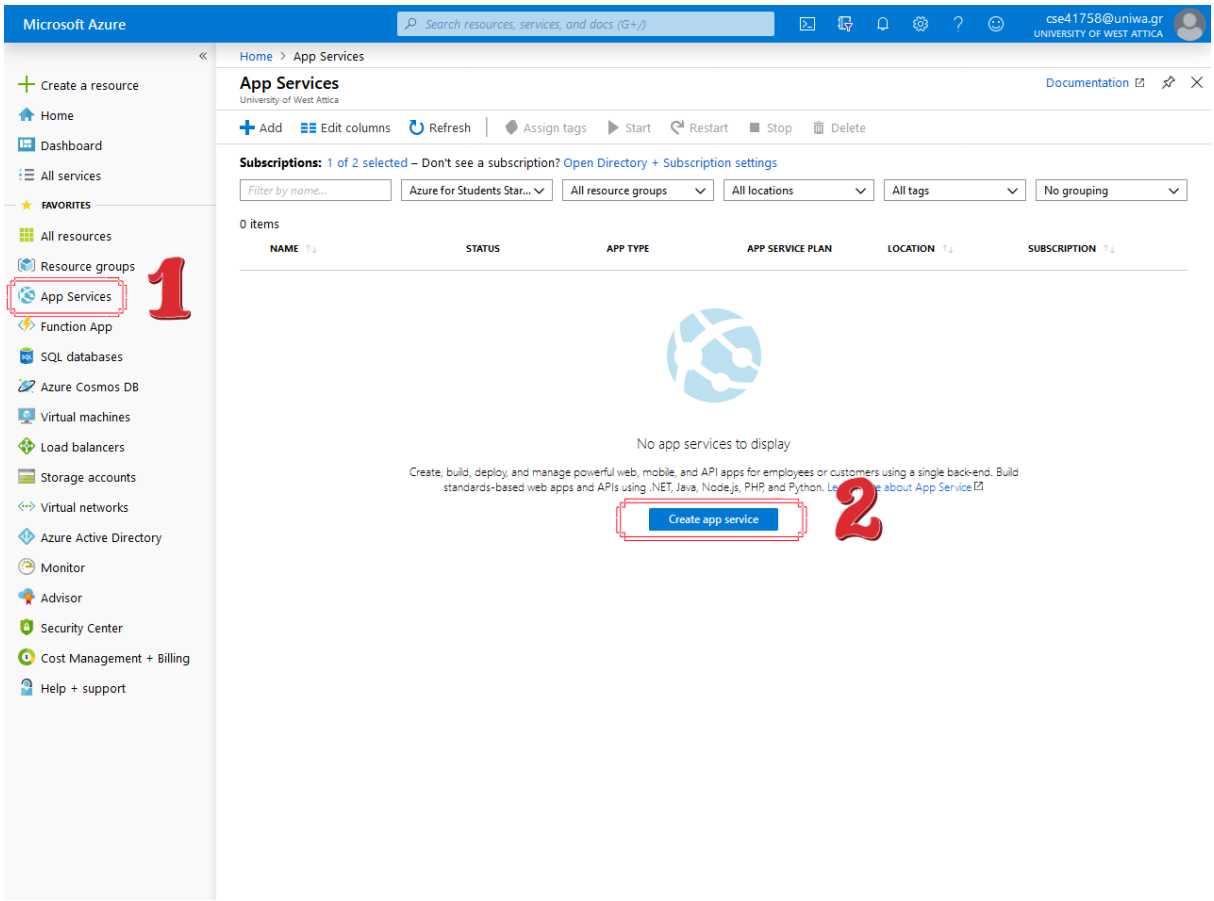
Μέρος 2ο

Υπόμνημα

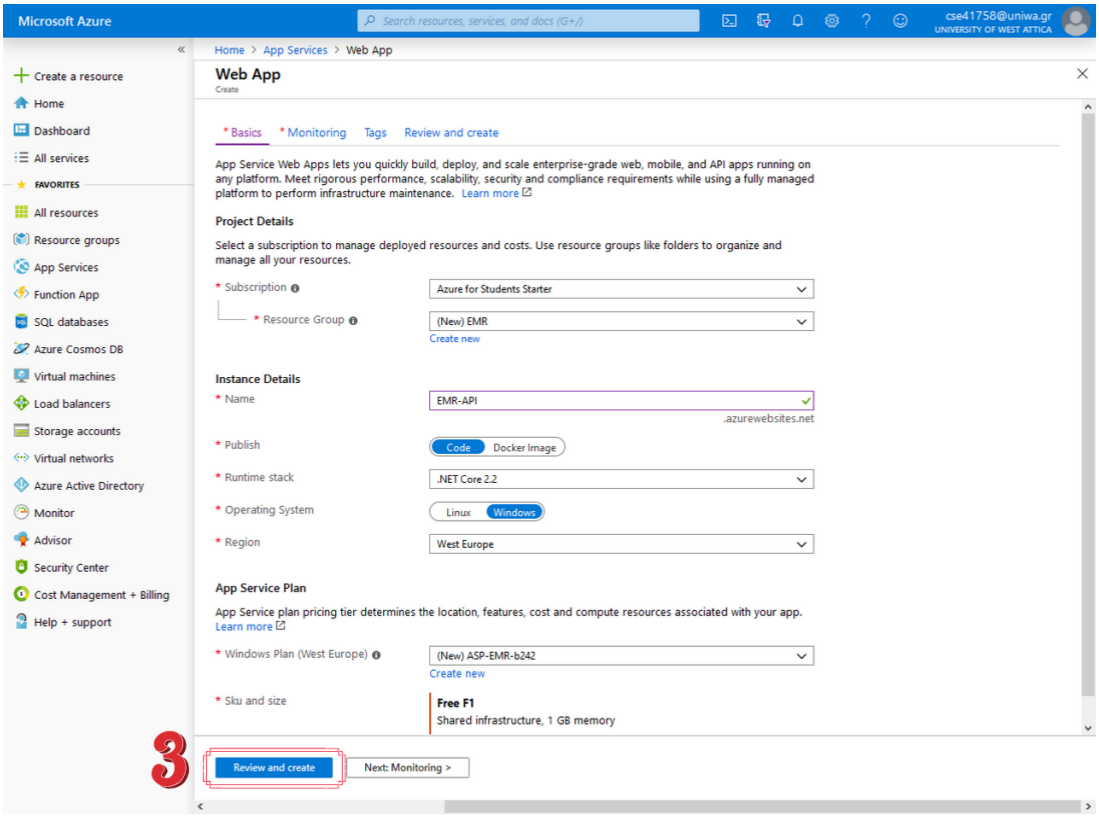
Σε αυτό το μέρος της εργασίας παραθέτονται εικόνες που περιγράφουν την διαδικασία δημιουργίας 3 κομβικής σημασίας, υλοποιήσεων για την ολοκλήρωση του project.

