Πρόγραμμα Μεταπτυχιακών Σπουδών
«Επιστήμη και Τεχνολογία της Πληροφορικής και των Υπολογιστών»

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΣΥΣΤΗΜΑΤΑ ΠΛΗΘΟΠΟΡΙΣΜΟΥ ΣΤΗΝ ΑΝΩΤΑΤΗ ΕΚΠΑΙΔΕΥΣΗ

ΘΕΟΦΑΝΗΣ ΚΑΡΑΜΙΧΑΛΕΛΗΣ
Α.Μ.: mcse19002

ΕΠΙΒΛΕΠΟΝΤΕΣ ΚΑΘΗΓΗΤΕΣ: Χ. Σκουρλάς, Α. Μαρινάγη

ΑΘΗΝΑ, ΙΑΝΟΥΑΡΙΟΣ 2021

Μέλη εξεταστικής επιτροπής

Christos Skourlas

Digitally signed by Christos Skourlas
Date: 2021.01.27 12:20:47 +02'00'

1. Χρήστος Σκουρλάς

2. Αικατερίνη Μαρινάγη

3. Βασίλειος Μάμαλης

ΔΗΛΩΣΗ ΣΥΓΓΡΑΦΕΑ ΜΕΤΑΠΤΥΧΙΑΚΗΣ ΕΡΓΑΣΙΑΣ

Ο κάτωθι υπογεγραμμένος Θεοφάνης Καραμιχαλέλης του Δημητρίου, με αριθμό μητρώου mcse19002 φοιτητής του Προγράμματος Μεταπτυχιακών Σπουδών Επιστήμη και Τεχνολογία της Πληροφορικής και των Υπολογιστών του Τμήματος Μηχανικών Πληροφορικής και Υπολογιστών της Σχολής Μηχανικών του Πανεπιστημίου Δυτικής Αττικής, δηλώνω ότι:

«Είμαι συγγραφέας αυτής της μεταπτυχιακής εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της, είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, οι όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών ή λέξεων, είτε ακριβώς είτε παραφρασμένες, αναφέρονται στο σύνολό τους, με πλήρη αναφορά στους συγγραφείς, τον εκδοτικό οίκο ή το περιοδικό, συμπεριλαμβανομένων και των πηγών που ενδεχομένως χρησιμοποιήθηκαν από το διαδίκτυο. Επίσης, βεβαιώνω ότι αυτή η εργασία έχει συγγραφεί από μένα αποκλειστικά και αποτελεί προϊόν πνευματικής ιδιοκτησίας τόσο δικής μου, όσο και του Ιδρύματος.

Παράβαση της ανωτέρω ακαδημαϊκής μου ευθύνης αποτελεί ουσιώδη λόγο για την ανάκληση του πτυχίου μου».

Επιθυμώ την απαγόρευση πρόσβασης στο πλήρες κείμενο της εργασίας μου μέχρι 19/01/2021 και έπειτα από αίτηση μου στη Βιβλιοθήκη και έγκριση του επιβλέποντα καθηγητή.

Ο Δηλών

Θεοφάνης Καραμιχαλέλης

Περιεχόμενα

Ευχαριστίες

Ευχαριστώ θερμά τους επιβλέποντες καθηγητές για την πολύτιμη καθοδήγηση τους.

Θα ήθελα να αφιερώσω την εργασία μου στην οικογένεια μου και ειδικότερα στον αγαπημένο μου πατέρα.

Περίληψη

Ένα άτομο δεν μπορεί να κατέχει όλες τις διαθέσιμες γνώσεις και δεξιότητες, ούτε καν μια μικρή ομάδα ατόμων μπορεί. Για αυτό από την αρχαιότητα παρατηρούνται φαινόμενα όπου εργασίες ανατίθενται στο πλήθος. Το φαινόμενο αυτό ονομάζεται πληθοπορισμός.

Σύμφωνα με τους Estellés-Arolas και González Ladrón-de-Guevara, ο πληθοπορισμός είναι συλλογική διαδικτυακή δραστηριότητα ομάδων εθελοντών, η οποία οργανώνεται μετά από ανοικτή πρόσκληση και επιβλέπεται από ένα ίδρυμα ή ένα μη κερδοσκοπικό οργανισμό ή και εταιρεία.

Πολλά πανεπιστήμια της Αμερικής και της Ευρώπης χρησιμοποιούν πλατφόρμες πληθοπορισμού, καθώς θεωρούν ότι ενθαρρύνει τόσο τους φοιτητές όσο και τους καθηγητές στη διαδικασία της συνδημιουργίας και της ανταλλαγής γνώσεων.

Η πλατφόρμα πληθοπορισμού «UniCrowd» που αναπτύχθηκε στα πλαίσια αυτής της διπλωματικής εργασίας, επιτρέπει σε πανεπιστήμια να δημιουργούν τις δικές τους εκστρατείες. Οι εθελοντές, με τη σειρά τους, μπορούν να συμμετέχουν σε οποιαδήποτε από τις υπάρχουσες εκστρατείες, συνεισφέροντας ενεργά με τις δικές τους γνώσεις και δεξιότητες.

Λέξεις – Κλειδιά: πληθοπορισμός, ανώτατη εκπαίδευση, πλατφόρμα, Microsoft, C#, ASP.NET Core, SQL, Server, MongoDB, microservices, Docker, Ocelot.

Abstract

One person may not possess all the available knowledge and skills, not even a small group of people can. That is why from antiquity there are phenomena where projects are assigned to the crowd. This phenomenon is called crowdsourcing.

According to Estellés-Arolas and González Ladrón-de-Guevara, crowdsourcing is a collective online activity of groups of volunteers, organized by open invitation and supervised by an institution or a non-profit organization or company.

Many universities in America and Europe use crowdsourcing platforms, as they believe that it encourages both students and professors in the process of co-creating and sharing knowledge.

The "UniCrowd" crowdsourcing platform developed as part of this dissertation allows universities to create their own campaigns. Volunteers, in turn, can participate in any of the existing campaigns, actively contributing with their own knowledge and skills.

Keywords: crowdsourcing, higher education, platform, Microsoft, C #, ASP.NET Core, SQL, Server, MongoDB, microservices, Docker, Ocelot.

1. Εισαγωγή

Τα βασικά στοιχεία της διπλωματικής εργασίας, εστιάζονται σε τέσσερις ενότητες. Στην πρώτη ενότητα, αναλύεται το θεωρητικό υπόβαθρο. Στη δεύτερη, παρουσιάζεται το τεχνικό υπόβαθρο που απαιτείται για την κατανόηση τεχνολογιών που χρησιμοποιήθηκαν για την ανάπτυξη την πλατφόρμας. Στη τρίτη, παρουσιάζεται η πλατφόρμα πληθοπορισμού που αναπτύχθηκε. Τέλος, στην τέταρτη ενότητα, γίνεται αποτίμηση της προσπάθειας να αναπτυχθεί μια τέτοια πλατφόρμα.

2. Θεωρητικό υπόβαθρο

2.1 Πληθοπορισμός

Ο όρος «crowdsourcing» προέρχεται από τις λέξεις crowd (πλήθος) και sourcing (εξωτερική ανάθεση εργασιών). Ο Αμερικανός δημοσιογράφος Jeff Howe εισήγαγε το 2006 για πρώτη φορά τον όρο. Σύμφωνα με τον Howe, πληθοπορισμός είναι όταν μία εταιρεία ή ένα ίδρυμα αναθέτει εξωτερικά σε ένα απροσδιόριστο αριθμό ατόμων καθήκοντα που παραδοσιακά εκτελούνταν από υπάλληλο. Η εξωτερική ανάθεση γίνεται συνήθως μέσω διαδικτυακής ανοικτής πρόσκλησης. Ο Howe θεωρεί πως το χάσμα μεταξύ επαγγελματιών και ερασιτεχνών στον τομέα της τεχνολογίας έχει μικρύνει και οι εταιρείες μπορούν να αξιοποιούν τις ικανότητες του κοινού.

Σύμφωνα με τους Estellés-Arolas και González Ladrón-de-Guevara ο πληθοπορισμός είναι συλλογική διαδικτυακή δραστηριότητα ομάδων εθελοντών η οποία οργανώνεται μετά από ανοικτή πρόσκληση και επιβλέπεται από ένα ίδρυμα ή ένα μη κερδοσκοπικό οργανισμό ή και εταιρεία.

Τα βασικά συστατικά του πληθοπορισμού:

- ο οργανισμός που αναθέτει το έργο
- το πλήθος που είναι πρόθυμο να εκτελέσει το έργο
- το διαδικτυακό περιβάλλον, όπου επιτρέπει την υλοποίηση του έργου και την αλληλεπίδραση μεταξύ του πλήθους και του οργανισμού
- αμοιβαίο όφελος για τον οργανισμό και την κοινότητα

Το πρόβλημα που καλούνται να λύσουν οι εθελοντές μπορεί να είναι αυξημένης πολυπλοκότητας και δυσκολίας. Οι εθελοντές πρέπει να αφιερώσουν προσωπικό χρόνο, χρήματα, γνώσεις και εμπειρίες.

Το όφελος μιας τέτοιας πράξης είναι αμοιβαίο και για τις δύο πλευρές. Ο πληθοποριστής χρησιμοποιεί προς όφελος του, τις προσπάθειες των εθελοντών. Ενώ οι εθελοντές λαμβάνουν συνήθως κάτι από τα παρακάτω: χρηματικές απολαβές, κοινωνική αναγνώριση ή και προσωπική ικανοποίηση.

Τα πλεονεκτήματα που προκύπτουν από την εφαρμογή πληθοπορισμού:

- Μείωση του κόστους και του χρόνου που χρειάζεται για να ολοκληρωθεί ένα έργο.
- Ο πληθοποριστής έχει πρόσβαση σε ένα μεγαλύτερο εύρος δεξιοτήτων από αυτό που διαθέτει.
- Οι οργανισμοί ακούγοντας το πλήθος αποκτούν γνώση των επιθυμιών των πελατών τους.
- Το πλήθος αναπτύσσει νέες δεξιότητες.
- Το πλήθος λαμβάνει την ανταμοιβή με βάση τα αποτελέσματα.

Τα μειονεκτήματα που προκύπτουν από την εφαρμογή πληθοπορισμού:

- Χαμηλής ποιότητας αποτελέσματα.
- Χαμηλές χρηματικές απολαβές και έλλειψη δέσμευσης για το πλήθος.
- Οι οργανισμοί συνεισφέρουν λιγότερα κεφάλαια
- Ανταγωνισμός και ανεργία σε διάφορα επαγγέλματα

Ενώ ο όρος πληθοπορισμός διαδόθηκε διαδικτυακά για να περιγράψει δραστηριότητες που βασίζονται στο Διαδίκτυο, υπάρχουν αρκετά παραδείγματα έργων, ανά τα χρόνια, που μπορούν να περιγραφούν ως crowdsourcing. Για παράδειγμα, το 594 π.Χ. ο Σόλων απαιτεί από όλους τους πολίτες να ορκίζονται να τηρήσουν τους νόμους του, οι οποίοι, μεταξύ άλλων, ενισχύουν την ένταξη των πολιτών και τη συμμετοχή τους στη διακυβέρνηση της Αρχαίας Αθήνας. Ενώ ο Βασιλιάς Λουδοβίκος ΙΣΤ’ της Γαλλίας, το 1783 προσέφερε ένα βραβείο (2400 λίρες) στο άτομο που θα παρασκεύαζε «σόδα τέφρας» αποσυνθέτοντας το θαλασσινό αλάτι με την απλούστερη και πιο οικονομική μέθοδο.

Το φαινόμενο του πληθοπορισμού πήρε την πλήρη του μορφή όπως τη γνωρίζουμε σήμερα, με την ανάπτυξη του διαδικτύου. Το διαδίκτυο θεωρείται ένα καλό μέσο ικανό να φιλοξενήσει πλατφόρμες πληθοπορισμού (web-based platforms).

Ο κύριος παράγοντας που καθιστά ένα μοντέλο πληθοπορισμού επιτυχημένο είναι η ευθυγράμμιση των κινήτρων του πλήθους. Στην ουσία, η ευθυγράμμιση των κινήτρων του πλήθους είναι ο βαθμός στον οποίο μπορούν να συνδεθούν με τον μακροπρόθεσμο στόχο της πρωτοβουλίας, ενθαρρύνοντας έτσι την ευρύτερη συμμετοχή τους.

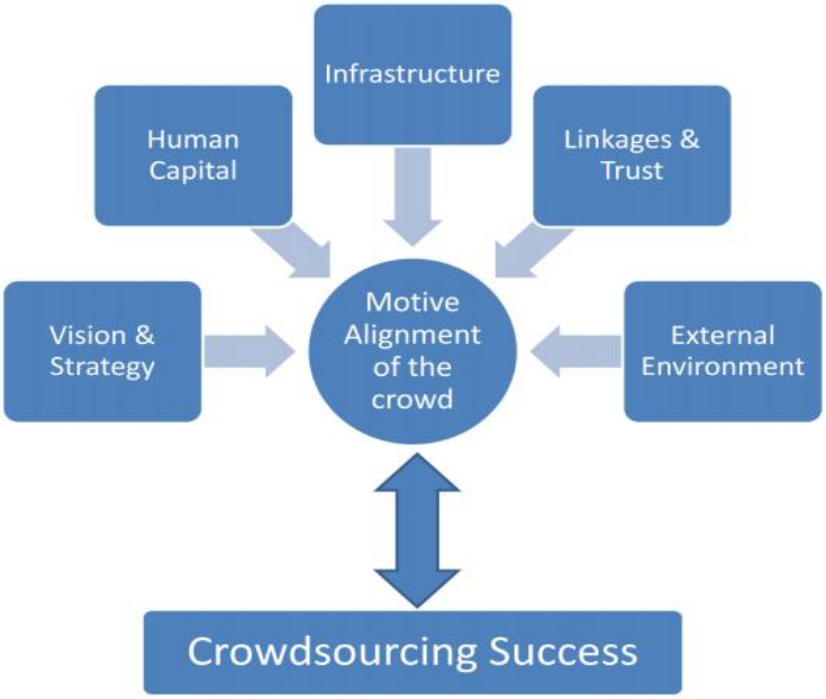
Δευτερεύοντες παράγοντες είναι το όραμα, η στρατηγική, το ανθρώπινο κεφάλαιο, οι υποδομές, οι διασυνδέσεις και η εμπιστοσύνη, και το εξωτερικό περιβάλλον.