Υλοποίηση 1η : API

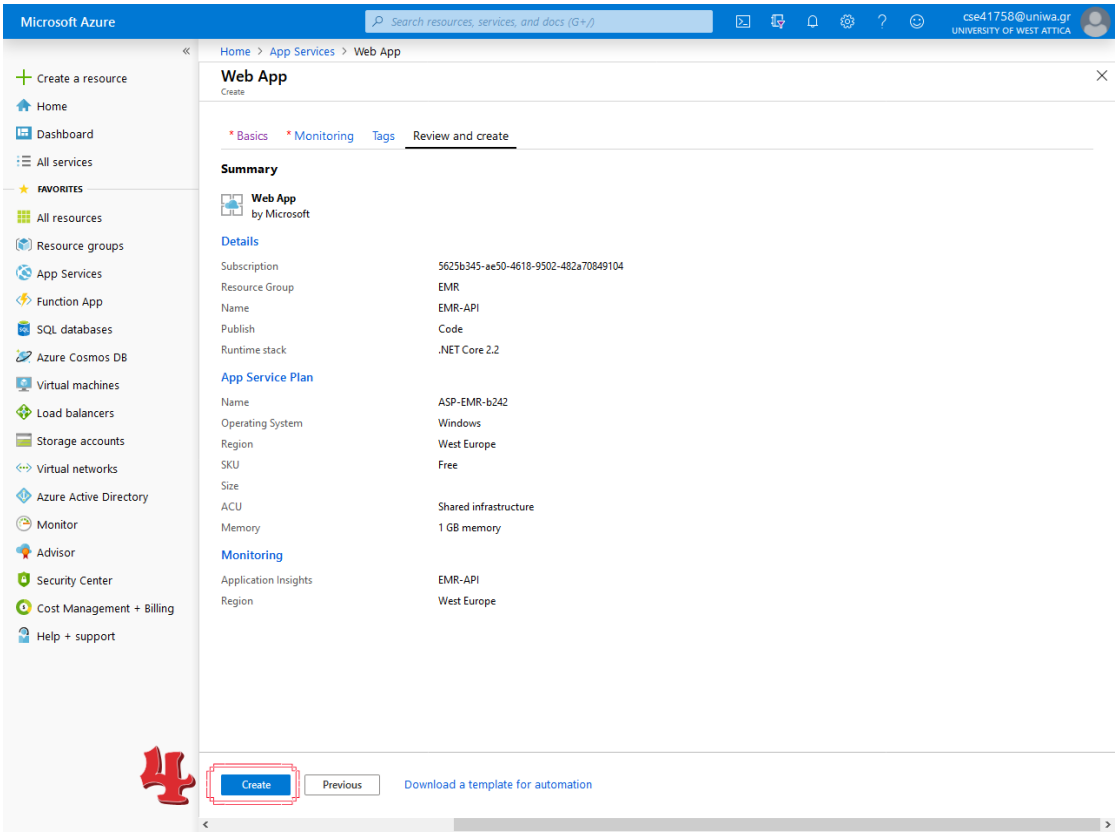
Η υπηρεσία API δημιουργείται μέσα από το πλαίσιο ελέγχου της υπηρεσίας Microsoft Azure Portal.



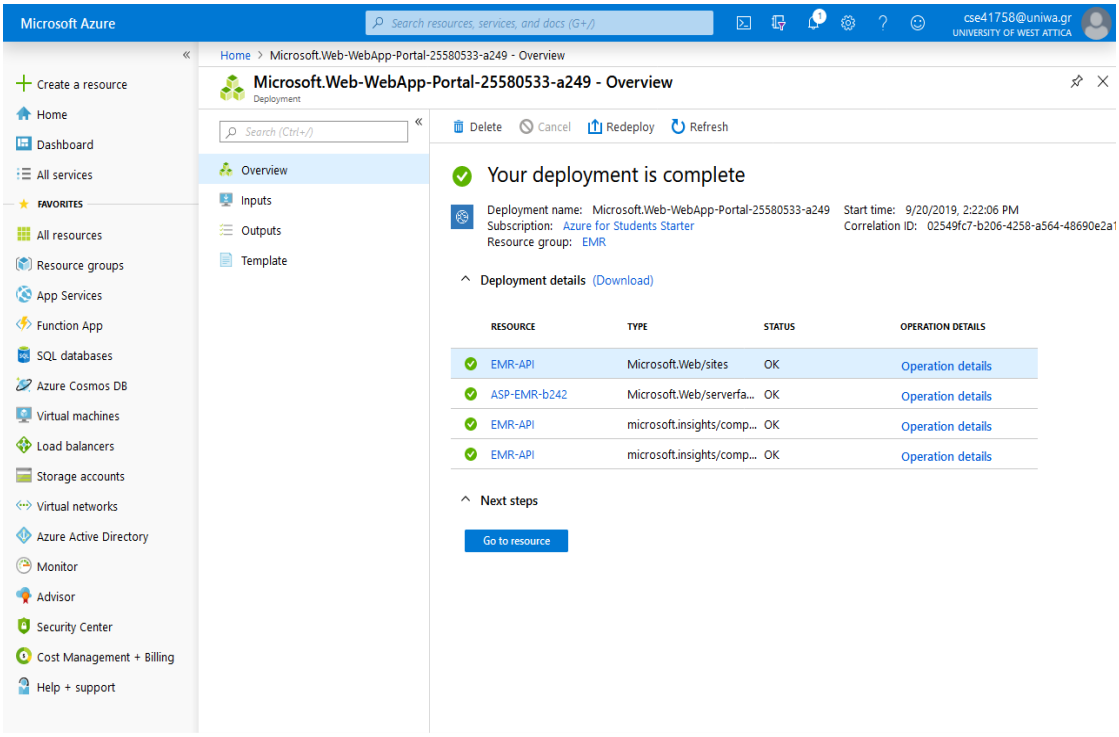
Κάνουμε κλικ στην καρτέλα **App services** (βήμα 1) και στην συνέχεια κάνουμε κλίκ στο κουμπί **create app service** (βήμα 2).



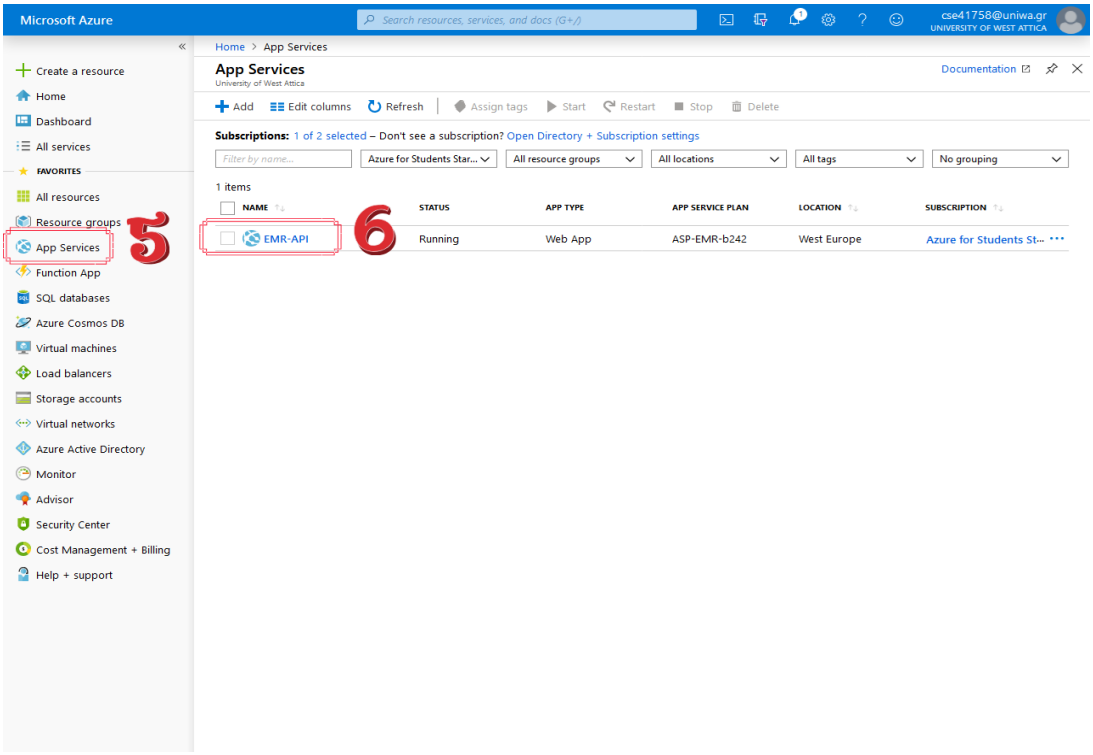
Αφου συμπληρώσουμε τα πεδία με τις επιλογές σχετικά με τις λεπτομέρειες της εφαρμογής πατάμε το πλήκτρο **review and create** (βήμα 3).