Όλες οι πρωτοβουλίες πληθοπορισμού πρέπει να εισέρχονται στην αγορά με ένα καλά καθορισμένο σύνολο ιδανικών και στόχων. Το όραμα της πρωτοβουλίας πρέπει να κάνει το πλήθος να αντιλαμβάνεται την πρωτοβουλία ως πολύτιμη και καλοπροαίρετη. Το ιδανικό είναι να παρέχονται κίνητρα στο πλήθος για τη συμμετοχή τους.

Οι δεξιότητες που διαθέτει το πλήθος, θεωρείται ένας εξίσου σημαντικός παράγοντας για την επιτυχία του πληθοπορισμού. Ο Erran Carmel κάνει αναφορά στις ικανότητες και στις δεξιότητες που διαθέτει συλλογικά το πλήθος ως Ανθρώπινο Κεφάλαιο (Human Capital).

Απαιτούνται κατάλληλες πηγές κεφαλαίων για την ανάπτυξη των υποδομών. Η χρηματοδότηση αυτή μπορεί να προέρχεται είτε από εγχώριες είτε από ξένες επενδύσεις. Η ύπαρξη τεχνολογικής υποδομής ενισχύει αισθητά τη συμμετοχή του πλήθους.

Η ελαχιστοποίηση του κόστους και η αύξηση της εμπιστοσύνης, επιτυγχάνεται μέσω σωστής ανάπτυξης των διασυνδέσεων. Σύμφωνα με τον Erran Carmel, η ύπαρξη σωστών διασυνδέσεων είναι απαραίτητη στα ανθρωπιστικά μοντέλα καθώς η μεταφορά γνώσης γίνεται ευκολότερη. Χρειάζονται αξιόπιστες πληροφορίες από το πλήθος, για να λειτουργήσει σωστά μια εφαρμογή πληθοπορισμού. Αυτές οι πληροφορίες καθιστούν απαραίτητη την ανάπτυξη του παράγοντα της εμπιστοσύνης.



Εικόνα 1: Παράγοντες διασφάλισης επιτυχούς crowdsourcing μοντέλου.

Η επιτυχία μιας πρωτοβουλίας πληθοπορισμού μπορεί να επιτευχθεί μέσω ενός οικονομικού περιβάλλοντος που προάγει μια κουλτούρα επιχειρηματικότητας.

Ο παρακάτω πίνακας απεικονίζει την επίδραση που έχουν οι δευτερεύοντες παράγοντες στους καθοριστικούς παράγοντες.

	Προσδοκία απόδοσης	Προσδοκία προσπάθειας	Κοινωνική επιρροή	Συνθήκες διευκόλυνσης
Όραμα & στρατηγική	✔		✔	
Ανθρώπινο δυναμικό	✔	✔		
Συνδέσεις & εμπιστοσύνη	✔		✔	
Υποδομές		✔		✔
Εξωτερικό περιβάλλον		✔	✔	✔

Πίνακας 1: Πίνακας Προσδιορισμού Επιδράσεων

2.2 Πλατφόρμες πληθοπορισμού

Από τις πιο γνωστές πλατφόρμες πληθοπορισμού είναι αυτή της Amazon, που ονομάζεται Amazon Mechanical Turk (MTurk). Η πλατφόρμα αυτή διευκολύνει άτομα και επιχειρήσεις να αναθέτουν διαδικασίες σε κατανεμημένο εργατικό δυναμικό που μπορεί να εκτελέσει αυτές τις εργασίες ουσιαστικά. Η πλατφόρμα περιλαμβάνει έργα όπως συμμετοχή σε έρευνα (μέσω ερωτηματολογίων), απλής επικύρωσης δεδομένων κ.α..

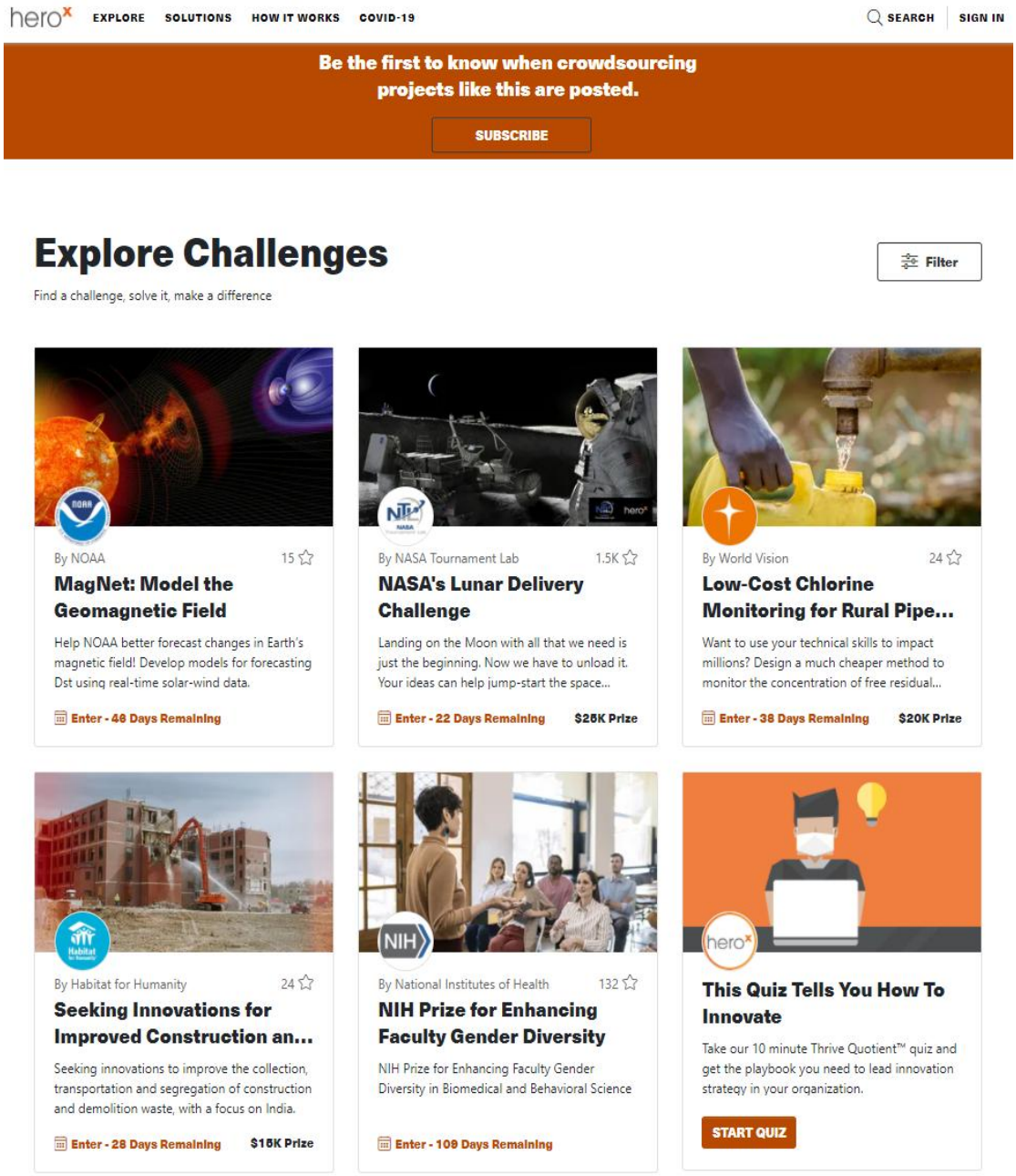
Για τις επιχειρήσεις παρέχεται η δυνατότητα εκμετάλλευσης της συλλογικής νοημοσύνης, των δεξιοτήτων και των γνώσεων σε ένα παγκόσμιο εργατικό δυναμικό. Με αυτό τον τρόπο επιτυγχάνεται η βελτιστοποίηση των επιχειρηματικών διαδικασιών, ξ αύξηση της συλλογής και ανάλυσης δεδομένων και η επιτάχυνση της ανάπτυξης της μηχανικής μάθησης.

Το πλήθος έχει τη δυνατότητα να επιλέξει μέσα από μια πληθώρα έργων. Σε μερικές περιπτώσεις μπορεί να απαιτηθεί από το χρήστη να περάσει κάποιο τεστ για να αποδείξει τις γνώσεις του. Η αμοιβή κάθε έργου είναι φανερή σε όλους τους χρήστες της πλατφόρμας.

Μια άλλη δημοφιλής πλατφόρμα πληθοπορισμού είναι η herox. Η herox παρέχει μια πλατφόρμα δύο «όψεων» όπου οι πελάτες της σχεδιάζουν προκλήσεις (challenges) γύρω από προβλήματα που χρειάζονται επίλυση και το πλήθος που εργάζεται για να βρει μια λύση με σκοπό να κερδίσει ένα βραβείο.

Χαρακτηριστικό παράδειγμα είναι όταν η NASA δημιούργησε μέσα από την πλατφόρμα της herox το challenge «Watts on the Moon» και ζήτησε από το πλήθος να υποβάλλει ιδέες για τη διανομή, αποθήκευση και διαχείριση ενέργειας στο φεγγάρι. Το challenge ολοκληρώνεται σε δύο στάδια. Το πρώτο στάδιο αφορούσε την εξεύρεση λύσης σε ζητήματα όπως η μεταφορά ενέργειας από εργοστάσιο σε μια φορητή πλατφόρμα, μια μονάδα επεξεργασίας νερού και σε μία μονάδα παραγωγής οξυγόνου μέσα σε κρατήρα. Σε αυτή τη φάση προσφέρθηκαν έπαθλα έως 500.000 δολάρια. Στη δεύτερη φάση έπρεπε να δημιουργήσουν πρωτότυπα που θα προσομοιώνουν κάποιες από τις παραπάνω λειτουργίες σε εγκαταστάσεις της NASA. Τα χρηματικά έπαθλα αυτής της φάσης ήταν 4,5 εκατομμύρια δολάρια.

Άλλες δημοφιλείς πλατφόρμες πληθοπορισμού είναι η Appen, η Clickworkers και η Microworkers.



Εικόνα 2: Η πλατφόρμα πληθοπορισμού της herox.

2.3 Πληθοπορισμός στην Ανώτατη Εκπαίδευση

Σύμφωνα με τους Llorente και Morant, η εφαρμογή τεχνικών πληθοπορισμού σε ιδρύματα τριτοβάθμιας εκπαίδευσης βελτιώνει την απόδοση των φοιτητών. Φοιτητές που συμμετέχουν σε έργα πληθοπορισμού ενισχύουν τις δεξιότητες τους. Το υλικό των διαλέξεων συγκεντρώνεται και μοιράζεται αποτελεσματικά. Επίσης, η οικονομική κατάσταση των φοιτητών βελτιώνονται καθώς τα δίδακτρα τους καλύπτονται μέσω πληθοχρηματοδότησης (crowdfunding).

Οι Llorente και Morant αναλύουν τις εξής τέσσερις τεχνικές πληθοπορισμού που παίζουν σημαντικό ρόλο στην απόδοση των φοιτητών: το crowdteaching, το crowdlearnig, το crowdtuition και το crowdfunding.

Το crowdteaching βελτιστοποιεί τη διαδικασία της διάλεξης και επιτρέπει την ανταλλαγή σημειώσεων. Οι καθηγητές αναζητούν εκπαιδευτικό υλικό με υψηλή ποιότητα περιεχόμενο που θα μπορεί να διδαχθεί αποτελεσματικά. Υπάρχουν ειδικές πλατφόρμες (π.χ. Renaissance), που επιτρέπουν την ανταλλαγή πόρων ανάμεσα σε καθηγητές. Η τεχνική αυτή είναι από τις πιο διαδεδομένες σήμερα. Οι καθηγητές εξοικονομούν αρκετό χρόνο κατά την προετοιμασία των μαθημάτων. Θα πρέπει όμως να δίνεται ιδιαίτερη προσοχή στη χρήση αξιόπιστων πηγών. Οι φοιτητές έχουν πρόσβαση σε καλύτερα προγράμματα σπουδών. Οι Llorente και Morant θεωρούν την τεχνική αυτή την πιο αποτελεσματική.

Το crowdlearnig έγινε ευρέως γνωστό με τη δημιουργία εκπαιδευτικών πλατφορμών όπως το SkillShare. Με το crowdlearning, ομάδες φοιτητών παίρνουν μέρος σε συνεργατικά έργα όπου διδάσκουν αποτελεσματικά ο ένας τον άλλον υπό την επίβλεψη των καθηγητών. Οι φοιτητές προτείνουν δεξιότητες που κατέχουν ήδη και μπορούν να βοηθήσουν στην επίτευξη των στόχων του έργου, με σκοπό τη συγκέντρωση διαφόρων δεξιοτήτων. Στο τέλος του έργου, ο φοιτητής αποκτά δεξιότητες άλλων φοιτητών. Αυτή η προσέγγιση έχει εφαρμοστεί με επιτυχία σε ιδρύματα της Αμερικής και της Ευρώπης.

Σε σχέση με την απόκτηση χρηματοδότησης, με το crowdtuition, φοιτητές με υψηλές επιδόσεις μπορούν να καλύψουν μέρος των διδάκτρων τους. Το crowdtuition αφορά κυρίως τη λήψη φοιτητικού δανείου από μια εταιρεία με ευνοϊκούς όρους. Η αποπληρωμή του δανείου γίνεται μετά την αποφοίτηση.

Οι κυβερνήσεις καλύπτουν το κόστος του υλικού των δημόσιων ιδρυμάτων αλλά η διαδικασία απόκτησης εργαστηριακού εξοπλισμού είναι πιο δύσκολη και περίπλοκη. Με το crowdfunding μπορεί ένα ίδρυμα να βελτιώσει τις εγκαταστάσεις και τις υπηρεσίες που προσφέρει. Το crowdfunding μπορεί επίσης να εφαρμοστεί για να υποστηρίξει τη διαμονή των μαθητών στο εξωτερικό.

Πολλά ιδρύματα εφαρμόζουν πληθοπορισμό με σκοπό να πάρουν τη γνώμη των φοιτητών για θέματα που αφορούν το ίδρυμα. Για παράδειγμα, το Πανεπιστήμιο της Κολούμπια πρότεινε στους φοιτητές να υποβάλλουν προτάσεις για να βελτιωθεί η πανεπιστημιούπολη.