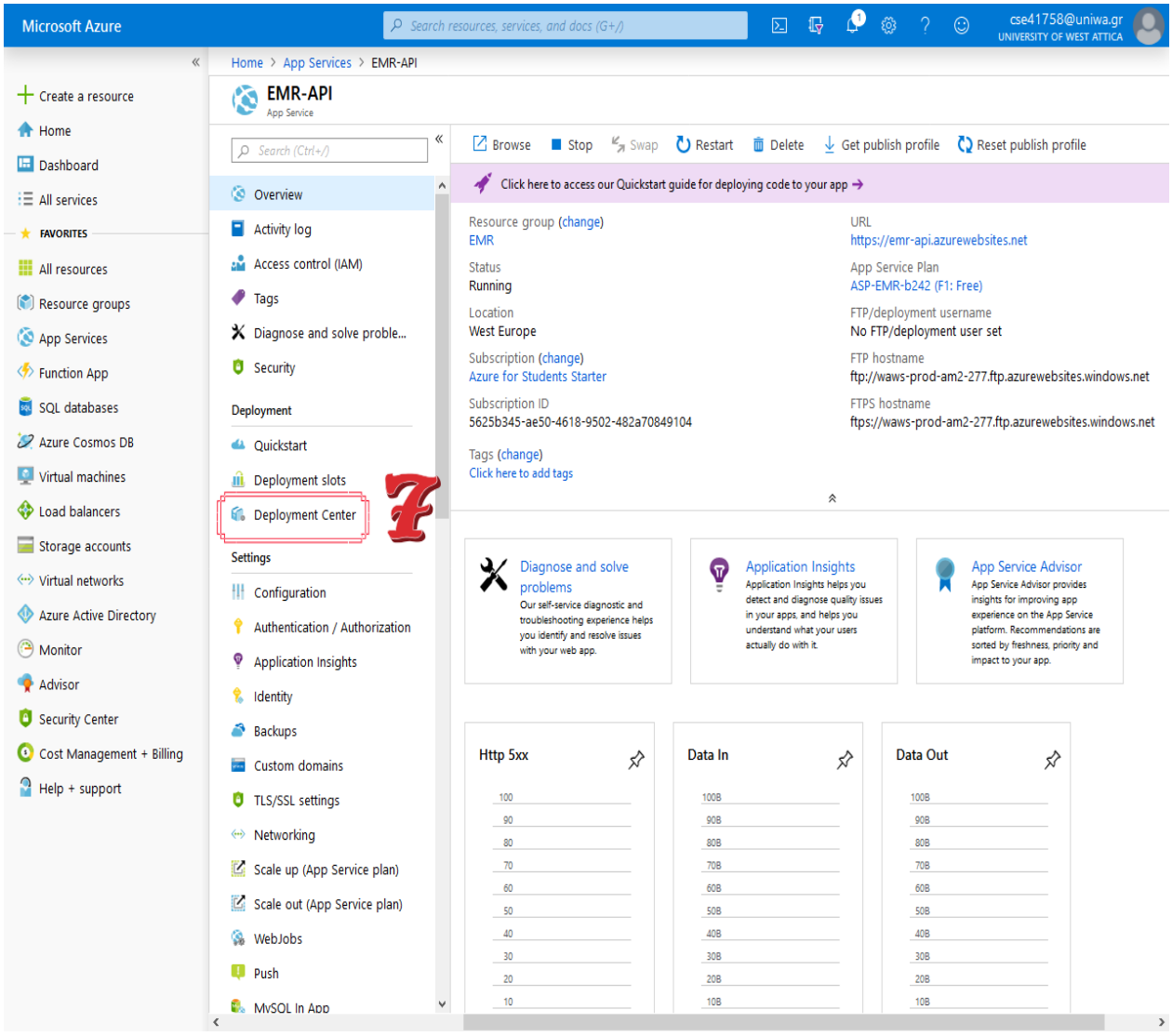
Στην συνέχεια μας παρουσιάζεται μια σύνοψη των επιλογών που έχουμε κάνει μέχρι τώρα και πατάμε το πλήκτρο **Create** (βήμα 4).



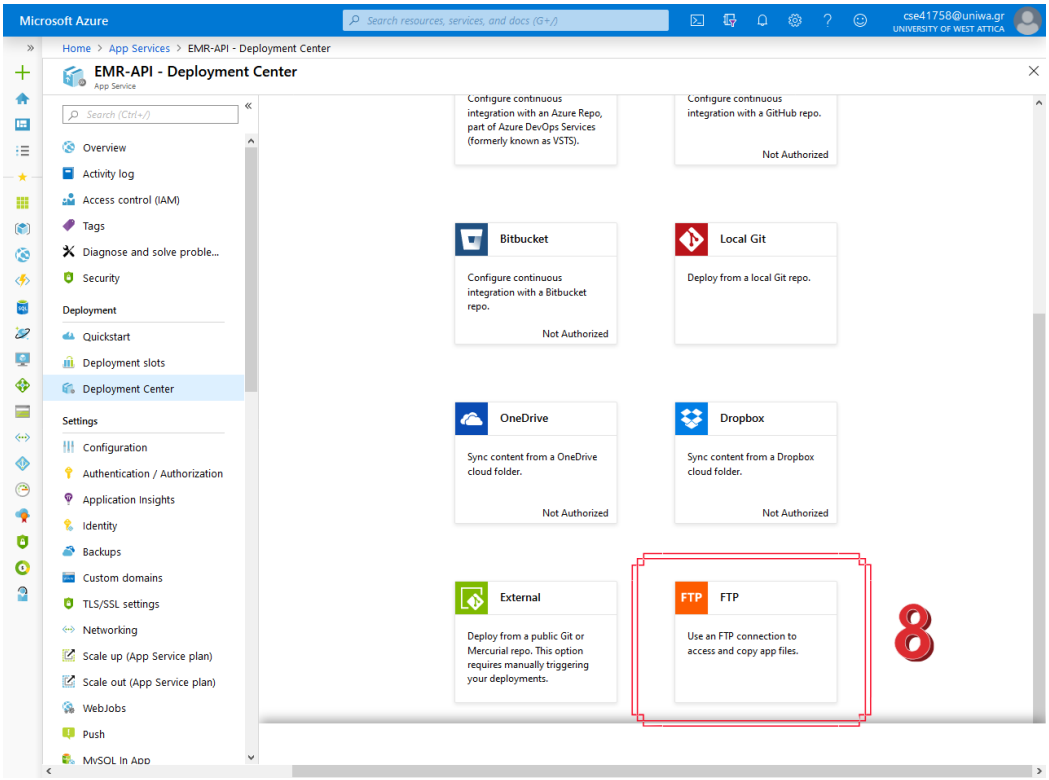
Μας εμφανίζεται η παραπάνω εικόνα με το μήνυμα επιτυχίας δημιουργίας της εφαρμογής.



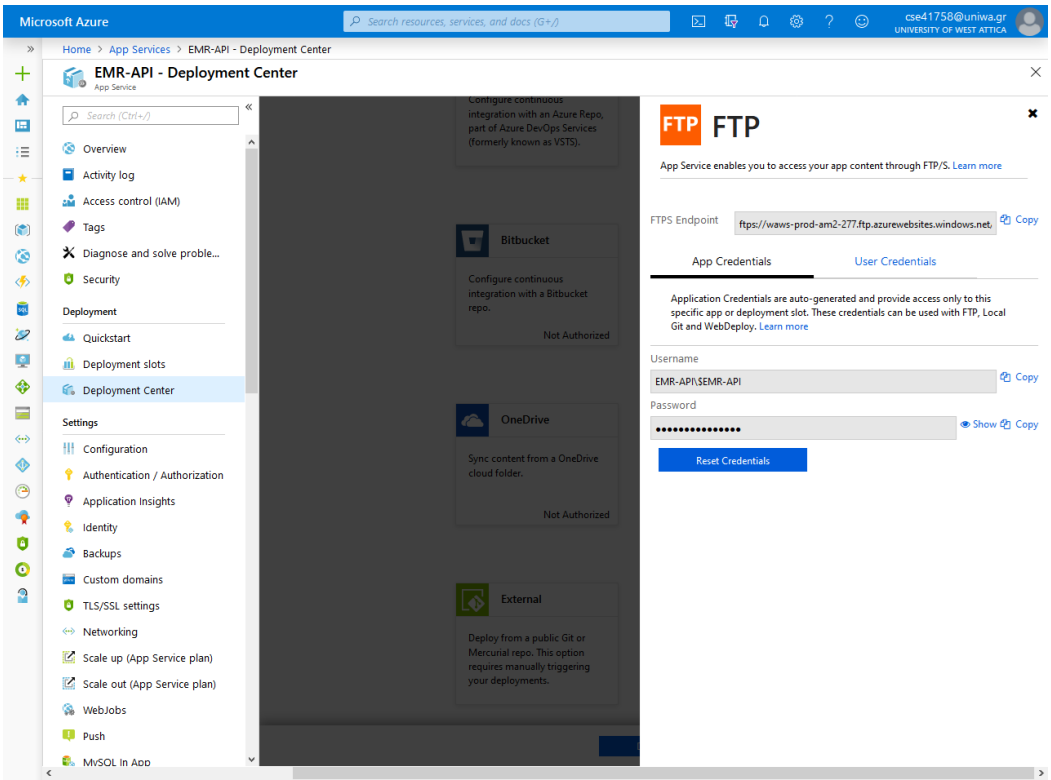
Στην συνέχεια επιστρέφουμε στην καρτέλα **App Services** (βήμα 5) και βλέπουμε ότι η υπηρεσία που μόλις δημιουργήσαμε κάνοντας τα παραπάνω βήματα, είναι ενεργή Όπως βλέπουμε και στην εικόνα το API που μόλις δημιουργήσαμε ονομάζεται **EMR_API** (το είχαμε δηλώσει πιο πάνω όταν ορίζαμε τις λεπτομέρειες). Κάνουμε κλικ επάνω στο EMR_API (βήμα 6) και στην συνέχεια επιλέγουμε την καρτέλα **deployment center** (βήμα 7) που βρίσκεται στο menu επιλογών **Deployment.**

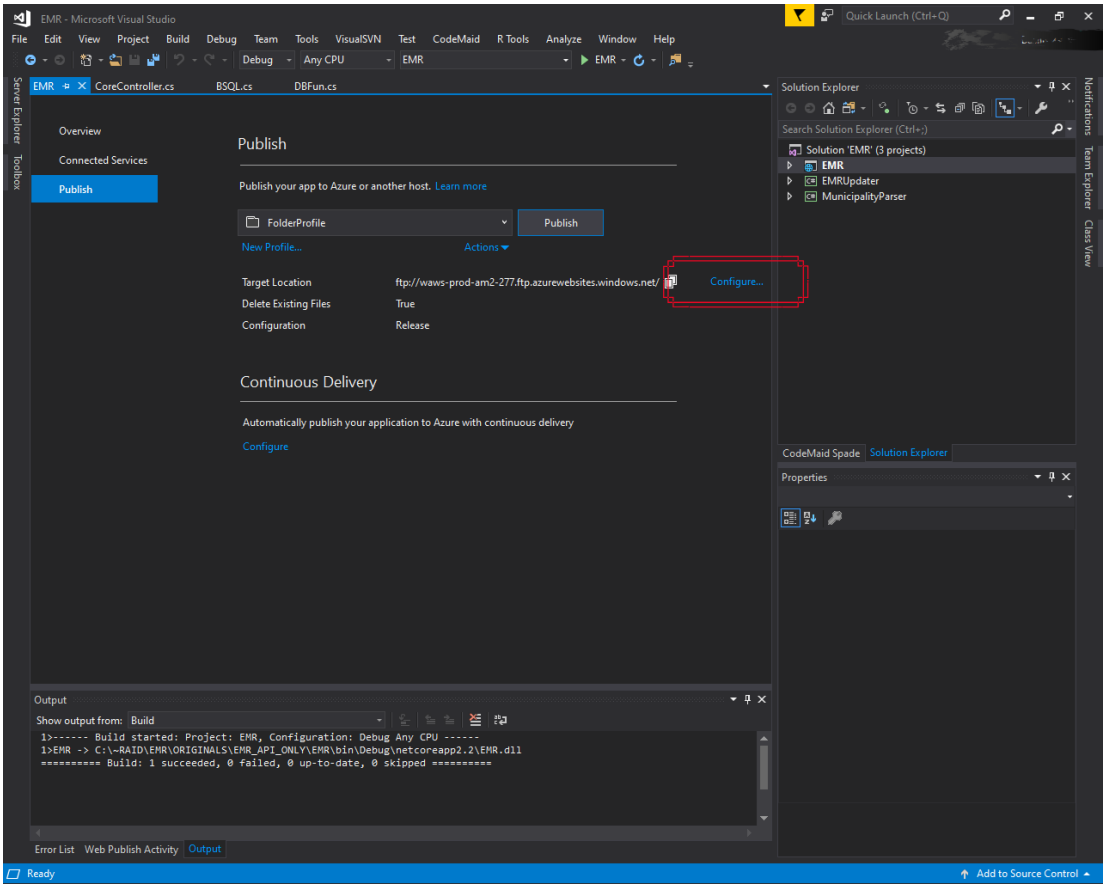


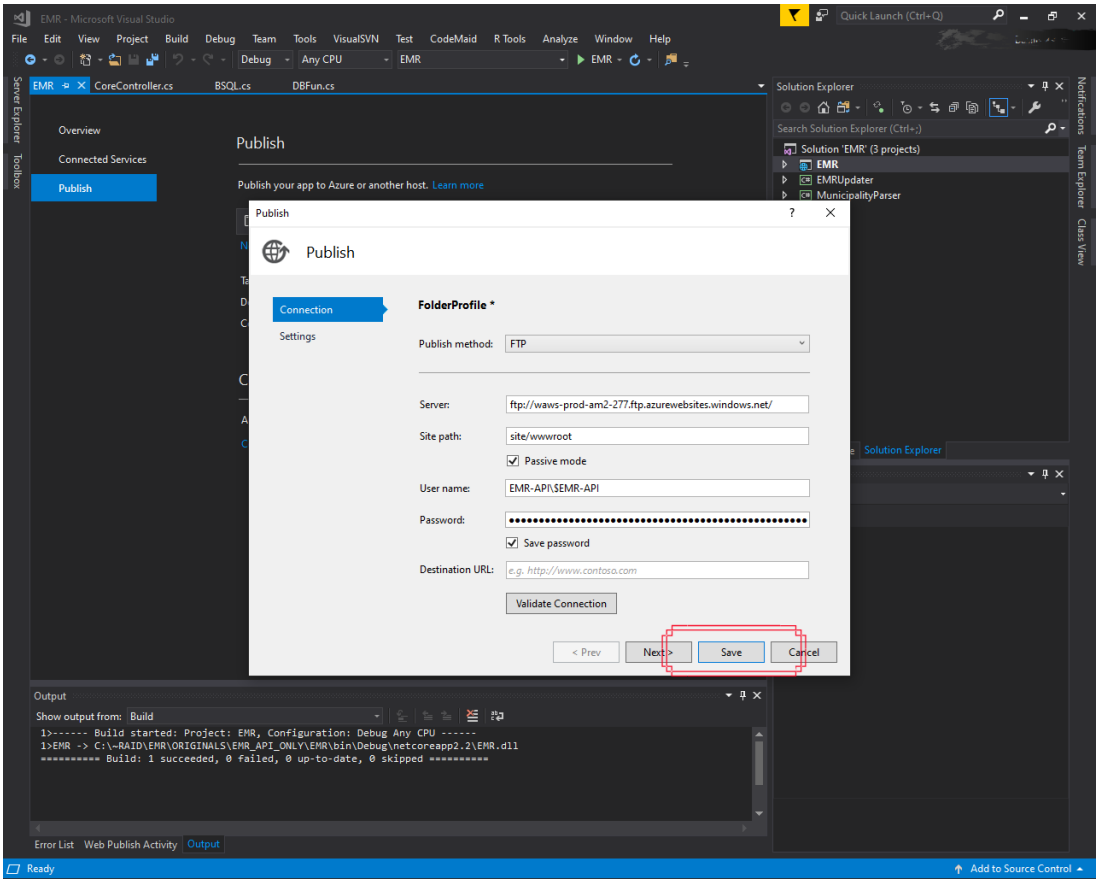
Στην συνέχεια στο πλαίσιο που εμφανίζεται κάνουμε κλικ στο πλαίσιο FTP (βήμα 8) με σκοπό να πάρουμε τις ρυθμίσεις FTP.