Το όφελος από την εφαρμογή του πληθοπορισμού στην ανώτατη εκπαίδευση είναι αμοιβαίο. Οι φοιτητές μπορούν να αλληλεπιδράσουν μεταξύ τους κατά τη διαδικασία της μάθησης και να λάβουν οικονομική υποστήριξη. Οι φοιτητές μπορούν να έχουν πρόσβαση στο καλύτερο εκπαιδευτικό υλικό και μπορούν να βελτιώσουν την μαθησιακή τους αποδοτικότητα. Ενώ τα ιδρύματα έχουν τη δυνατότητα να αυξήσουν τον προϋπολογισμό τους και να κάνουν πιο αποτελεσματική χρήση του χρόνου για μάθηση.

3 Τεχνικό υπόβαθρο

3.1 ASP.NET Core

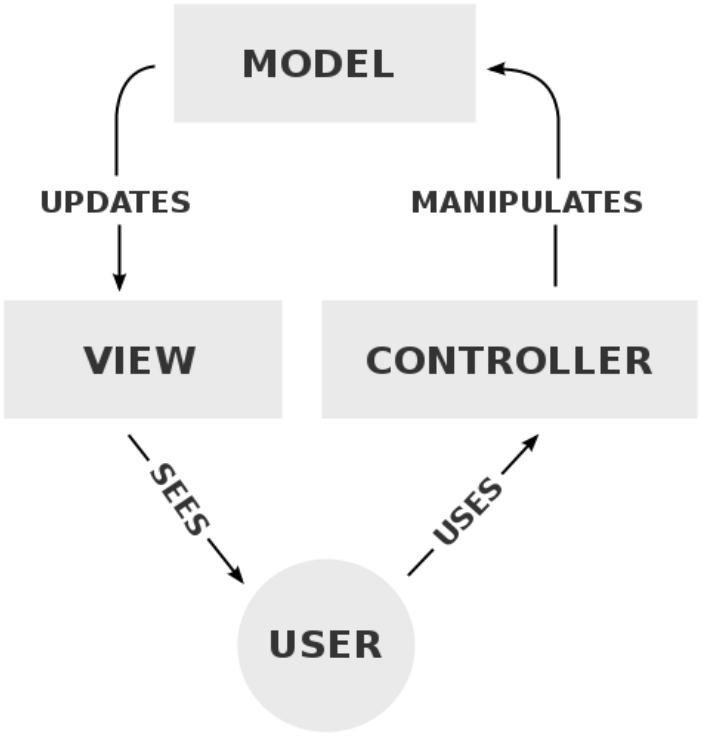
Το ASP.NET Core είναι ένα cross-platform framework της Microsoft που κάνει εύκολη τη δημιουργία δυναμικών ιστοσελίδων στο διαδίκτυο. Επίσης μας δίνει τη δυνατότητα να φτιάχνουμε διαδικτυακές εφαρμογές και υπηρεσίες. Οι εφαρμογές αυτές μπορούν να αναπτυχθούν και να λειτουργήσουν στα περισσότερα λειτουργικά συστήματα.

Η ASP.NET Core υποστηρίζει το μοντέλο Model–View–Controller (MVC). Αυτό το μοντέλο αρχιτεκτονικής λογισμικού χρησιμοποιείται για την δημιουργία περιβαλλόντων αλληλεπίδρασης χρήστη. Στο μοντέλο αυτό η εφαρμογή χωρίζεται σε τρία μέρη ώστε να υπάρχει διαχωρισμός της πληροφορίας που είναι αποθηκευμένη από την πληροφορία που παρουσιάζεται στο χρήστη.

Ο **Controller** μπορεί να στέλνει εντολές στο μοντέλο (Model) και να ενημερώνει την κατάσταση του μοντέλου. Μπορεί επίσης να στέλνει εντολές ώστε να γίνει η αντίστοιχη αναπαράσταση των δεδομένων του μοντέλου μέσω του View.

Το **Model** ενημερώνει τις αντίστοιχες αναπαραστάσεις Views και τους Controllers όταν υπάρχει αλλαγή στα δεδομένα. Αυτή η ενημέρωση επιτρέπει στα Views να ενημερώνουν την γραφική απεικόνιση.

Το **View** παρουσιάζει την πληροφορία του Model στο χρήστη.



Εικόνα 3: Η αρχιτεκτονική του MVC.

3.2 SQL Server

Η SQL Server είναι μια σχεσιακή βάση δεδομένων, η οποία αναπτύχθηκε από την Microsoft. Κυκλοφόρησε σε συνεργασία με τη Sybase για πρώτη φορά το 1989 και χρησιμοποιεί ως κύριες γλώσσες την T-SQL και την ANSI SQL. Συχνά χρησιμοποιείται σε συνδυασμό με το εργαλείο Microsoft SQL Server Management Studio.

Εικόνα 4: Το λογότυπο της SQL Server.

3.3 MongoDB

Η MongoDB είναι μία document oriented database, αυτό σημαίνει ότι η αποθήκευση των εγγραφών σε ένα πίνακα της βάσης μπορεί να έχει τη μορφή δεδομένων JSON. Δεν είναι απαραίτητο ο πίνακας να έχει συγκεκριμένο αριθμό πεδίων όπως στις σχεσιακές βάσεις. Δηλαδή, το schema της βάσης δεν είναι απαραίτητα σταθερό.

Εικόνα 5: Το λογότυπο της MongoDB.

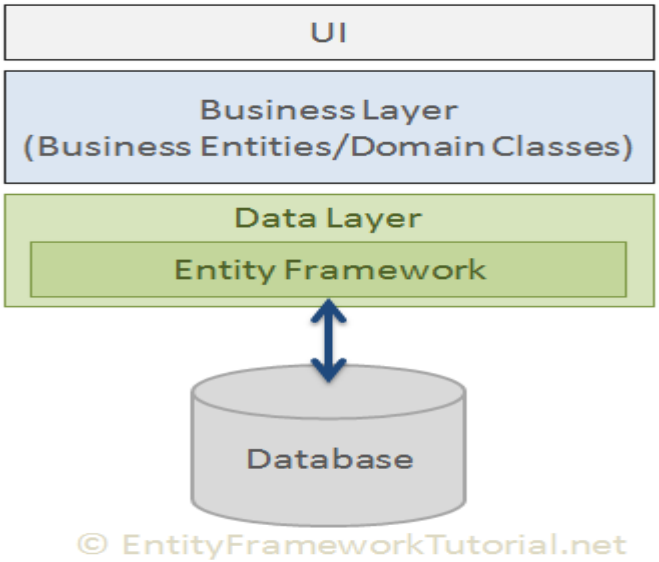
Τα έγγραφα (documents) είναι ευέλικτα. Κάθε έγγραφο μπορεί να αποθηκεύσει δεδομένα με διαφορετικά χαρακτηριστικά από άλλα έγγραφα. Επίσης τα έγγραφα έχουν την ίδια φιλοσοφία με τα JSON Objects, που θα χρησιμοποιήσουμε στην εργασία.

Τα έγγραφα κάνουν τις εφαρμογές πιο γρήγορες. Τα δεδομένα μιας οντότητας είναι αποθηκευμένα σε ένα μόνο έγγραφο, αντί να εξαπλώνονται σε πολλούς σχεσιακούς πίνακες. Η βάση δεδομένων χρειάζεται μόνο να διαβάζει και να γράφει σε ένα μόνο μέρος. Η κατοχή όλων των δεδομένων μιας οντότητας σε ένα μέρος διευκολύνει επίσης τους προγραμματιστές να κατανοήσουν και να βελτιστοποιήσουν την απόδοση του ερωτήματος.

Επιπλέον τα ερωτήματα που γίνονται για τη συλλογή των δεδομένων από τη βάση είναι δυναμικά και δεν απαιτούν συγκεκριμένους δείκτες, αλλά γράφονται σύμφωνα με την MongoDB document-based query language.

3.4 Entity Framework Core

Το Entity Framework Core είναι ένα object-relational mapping (ORM) framework, που μας επιτρέπει να εργαστούμε με σχεσιακά και μη σχεσιακά δεδομένα χρησιμοποιώντας domain-specific αντικείμενα.



Εικόνα 6: Η αρχιτεκτονική του Entity Framework Core.

Με το Entity Framework Core μπορούμε να:

- δημιουργήσουμε και να διαχειριστούμε μία Βάση Δεδομένων χωρίς να γράψουμε κώδικα SQL (Code-First προσέγγιση).
- παράξουμε κώδικα C# με βάση το μοντέλο οντοτήτων της Βάσης Δεδομένων. (Database-First προσέγγιση).

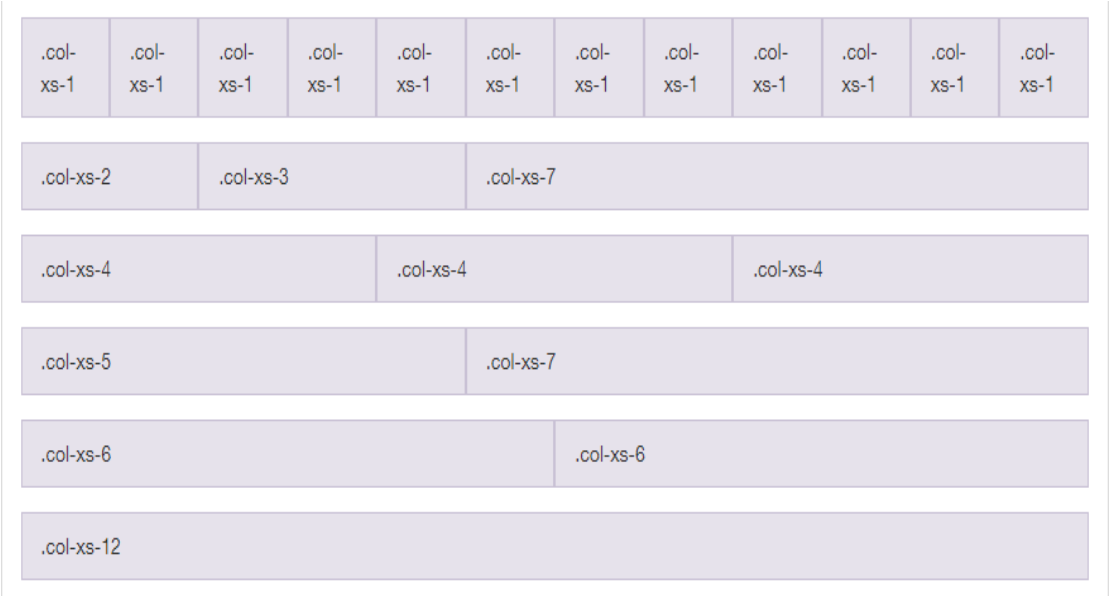
Το Entity Framework Core παρέχει οδηγούς (drivers) για τις περισσότερες σχεσιακές δεδομένων (π.χ. MySQL, SQL Server) και μη σχεσιακές βάσεις δεδομένων (π.χ. MongoDB).

3.5 Bootstrap

Το Bootstrap είναι ένα πανίσχυρο ανοικτού κώδικα front-end framework για ταχύτερη και ευκολότερη ανάπτυξη ιστοσελίδων και διαδικτυακών εφαρμογών. Είναι μια συλλογή εργαλείων ανοιχτού κώδικα που περιέχει HTML και CSS για τις μορφές τυπογραφίας, κουμπιά πλοήγησης και άλλων στοιχείων του περιβάλλοντος, καθώς και προαιρετικές επεκτάσεις JavaScript (κυρίως τη βιβλιοθήκη jQuery) για να αυξάνει το επίπεδο διαδραστικότητας των ιστοσελίδων.

Το Bootstrap (έκδοση 4) υπάρχει προεγκατεστημένο, όταν δημιουργούμε ένα ASP.NET WEB Application Project.

Το σύστημα πλέγματος (grid system) που διαθέτει το Bootstrap, κάνει εύκολο τον ανταποκρίσιμο σχεδιασμό (responsive design).



Εικόνα 7: Το σύστημα πλέγματος του Bootstrap.

3.6 jQuery

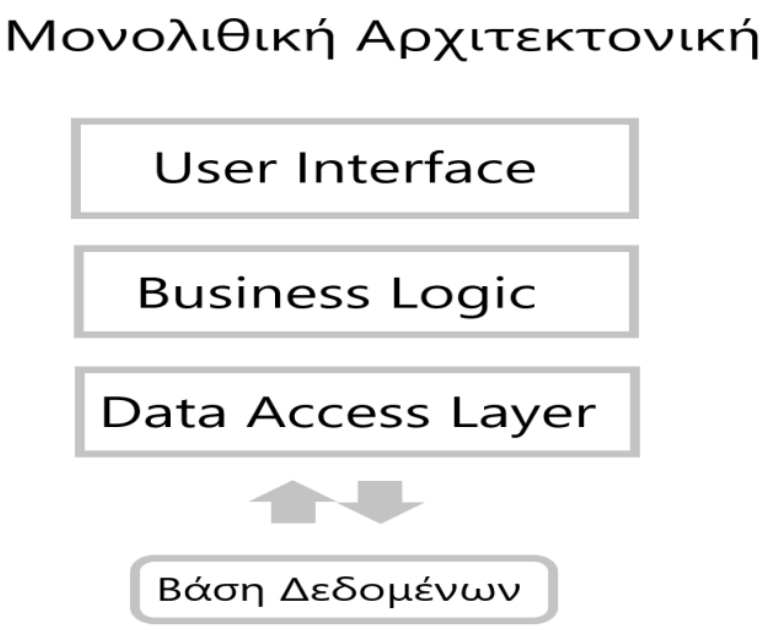
Η jQuery είναι μια βιβλιοθήκη της JavaScript. Σκοπός της είναι η απλοποίηση των σεναρίων (scripts). Γράφουμε, δηλαδή, σενάρια με πολύ λιγότερες γραμμές κώδικα από ότι με JavaScript. Η jQuery υπάρχει προεγκατεστημένη όταν δημιουργούμε ένα ASP.NET Core Web Application Project.

Εικόνα 8: Το λογότυπο του jQuery.

3.7 Microservices

Παραδοσιακά, οι προγραμματιστές αντιμετωπίζουν τις εφαρμογές που θέλουν να αναπτύξουν ως μια μονάδα. Η μονολιθική αρχιτεκτονική (Monolithic Architecture), όπως

ονομάζεται αυτή η προσέγγιση, αποτελείται συνήθως από τρεις στρώσεις (layers): το User Interface, το Business Logic και το Data Access Layer.