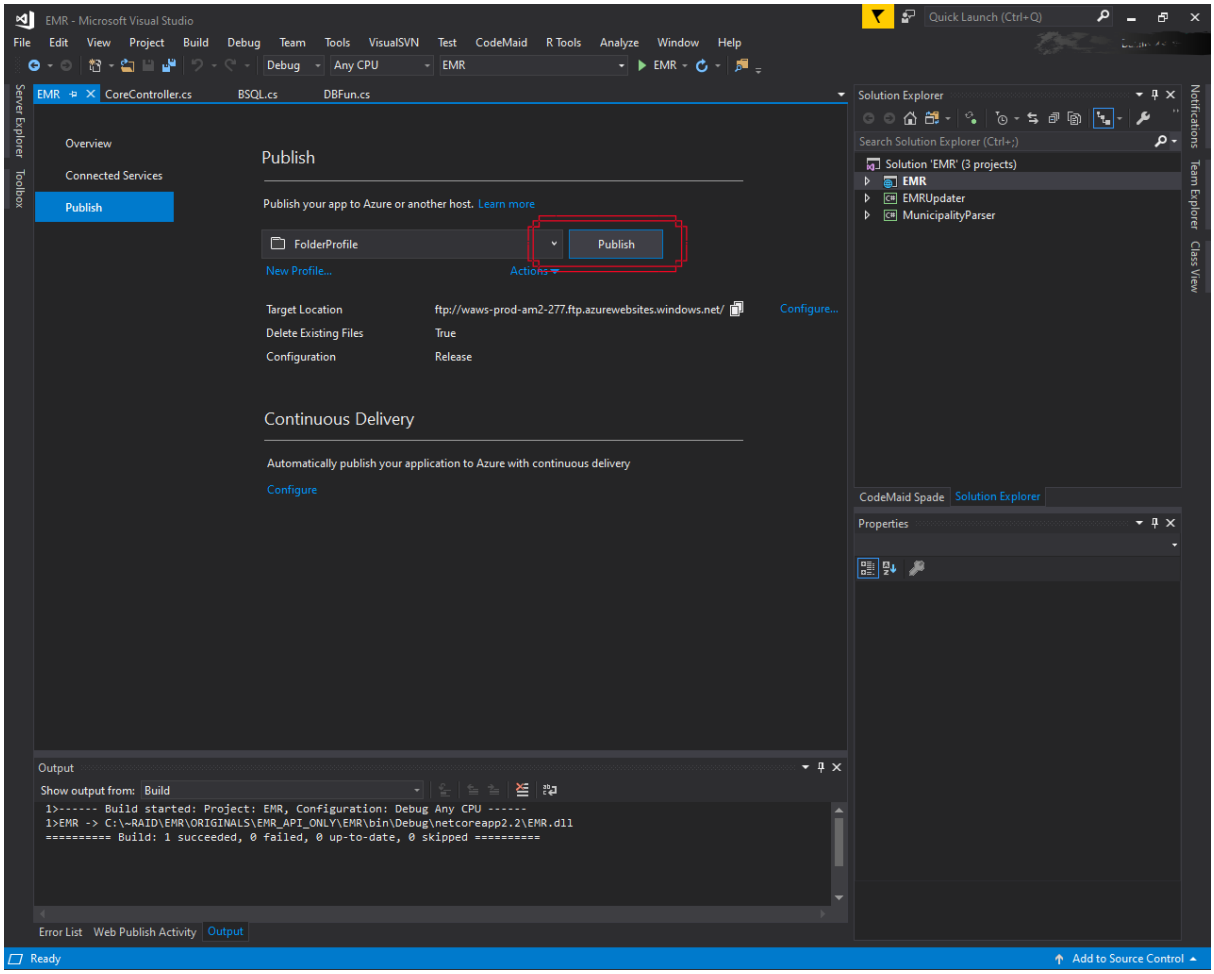


Στο παρακάτω πλαίσιο βλέπουμε τις ρυθμίσεις FTP τις οποίες της κάνουμε αντιγραφή και κλείνουμε το παράθυρο.




EMR - Microsoft Visual Studio
Publish
Overview
Connected Services
Publish
Publish your app to Azure or another host. Learn more
FolderProfile
Publish
New Profile...
Actions
Target Location
ftp://waws-prod-am2-277.ftp.azurewebsites.windows.net/
Configure...
Delete Existing Files
True
Configuration
Release
Continuous Delivery
Automatically publish your application to Azure with continuous delivery
Configure
Solution Explorer
Solution 'EMR' (3 projects)
EMR
EMRUpdater
MunicipalityParser
Output
Ready
Add to Source Control


EMR - Microsoft Visual Studio
Publish
Publish
Connection
Settings
FolderProfile *
Publish method:
FTP
Server:
ftp://waws-prod-am2-277.ftp.azurewebsites.windows.net/
Site path:
site/wwwroot
Passive mode
User name:
EMR-API\$EMR-API
Password:
Save password
Destination URL:
e.g. http://www.contoso.com
Validate Connection
< Prev
Next >
Save
Cancel
Output
Ready
Add to Source Control

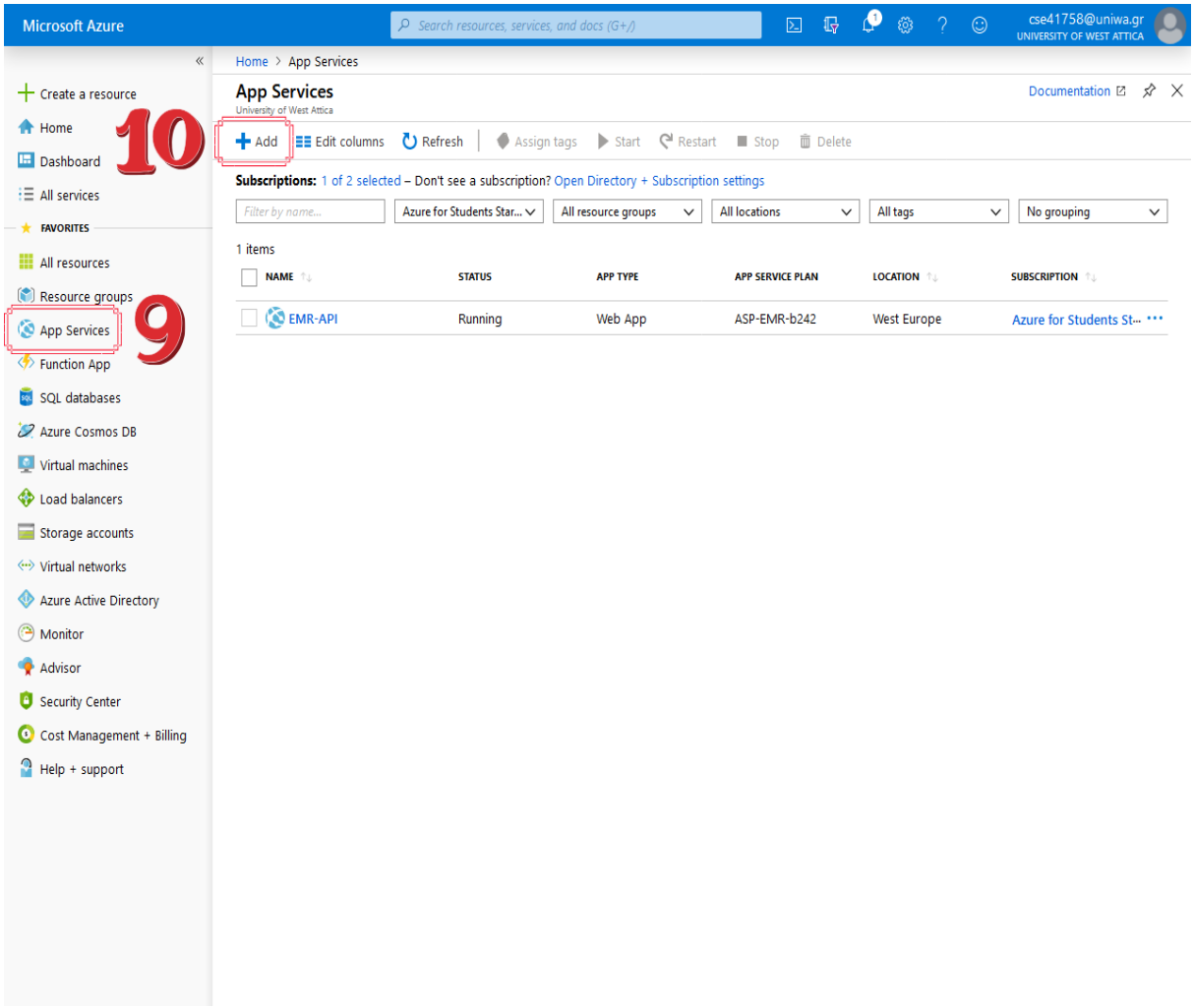


Παραπάνω σας παρουσιάζεται η διαδικασία κατά την οποία δημοσιοποιείται «publish» η εφαρμογή μέσω του visual studio χρησιμοποιώντας τα στοιχεία FTP τα οποία πήραμε από τα παραπάνω βήματα.

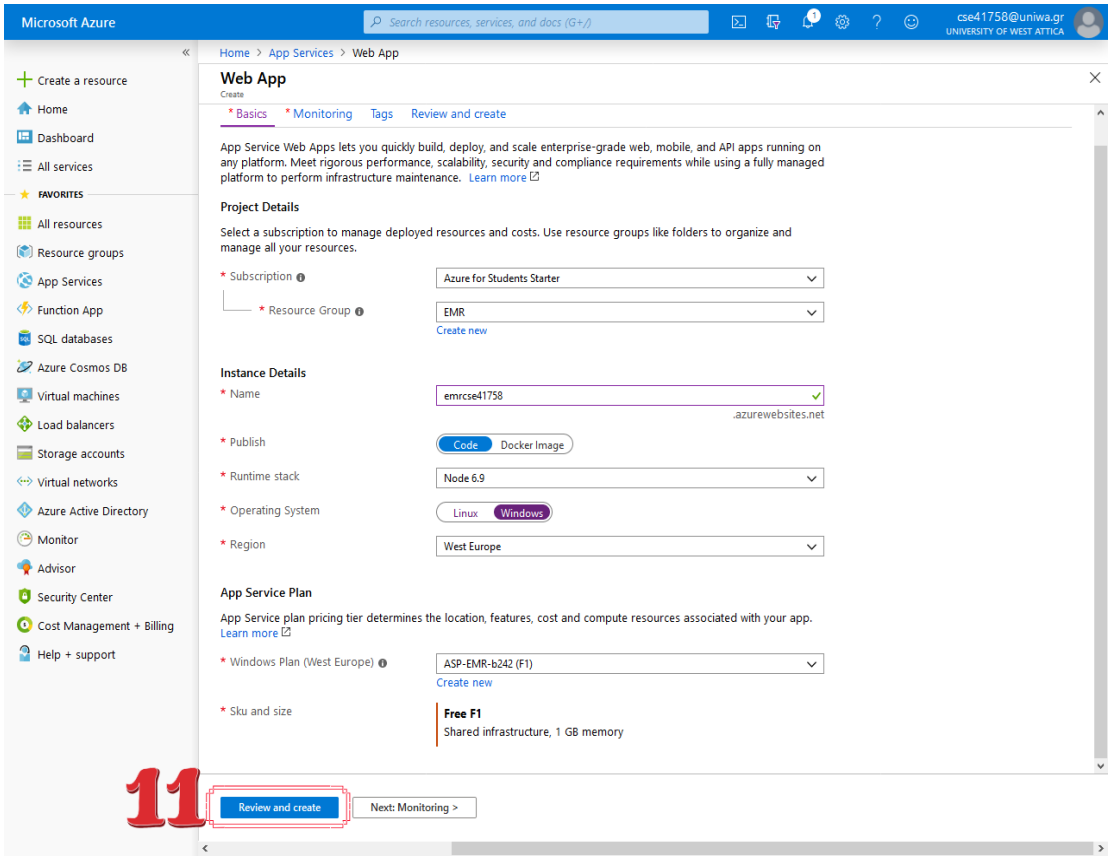
Υλοποίηση 2η : Web App Υπηρεσία

Η Web app υπηρεσία που χρησιμοποιήθηκε για την ολοκλήρωση του project δημιουργήθηκε και αυτή μέσα από το πλαίσιο ελέγχου της υπηρεσίας Microsoft Azure Portal.

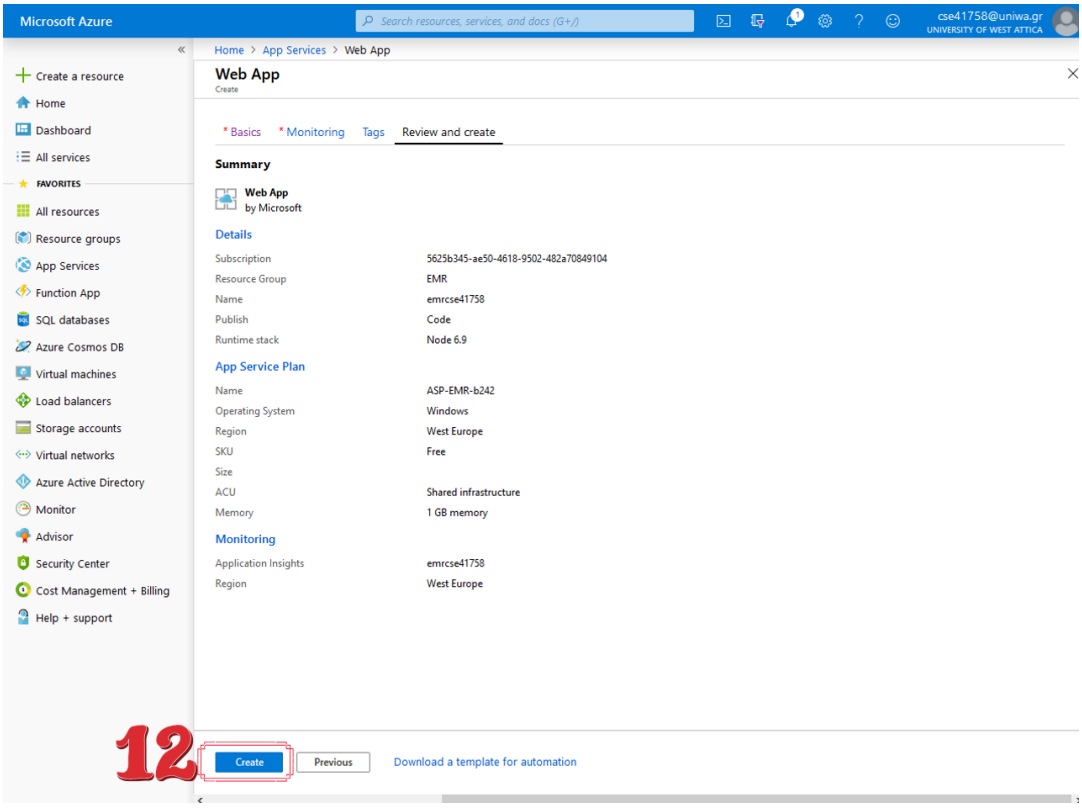
Όπως βλέπουμε και στην εικόνα παρακάτω επιλέγουμε ξανά την καρτέλα **App Services** (βήμα 9) και στην συνέχεια πατάμε το πλήκτρο **Add** (βήμα 10).