Εικόνα 9: Τα layers της μονολιθικής αρχιτεκτονικής.

Τα πλεονεκτήματα αυτής της αρχιτεκτονικής είναι:

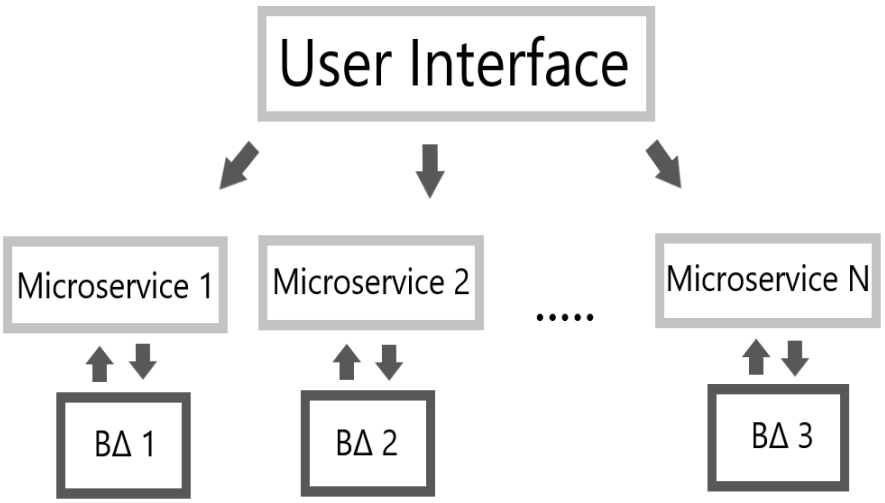
- εύκολη ανάπτυξη
- εύκολη επεκτασιμότητα
- εύκολος έλεγχος

Παρόλα αυτά, αυτή η αρχιτεκτονική έχει και αρκετά μειονεκτήματα. Μεγάλες εφαρμογές που έχουν υιοθετήσει αυτή την αρχιτεκτονική, είναι δύσκολες στη διαχείριση και στη συντήρηση. Ακόμα και η κατανόηση του κώδικα από τρίτους δεν είναι εύκολη. Σφάλματα μπορεί να επηρεάσουν την λειτουργία ολόκληρης της εφαρμογής. Γι' αυτό, για κάθε αλλαγή και βελτίωση που γίνεται στην εφαρμογή πρέπει να γίνεται έλεγχος.

Τα microservices (μικρο-υπηρεσίες) αποτελούν μια νέα τάση στην ανάπτυξη λογισμικού. Τα τελευταία χρόνια όλο και περισσότερες εταιρίες επιλέγουν αυτή την προσέγγιση κατά την ανάπτυξη των εφαρμογών τους.

Τα microservices είναι μια αρχιτεκτονική κατά την οποία φτιάχνουμε ένα σύνολο από μικρά services (υπηρεσίες). Αυτά τα services κάνουν λίγες και συγκεκριμένες δουλειές το καθένα. Τα microservices πρέπει να είναι όσο το δυνατόν πιο ανεξάρτητα μεταξύ τους. Μπορούν να χρησιμοποιούν ακόμα και ξεχωριστές Βάσεις Δεδομένων.

Για παράδειγμα, σε ένα ηλεκτρονικό κατάστημα, η υπηρεσία καταχώρησης παραγγελίας είναι ανεξάρτητη από την υπηρεσία που προτείνει προϊόντα στον επισκέπτη.



Εικόνα 10: Η αρχιτεκτονική των microservices.

Η επικοινωνία μεταξύ των microservices γίνεται με ουρές μηνυμάτων (message queues).

Τα πλεονεκτήματα αυτής της εφαρμογής:

- η ανάπτυξη της εφαρμογής γίνεται ακόμα πιο εύκολα και γρήγορα.
- κάθε microservice μπορεί να αναπτυχθεί από διαφορετικές ομάδες προγραμματιστών.
- κάθε microservice μπορεί να αναπτυχθεί με διαφορετικές τεχνολογίες.
- κάθε microservice μπορεί να γίνει επεκταθεί ξεχωριστά.
- τα microservices μιας εφαρμογής, είναι εφικτό να μπορούν να χρησιμοποιηθούν και σε άλλες εφαρμογές.

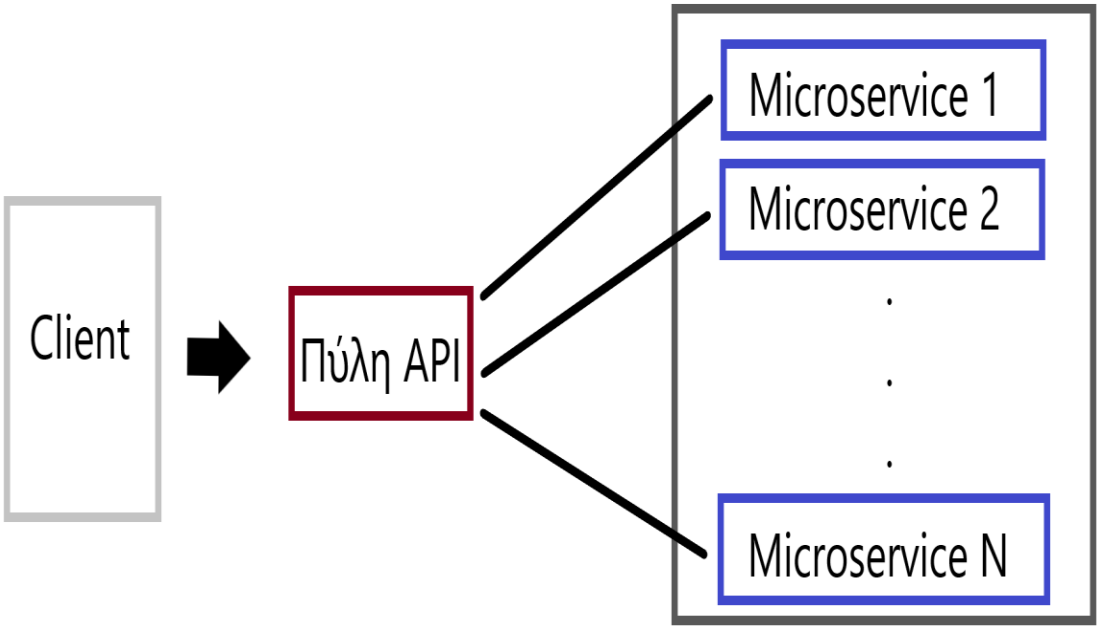
Αυτή η αρχιτεκτονική μπορεί να ακούγεται ως μια ιδανική λύση για τα περισσότερα προβλήματα σχεδίασης λογισμικού, αλλά έχει και αρκετά μειονεκτήματα.

Όταν υπάρχουν αρκετά microservices σε μια εφαρμογή, αυξάνεται ο βαθμός πολυπλοκότητας της. Απαιτείται καλή συνεργασία μεταξύ των ομάδων προγραμματιστών. Επίσης μπορεί να παρατηρηθούν καθυστερήσεις στην εφαρμογή, λόγω των πολλών κλήσεων (ανάμεσα στα microservices). Με λίγα λόγια, η χρήση των microservices σε μεγάλες εφαρμογές απαιτεί συμβιβασμούς.

3.8 Πύλη API

Κάθε microservice φιλοξενείται στο δικό του σύνδεσμο. Έτσι ο χρήστης (client) πρέπει να γνωρίζει όλους αυτούς τους συνδέσμους. Αυτό το πρόβλημα, λύνεται με τις πύλες API (gateway API). Το gateway API, ενοποιεί όλους αυτούς τους συνδέσμους σε ένα κοινό

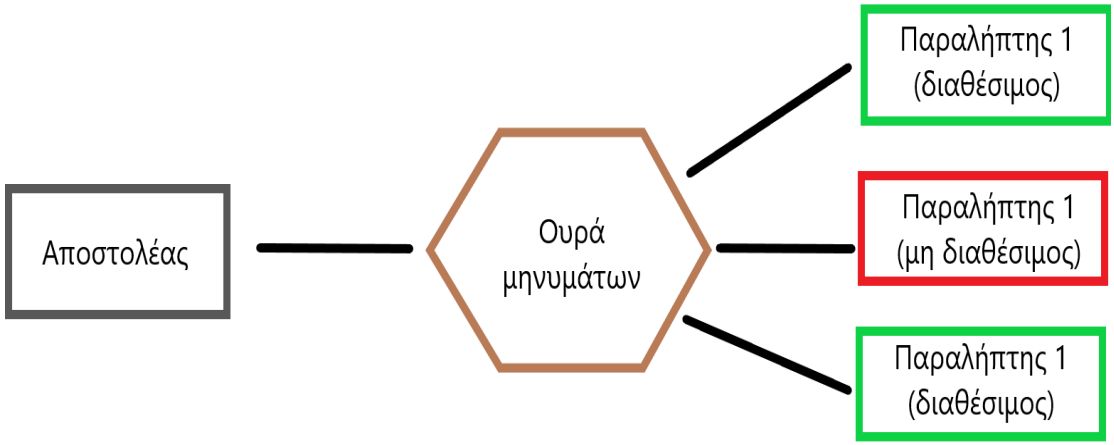
σύνδεσμο. Προσθέτουμε δηλαδή, ένα νέο στάδιο ανάμεσα στις εφαρμογές και στα microservices.



Εικόνα 11: Gateway API - Ένα ενδιάμεσο στάδιο.

3.9 Κανάλι μηνυμάτων

Το κανάλι μηνυμάτων (service bus) χρησιμοποιείται για την επικοινωνία μεταξύ των microservices. Η επικοινωνία επιτυγχάνεται με τη χρήση ουρών (queues). Η χρήση των ουρών μας δίνει εγγύηση ότι τα μηνύματα θα παραδοθούν. Ακόμα και αν ένα microservice δεν είναι διαθέσιμο εκείνη τη στιγμή, το μήνυμα θα παραμείνει στην ουρά και θα σταλεί μόλις η επανέλθει η κατάσταση του. Με τη χρήση των ουρών, μπορούμε να στείλουμε μηνύματα σε πολλούς παραλήπτες.



Εικόνα 12: το κανάλι μηνυμάτων γίνεται με τη χρήση ουρών.

3.10 Docker

Η εικονοποίηση (virtualization) είναι η τεχνολογία που μετατρέπει τα φυσικά συστήματα σε ιδεατά (virtual machines). Έτσι, οι φυσικοί πόροι του συστήματος μας, γίνονται ένας ενιαίος που είναι διαθέσιμος σε πολλά εικονικά συστήματα.

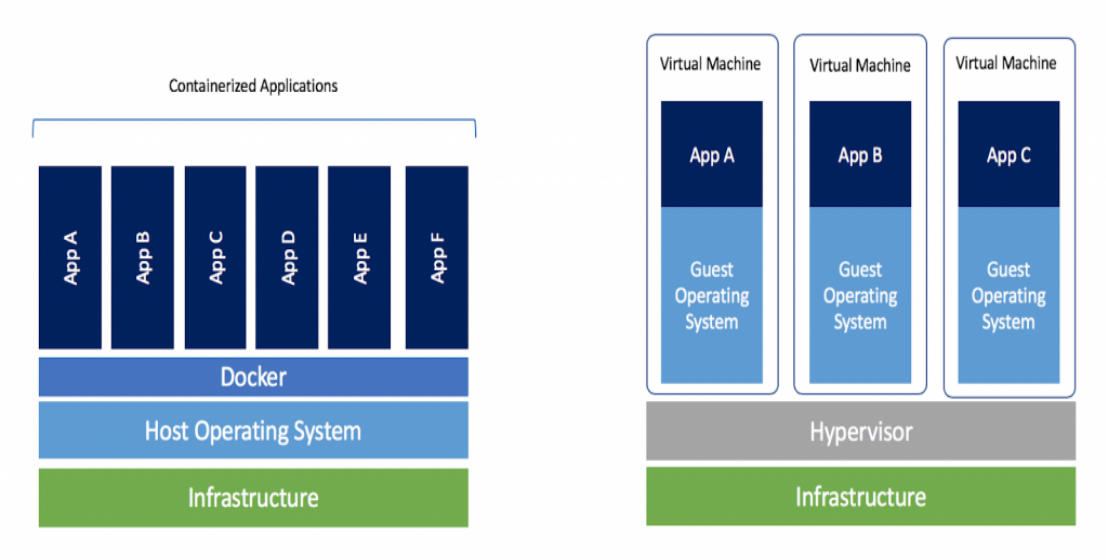
Η εικονοποίηση εφαρμογών χρησιμοποιείται για να διαχωρίσει τις εφαρμογές από φιλοξενούμενο λειτουργικό σύστημα τους. Έτσι επιτυγχάνουμε χαμηλότερο κόστος και προσαρμοστικότητα των εφαρμογών σε διαφορετικά συστήματα.

Η εικονοποίηση χωρίζεται στις εξής βασικές κατηγορίες:

- Εικονοποίηση υλικού (Hypervisor based virtualization)
- Εικονοποίηση λογισμικού (Container based virtualization)
- Εικονοποίηση δικτύου (Network virtualization)
- Εικονοποίηση Desktop (Desktop virtualization)

Η εικονοποίηση λογισμικού είναι η τοποθέτηση μιας εφαρμογής σε ένα πακέτο (container). Μέσα στο container υπάρχουν όλες οι εξαρτήσεις (dependencies) της εφαρμογής. Το container είναι ένα απομονωμένο και αυτόνομο περιβάλλον. Έτσι τα containers μπορούν να λειτουργούν, χωρίς να χρησιμοποιούν όλους του πόρους μιας εικονικής μηχανής.

Εικόνα 13: Το λογότυπο του Docker

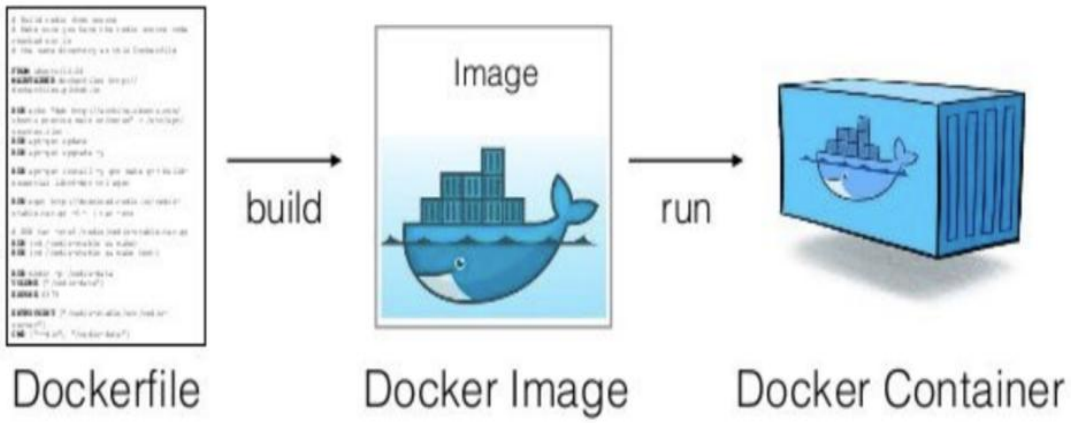


Εικόνα 14: Containers και Εικονικές μηχανές.