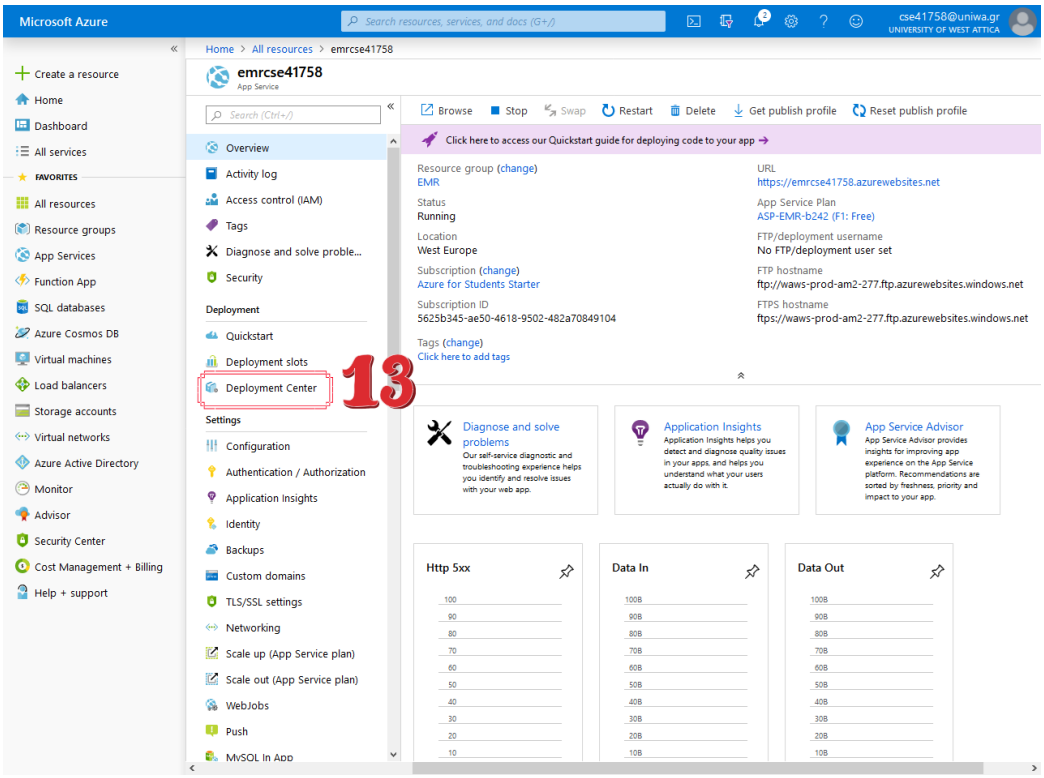
Συμπληρώνουμε τα πεδία που εμφανίζονται στην καρτέλα BASICS με τις επιθυμητές επιλογές για την υπηρεσία μας και πατάμε το πλήκτρο **review and create** (βήμα 11).



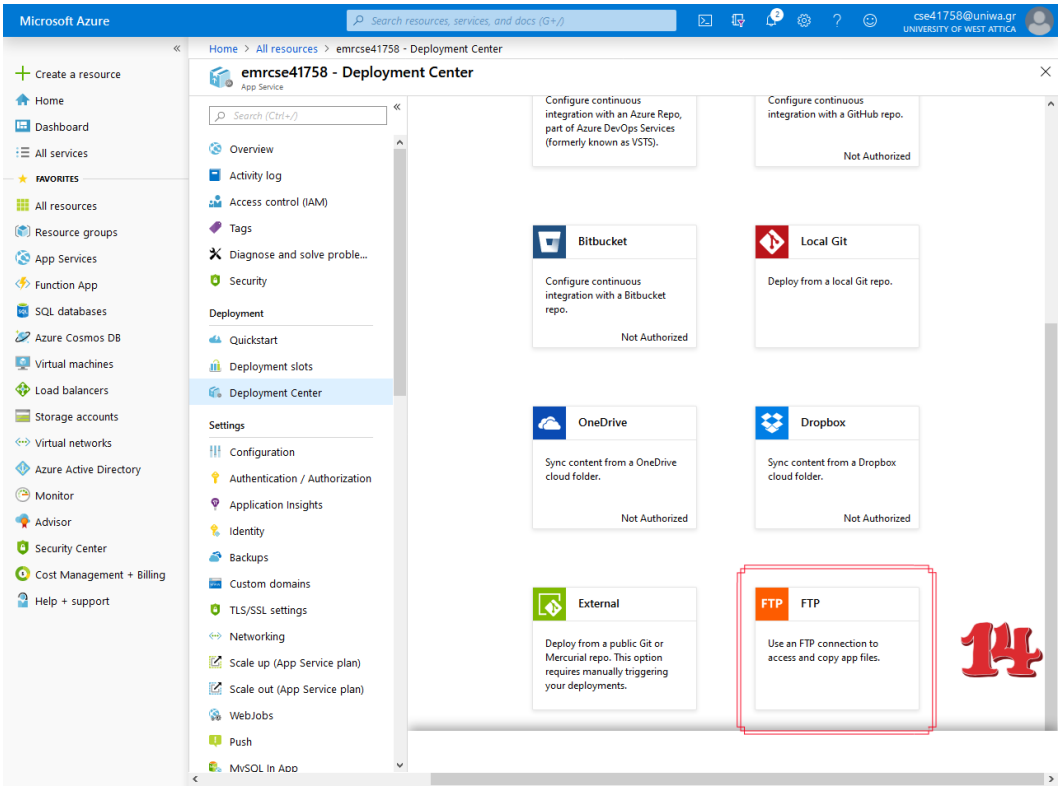
Στην συνέχεια πατάμε **create** (βήμα 12).



Στην συνέχεια επιλέγουμε την καρτέλα **Deployment Center** (βήμα 13).



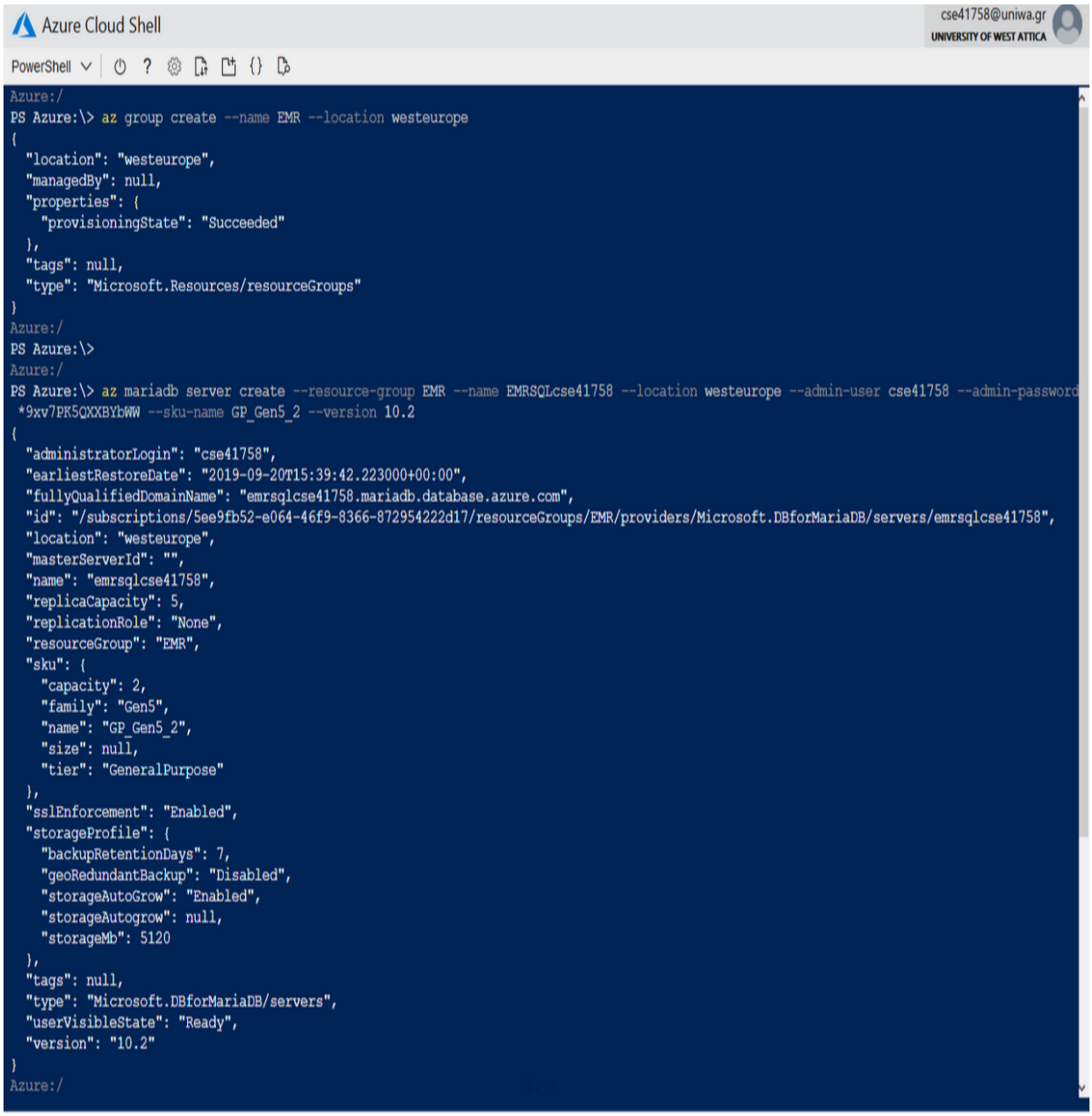
Και μέσα από την επιλογή FTP (βήμα 14) παίρνουμε στοιχεία FTP τα οποία χρησιμοποιούμε για να ανεβάσουμε την εφαρμογή.



Υλοποίηση 3η: Δημιουργία Βάσης Δεδομένων

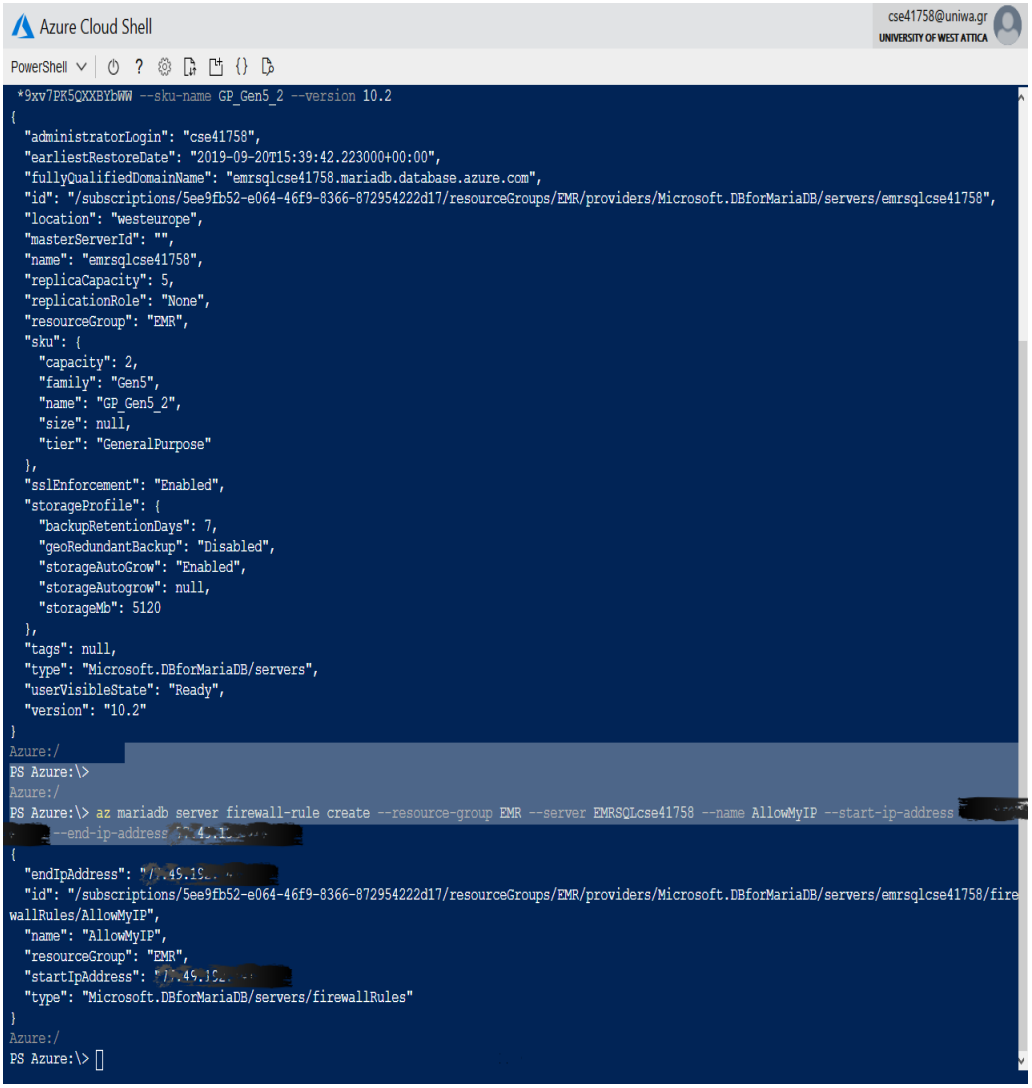
Παρακάτω παρουσιάζεται η δημιουργία της βάσης δεδομένων χρησιμοποιώντας την υπηρεσία Azure Cloud Shell.

Σε 1η φάση δημιουργούμε το resource group και την υπηρεσία MariaDB.



Στην συνέχεια δημιουργούμε έναν κανόνα τείχους προστασίας για να μπορούμε να αποκτήσουμε πρόσβαση από τον τοπικό μας υπολογιστή, έτσι ώστε να ανεβάσουμε

και να επεξεργαστούμε την βάση δεδομένων.



Τέλος ενεργοποιούμε τον event scheduler.