Η χρήση των containers είναι ιδιαίτερα διαδεδομένη σε εφαρμογές που χρησιμοποιούν την αρχιτεκτονική των microservices. Θεωρείται καλή πρακτική να απομονώνουμε τα microservices σε containers.

Το Docker είναι μια πλατφόρμα λογισμικού που υλοποιεί την εικονοποίηση λογισμικού. Το Docker μας επιτρέπει να δημιουργούμε και διαχειριζόμαστε containers.

Βασικό στοιχείο του Docker, είναι το Docker Image. Το Docker Image είναι ένα δυαδικό αρχείο που περιέχει όλο το λογισμικό και τη διαχείριση που χρειάζεται για να τρέξει μια εφαρμογή. Το container είναι ένα στιγμιότυπο (instance) ενός Docker Image.



Εικόνα 15: Τα βασικά στοιχεία του Docker.

Ένα άλλο βασικό στοιχείο του Docker, είναι το Dockerfile. Το Dockerfile είναι ένα αρχείο κειμένου (text file), που περιέχει οδηγίες για την δημιουργία ενός Docker Image.

3.11 Git

Ο έλεγχος εκδόσεων (version control) είναι ένα σύστημα το οποίο καταγράφει αλλαγές σε ένα αρχεία έτσι ώστε να είναι εφικτή η ανάκληση σε συγκεκριμένες εκδόσεις.

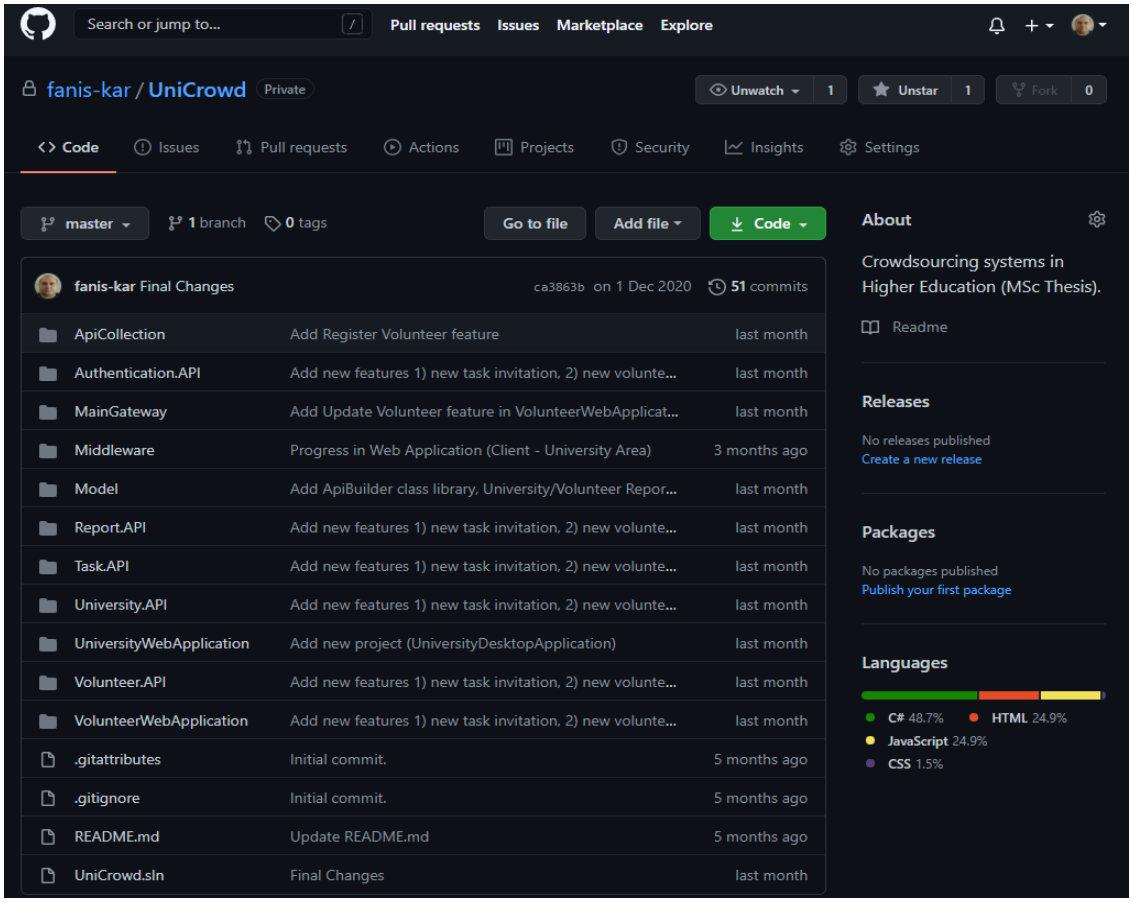
Το Git είναι ένα σύστημα ελέγχου εκδόσεων (Version Control System - VCS) που δίνει έμφαση στην ταχύτητα και στην ακεραιότητα των δεδομένων καθώς και στην υποστήριξη κατανεμημένων μη γραμμικών ροών εργασιών.

Είναι το πιο διαδεδομένο σύστημα εκδόσεων για ανάπτυξη λογισμικού. Κάθε κατάλογος εργασίας του Git είναι ένα ολοκληρωμένο αποθετήριο λογισμικού που παρέχει πλήρες ιστορικό και δυνατότητες πλήρους παρακολούθησης της έκδοσης, ανεξάρτητα από την πρόσβαση δικτύου ή ενός κεντρικού διακομιστή (server).

Το Git επιτρέπει στους προγραμματιστές να:

- Διατηρούν πλήρη αντίγραφα του πηγαίου κώδικα

- Συγκρίνουν τις εκδόσεις
- Συγχωνεύουν τις αλλαγές από διάφορες εκδόσεις
- Παρακολουθούν τις αλλαγές
- Δίνουν την δυνατότητα να μοιράζεται εύκολα ο κώδικας μεταξύ της ομάδας
- Γυρνάνε εύκολα σε προηγούμενη έκδοση



Εικόνα 16: Στιγμιότυπο από το αποθετήριο της πλατφόρμας «UniCrowd» στο GitHub.

4 Πλατφόρμα πληθοπορισμού «UniCrowd»

Στα πλαίσια αυτής της εργασίας αναπτύχθηκε ειδική πλατφόρμα πληθοπορισμού για πανεπιστήμια. Η πλατφόρμα αποτελείται από δύο εφαρμογές:

A. διαχειριστική εφαρμογή πληθοποριστών (UniCrowd – university area)
B. εφαρμογή εθελοντών (UniCrowd – volunteer area)

Εικόνα 17: Τα λογότυπα των δύο εφαρμογών που αναπτύχθηκαν.

4.1 Προσδιορισμός ρόλων

Στην πλατφόρμα πληθοπορισμού έχουν πρόσβαση δύο ειδών χρηστών:

A. πανεπιστήμια (πληθοποριστές)
B. εθελοντές (πλήθος)

Κάθε ρόλος χρήστη μπορεί να εκτελέσει συγκεκριμένες λειτουργίες.

4.2 Ανάλυση απαιτήσεων

Η ανάλυση απαιτήσεων αποτελεί το βασικό πλάνο μας κατά τη σχεδίαση του λογισμικού και απαντάει στο βασικό ερώτημα «Τι θέλουμε να κάνει η Πλατφόρμα;».

4.2.1 Λειτουργικές απαιτήσεις

Για οποιαδήποτε ενέργεια απαιτείται σύνδεση στην πλατφόρμα. Οι πληθοποριστές πρέπει να έχουν τη δυνατότητα να δημιουργούν έργα (Tasks) και ομάδες εθελοντών (Groups). Είναι ευθύνη των πληθοποριστών η διαχείριση των Task και Groups. Οι εθελοντές πρέπει να μπορούν να στέλνουν προσκλήσεις για να συμμετάσχουν σε Tasks. Εθελοντές και πληθοποριστές πρέπει να έχουν τη δυνατότητα να συντάσσουν αναφορές (Reports) σχετικά με τα Tasks τους.

4.2.2 Μη λειτουργικές απαιτήσεις

Το σύστημα πληθοπορισμού πρέπει να:

- ✓ Λειτουργεί αδιάλειπτα
- ✓ Είναι εύκολο στη χρήστη
- ✓ Είναι προσβάσιμο σε AMEA (πρότυπο WCAG 2.0)

Τα στοιχεία των εθελοντών και των Tasks είναι διαθέσιμα για όλους τους εγγεγραμμένους χρήστες.

4.3 Περιγραφή λειτουργιών πανεπιστημίων

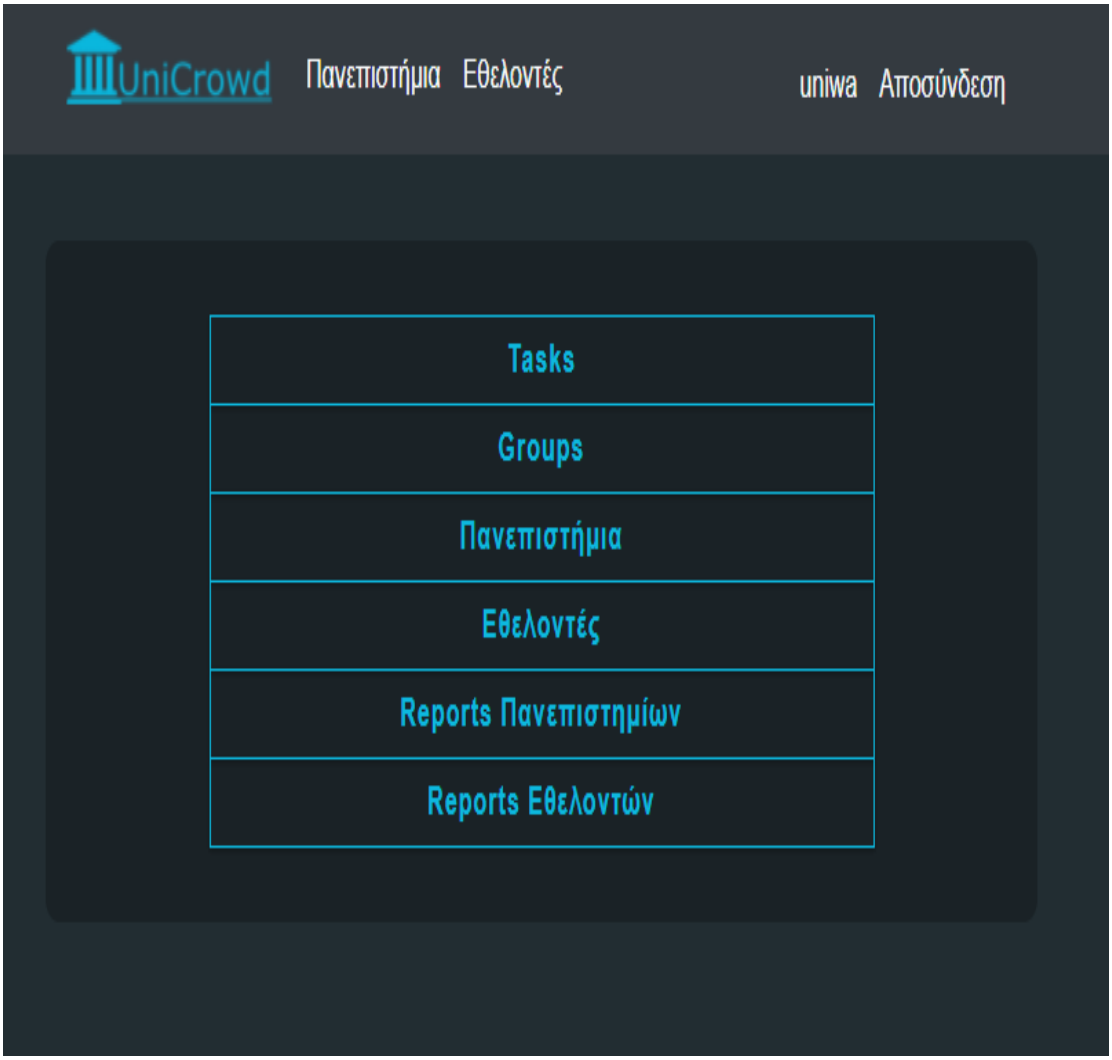
Ο χρήστης για να εισέλθει στην εφαρμογή (UniCrowd – university area), θα πρέπει να εισάγει το όνομα χρήστη και τον κωδικό πρόσβασης του.

Εικόνα 18: Στιγμιότυπο από την εφαρμογή «UniCrowd – university area» - Η σελίδα εισόδου.

Εφόσον τα στοιχεία είναι σωστά ο χρήστης, θα μεταφερθεί στην αρχική σελίδα της εφαρμογής.

Μέσα από το μενού, το πανεπιστήμιο μπορεί να εκτελέσει τις εξής λειτουργίες:

- Προβολή προσωπικού προφίλ
- Προβολή πανεπιστημίων, εθελοντών, tasks, groups και reports
- Δημιουργία και διαχείριση Tasks
- Διαχείριση Groups
- Σύνταξη Report

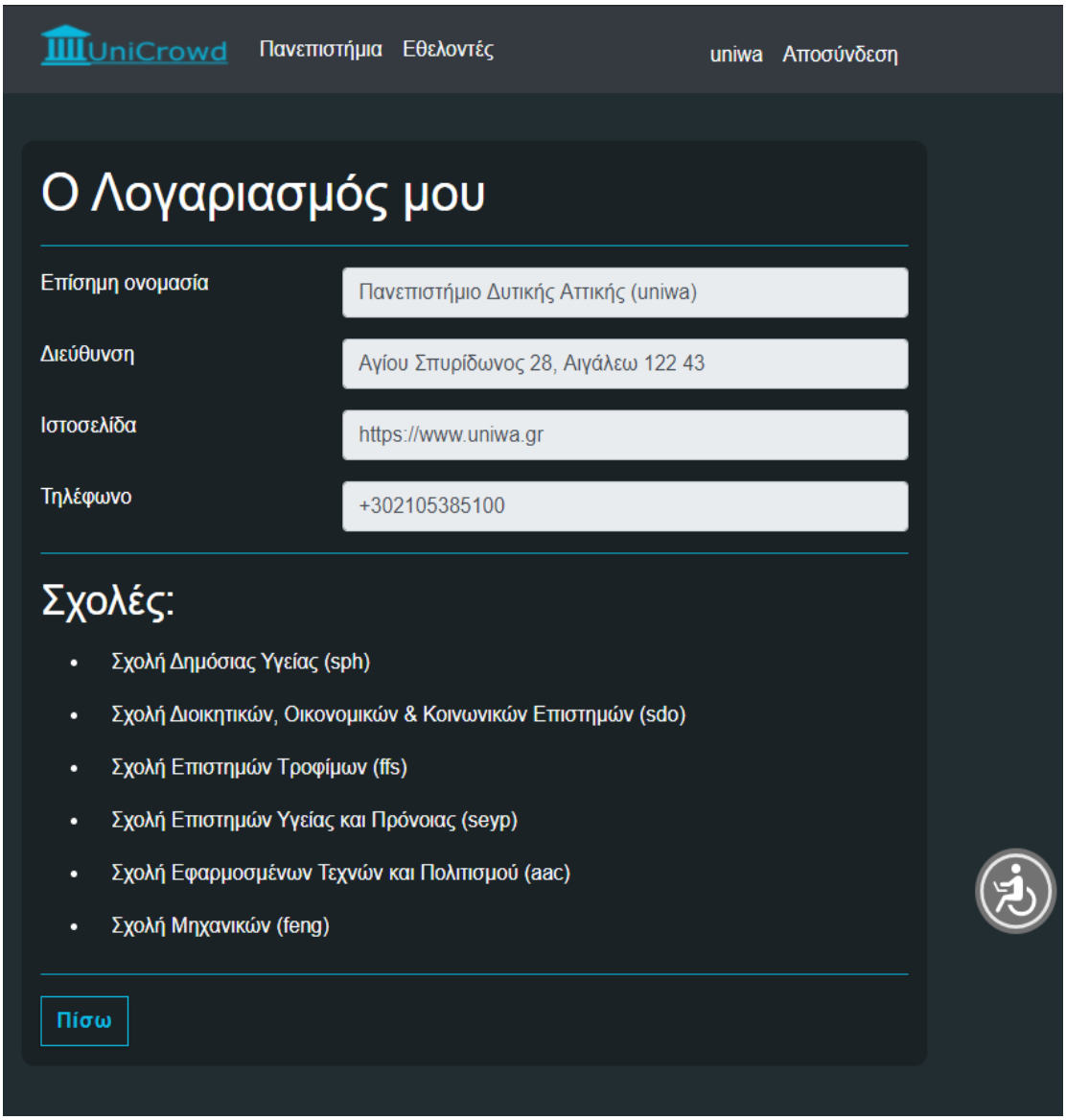


Εικόνα 19: Στιγμιότυπο από την εφαρμογή «UniCrowd – university area» - Το κύριο μενού της εφαρμογής .

4.3.1 Προβολή προσωπικού προφίλ

Ο χρήστης μπορεί να δει το προφίλ του πανεπιστημίου που διαχειρίζεται. Το προφίλ περιέχει τις παρακάτω πληροφορίες:

- Επίσημη ονομασία
- Διεύθυνση
- Ιστοσελίδα
- Τηλέφωνο
- Στοιχεία σχολών και τμημάτων

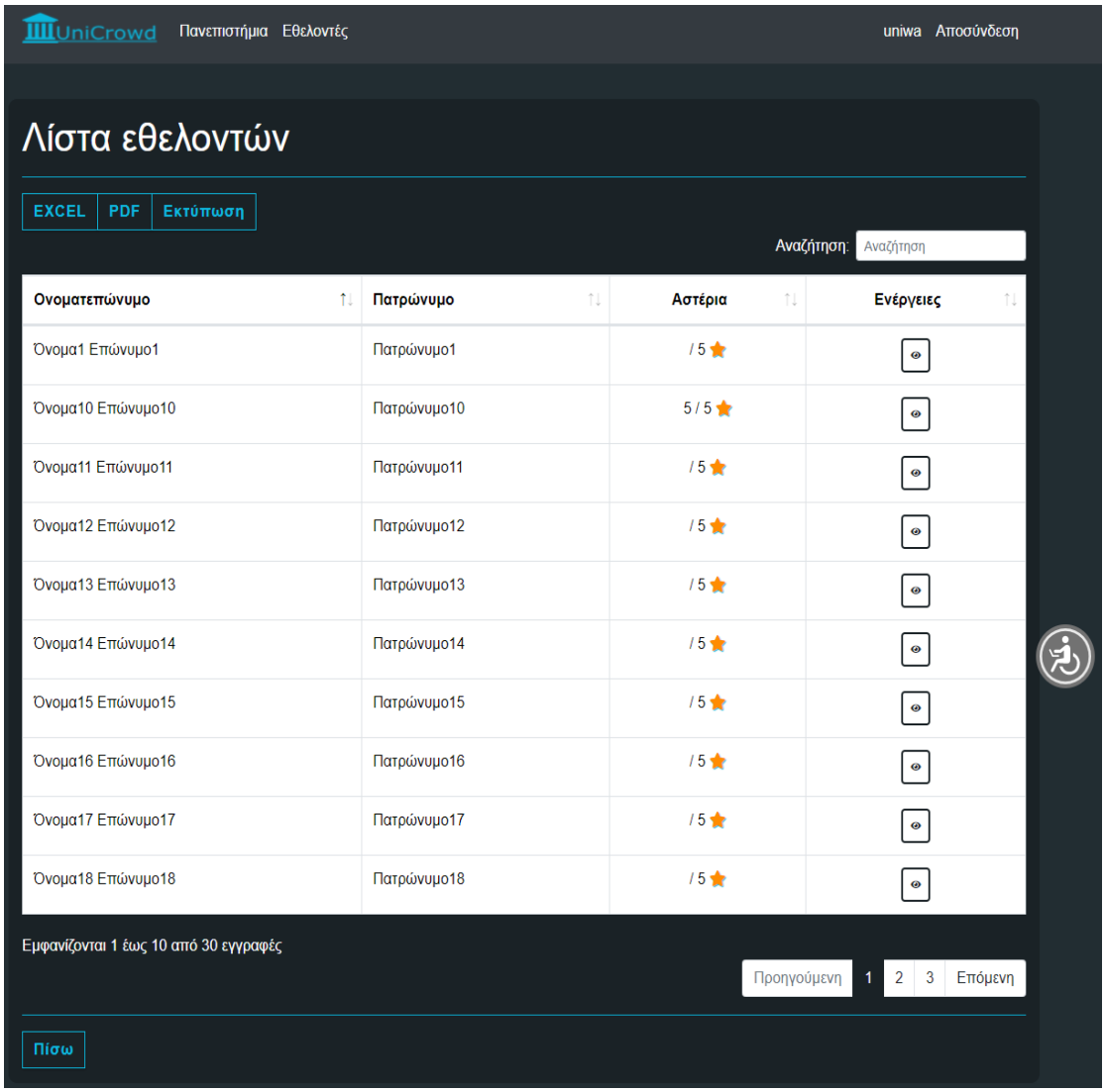


Εικόνα 20: Στιγμιότυπο από την εφαρμογή «UniCrowd – university area» - Προβολή προφίλ πανεπιστημίου.

4.3.2 Προβολή λιστών

Μέσα από το κύριο μενού, ο χρήστης μπορεί να δει λίστες από πανεπιστήμια, εθελοντές, Tasks, Groups και Reports που υπάρχουν στο σύστημα.

Οι παραπάνω πληροφορίες παρουσιάζονται με τη μορφή λιστών και μπορούν να εκτυπωθούν ή να εξαχθούν (υπό τη μορφή .xlsx ή .pdf αρχείου).

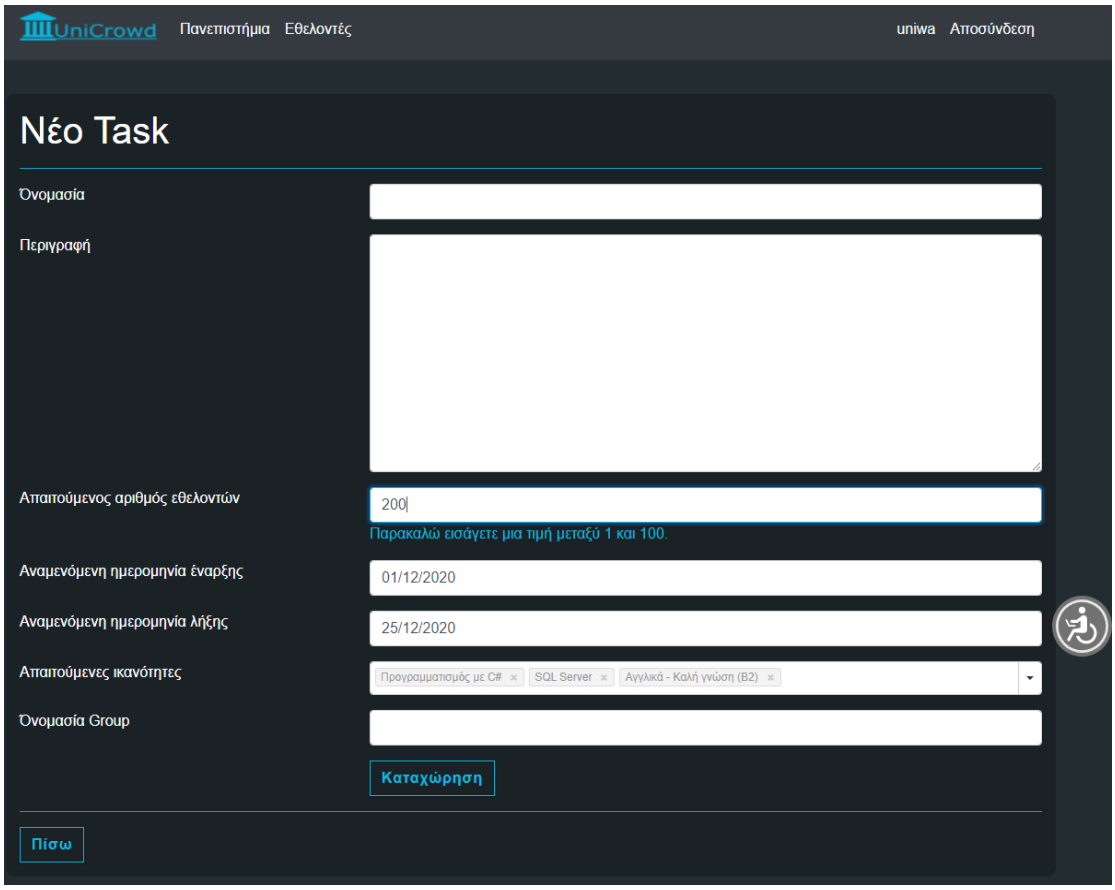


Εικόνα 21: Στιγμιότυπο από την εφαρμογή «UniCrowd – university area» - Προβολή λίστας εθελοντών.

4.3.3 Δημιουργία Task

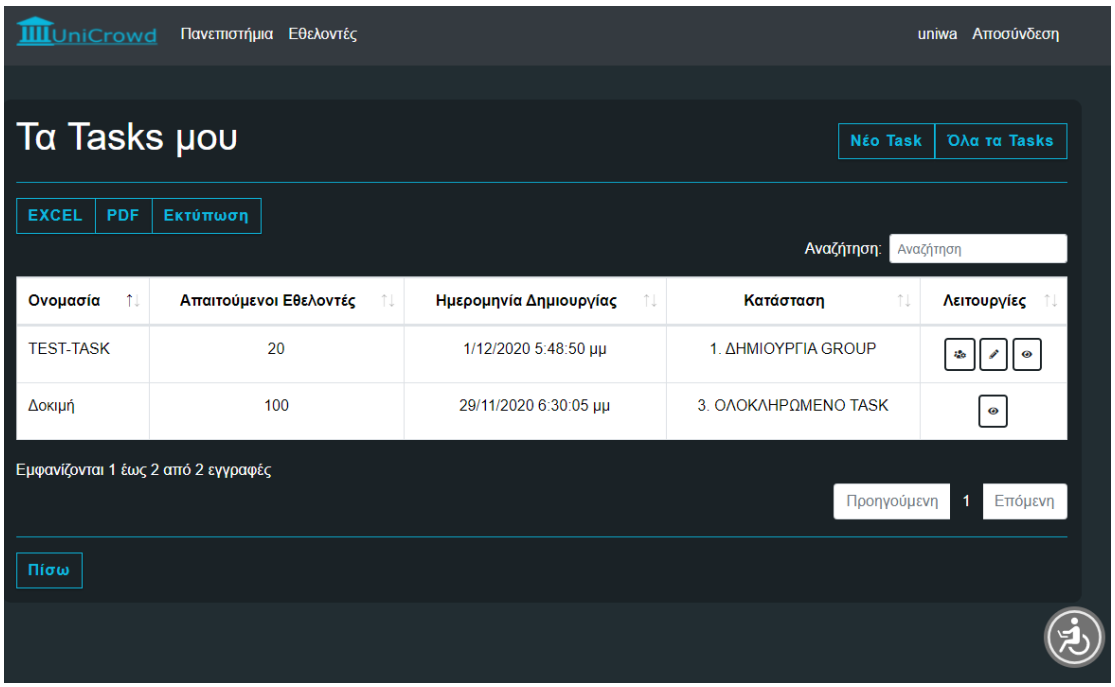
Ο χρήστης, ως πληθοποριστής, έχει την ικανότητα να δημιουργεί έργα (Tasks). Ο χρήστης πρέπει να εισάγει την εξής πληροφορία:

- Ονομασία έργου
- Περιγραφή έργου
- Απαιτούμενος αριθμός εθελοντών
- Αναμενόμενη ημερομηνία έναρξης
- Αναμενόμενη ημερομηνία λήξης
- Απαιτούμενες ικανότητες
- Ονομασία Group



Εικόνα 22: Στιγμιότυπο από την εφαρμογή «UniCrowd – university area» - Εισαγωγή νέου Task

Ο χρήστης μπορεί ανά πασά στιγμή να δει και να διαχειριστεί τα Tasks του.

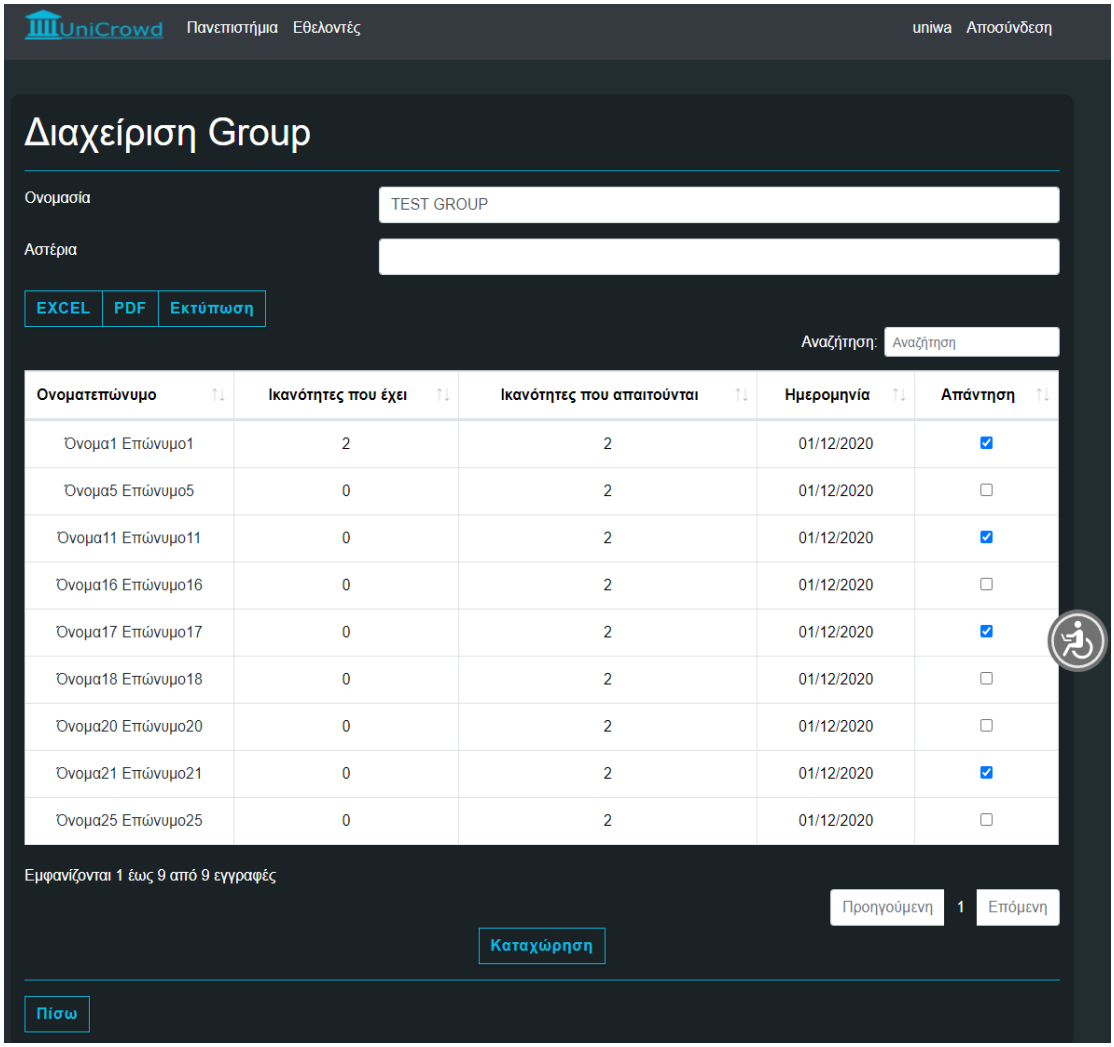


Εικόνα 23: Στιγμιότυπο από την εφαρμογή «UniCrowd – university area» - Διαχείριση των Tasks.

Μόλις δημιουργηθεί ένα Task, βρίσκεται στην κατάσταση «1. ΔΗΜΙΟΥΡΓΙΑ GROUP». Σε αυτό το στάδιο, ο χρήστης πρέπει να συνθέσει την ομάδα εθελοντών που θα αναλάβουν το Task.

4.3.4 Διαχείριση Group

Εφόσον το Task βρίσκεται στην κατάσταση «1. ΔΗΜΙΟΥΡΓΙΑ GROUP», ο χρήστης μπορεί να επιλέξει την ομάδα εθελοντών που θα αναλάβουν το Task. Ο χρήστης επιλέγει από μια λίστα τους εθελοντές που θεωρεί κατάλληλους. Στη λίστα εμφανίζονται εθελοντές που έχουν εκφράσει επιθυμία να συμμετάσχουν (έχουν απαντήσει στην ανοικτή πρόσκληση).



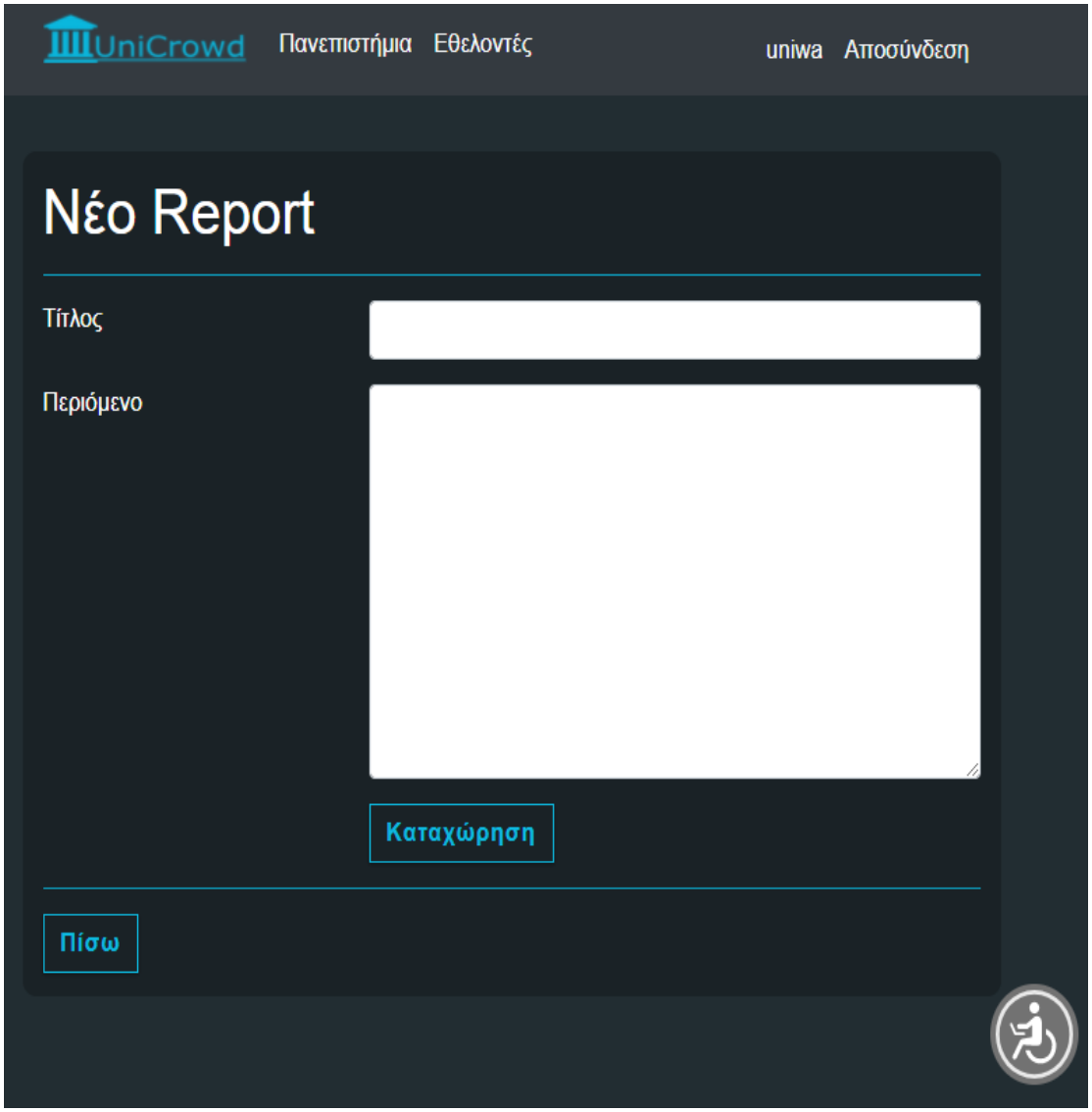
Εικόνα 24: Στιγμιότυπο από την εφαρμογή «UniCrowd – university area» - Διαχείριση των Groups.

Όταν το Task βρίσκεται σε κατάσταση «3. ΟΛΟΚΛΗΡΩΜΕΝΟ TASK», ο πληθοποριστής πρέπει να βαθμολογήσει το αντίστοιχο Group. Ταυτόχρονα, ενημερώνονται και οι βαθμολογίες των εθελοντών.

Η βαθμολογία κάθε εθελοντή προκύπτει από το μέσο όρο των βαθμολογιών που έχει λάθει για κάθε έργο.

4.3.5 Σύνταξη Report

Εφόσον το Task βρίσκεται στην κατάσταση «2. ΕΝΕΡΓΟ TASK», ο χρήστης μπορεί να συντάξει μια αναφορά (Report) που αφορά το Task. Τα Reports του πληθοποριστή είναι ορατά και στους εθελοντές.



Εικόνα 25: Στιγμιότυπο από την εφαρμογή «UniCrowd – university area» - Σύνταξη Report.

4.4 Περιγραφή λειτουργιών εθελοντών

Ο χρήστης για να εισέλθει στην εφαρμογή (UniCrowd – volunteer area), θα πρέπει να εισάγει το όνομα χρήστη (username) και τον κωδικό πρόσβασης του.

Εικόνα 26: Στιγμιότυπο από την εφαρμογή «UniCrowd – volunteer area» - Η σελίδα εισόδου.

Εφόσον τα στοιχεία είναι σωστά ο εθελοντής, θα μεταφερθεί στην αρχική σελίδα της εφαρμογής.

Εικόνα 27: Στιγμιότυπο από την εφαρμογή «UniCrowd – volunteer area» - Το κύριο μενού της εφαρμογής.

Μέσα από το μενού, ο εθελοντής μπορεί να εκτελέσει τις εξής λειτουργίες:

- Εγγραφή στο σύστημα
- Προβολή προσωπικού προφίλ
- Προβολή πανεπιστημίων, εθελοντών, tasks, groups και reports
- Αίτηση συμμετοχής σε Task
- Σύνταξη Report

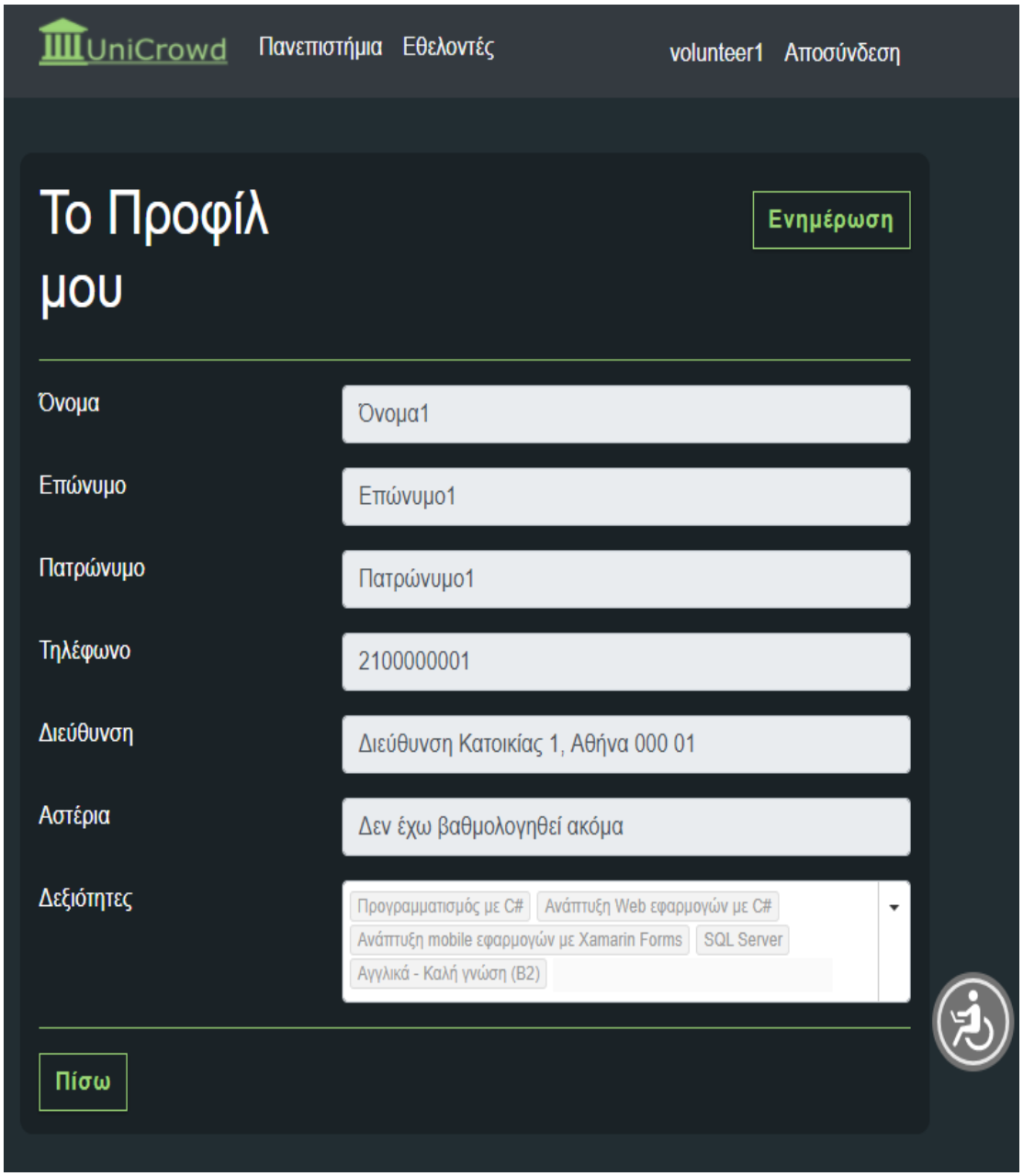
4.4.1 Εγγραφή στο σύστημα

Νέοι εθελοντές έχουν την ικανότητα να εγγραφτούν στο σύστημα. Ο νέος εθελοντής πρέπει να συμπληρώσει τα στοιχεία του: Όνομα χρήστη, email, κωδικός πρόσβασης, ονοματεπώνυμο, πατρώνυμο, τηλέφωνο και διεύθυνση.

Εικόνα 28: Στιγμιότυπο από την εφαρμογή «UniCrowd – volunteer area» - Εγγραφή νέου εθελοντή στο σύστημα.

4.4.2 Προβολή προσωπικού προφίλ

Ο χρήστης μπορεί να δει και να διαχειριστεί το προφίλ του. Το προφίλ του χρήστη, περιέχει τις πληροφορίες που έδωσε κατά την εγγραφή του στην εφαρμογή. Σε αυτό το σημείο, ο χρήστης μπορεί να καταχωρήσει τις δεξιότητες που κατέχει.

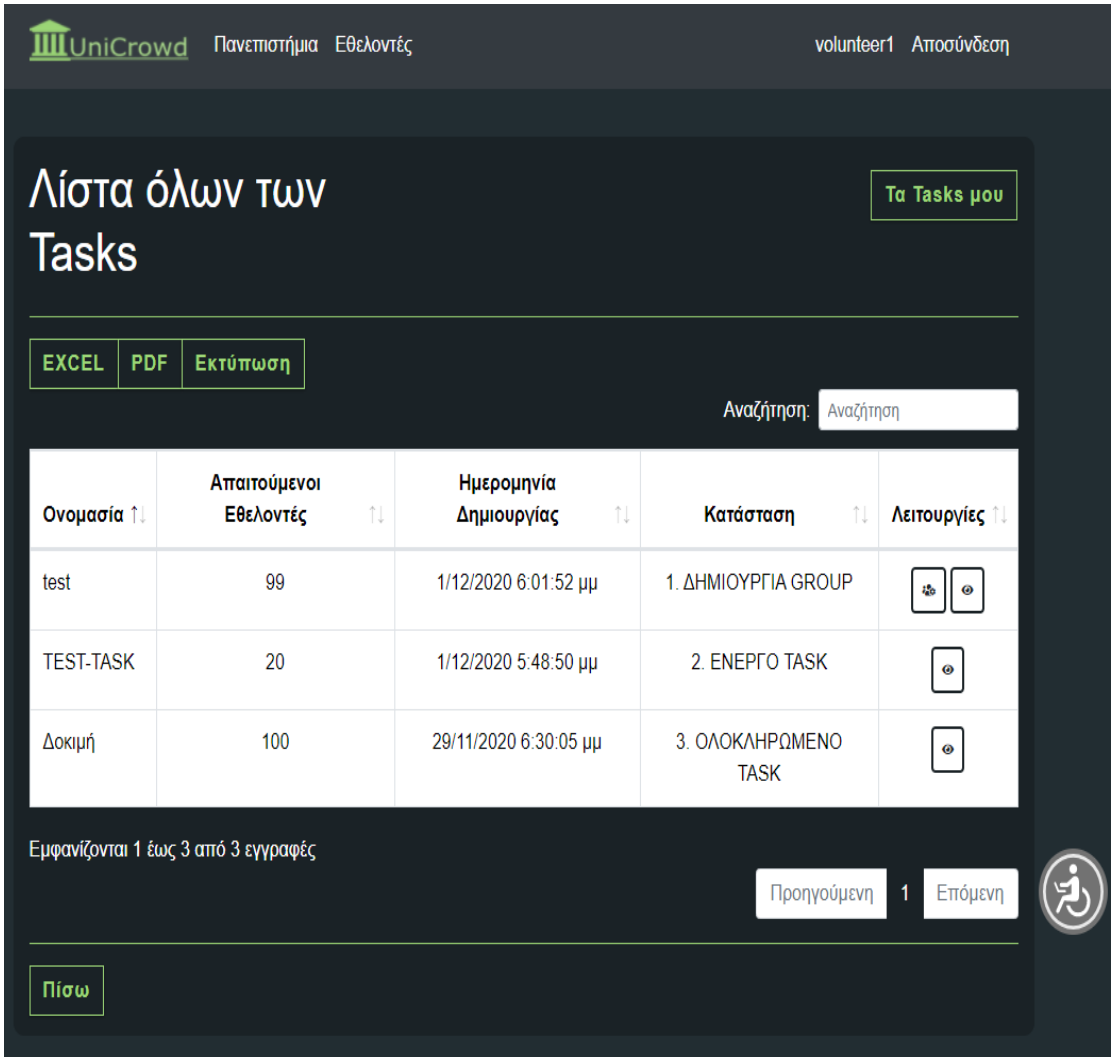


Εικόνα 29: Στιγμιότυπο από την εφαρμογή «UniCrowd – volunteer area» - Προβολή προφίλ.

4.4.3 Προβολή λιστών

Μέσα από το κύριο μενού, ο χρήστης μπορεί να δει λίστες από πανεπιστήμια, εθελοντές, Tasks, Groups και Reports που υπάρχουν στο σύστημα. Οι παραπάνω πληροφορίες

παρουσιάζονται με τη μορφή λιστών και μπορούν να εκτυπωθούν ή να εξαχθούν (υπό τη μορφή .xlsx ή .pdf αρχείου).

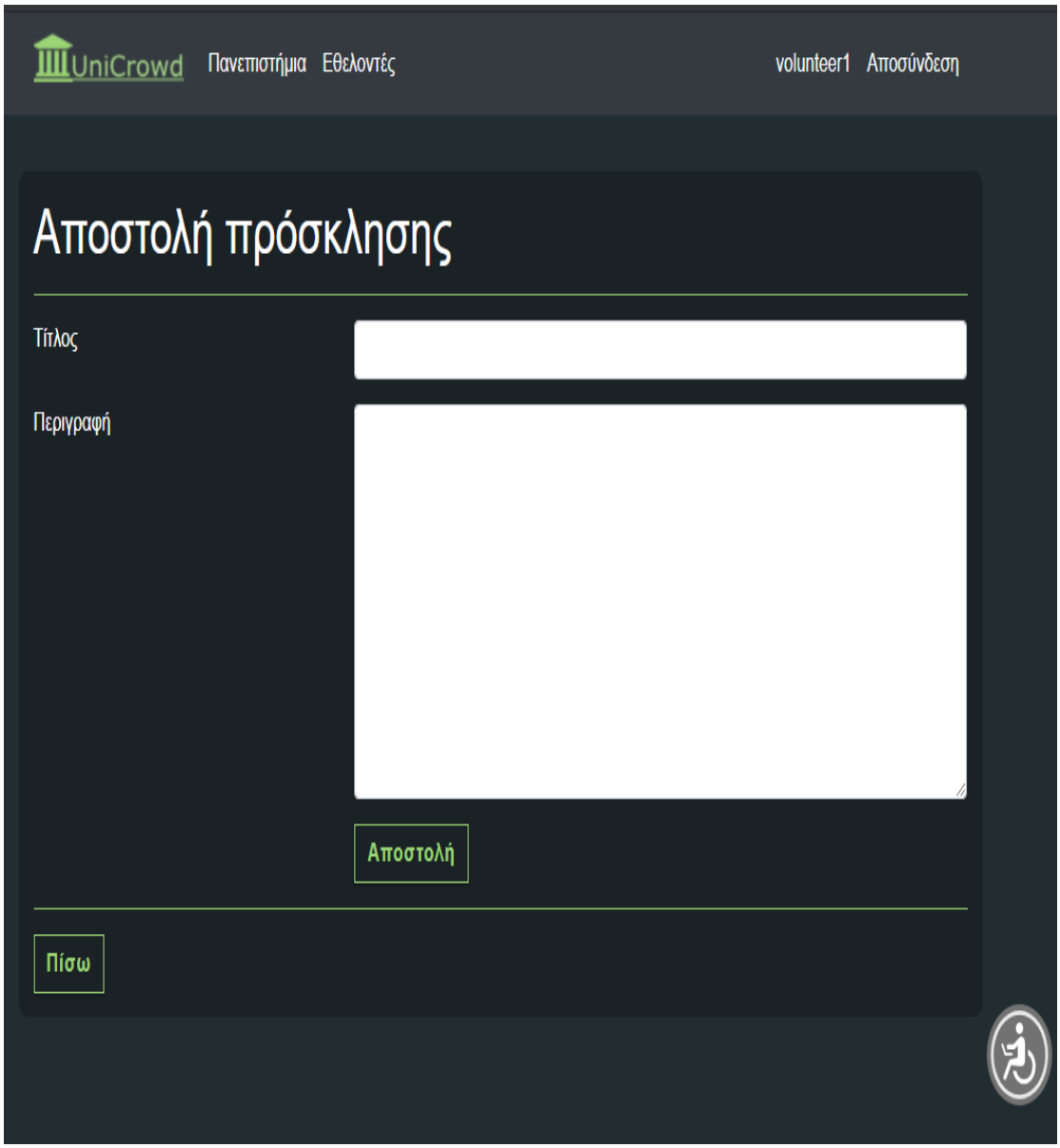


Εικόνα 30: Στιγμιότυπο από την εφαρμογή «UniCrowd – volunteer area» - Λίστα όλων των Tasks.

4.4.4 Αίτηση συμμετοχής σε Task

Εφόσον το Task βρίσκεται στην κατάσταση «1. ΔΗΜΙΟΥΡΓΙΑ GROUP», ο χρήστης εφόσον το επιθυμεί μπορεί να ζητήσει να συμμετάσχει στο αντίστοιχο Group.

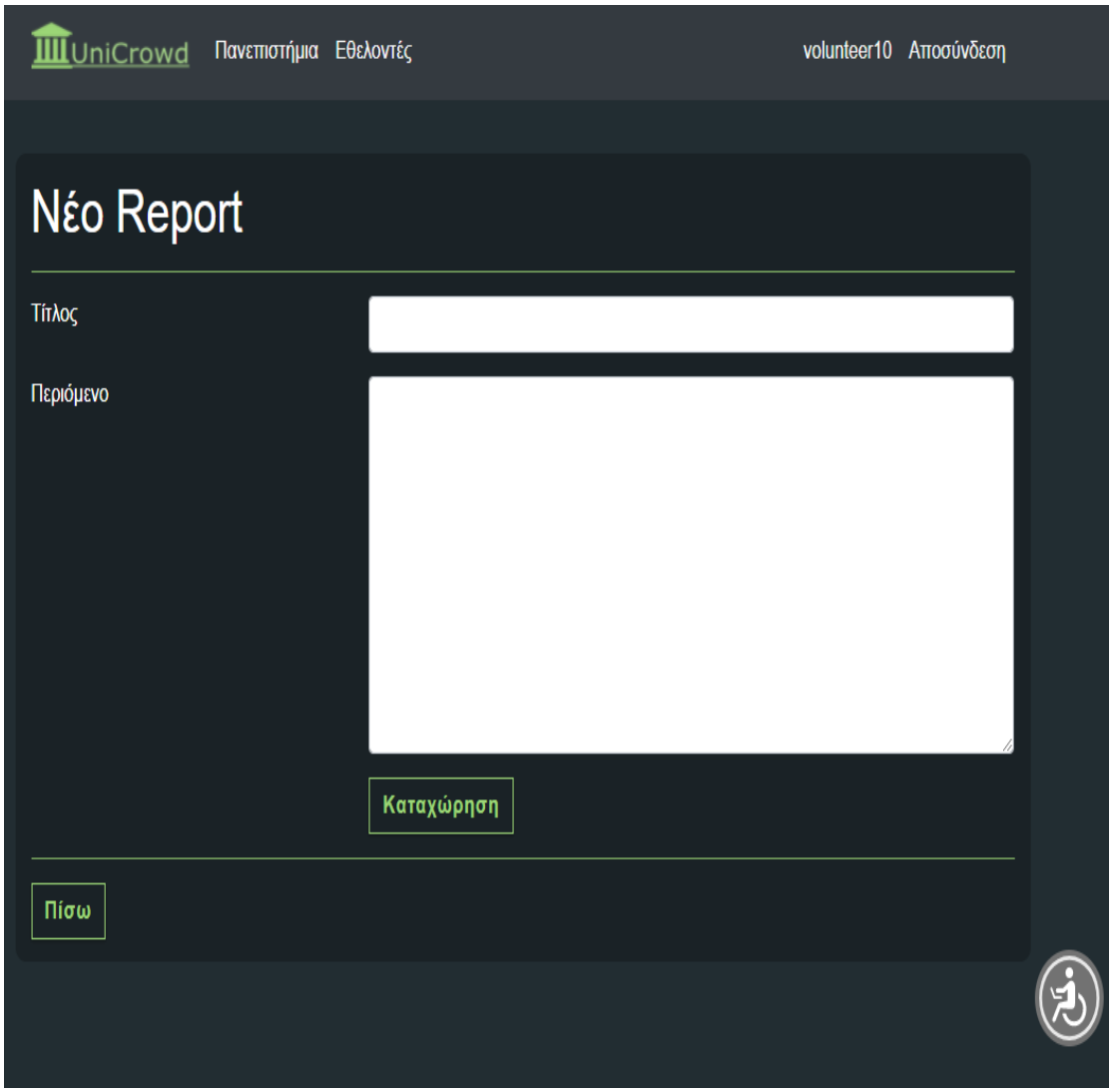
Ο πληθοποριστής όμως είναι αυτός που θα αποφασίσει ποιοι εθελοντές θα συμμετάσχουν στο έργο. Ο πληθοποριστής θα πρέπει να επιλέγει εθελοντές βάσει των ικανοτήτων και των βαθμολογιών τους.



Εικόνα 31: Στιγμιότυπο από την εφαρμογή «UniCrowd – volunteer area» - Αποστολή πρόσκλησης.

4.4.5 Σύνταξη Report

Εφόσον το Task βρίσκεται στην κατάσταση «2. ΕΝΕΡΓΟ TASK», οι εθελοντές που έχουν αναλάβει αυτό το έργο μπορούν να συντάξουν μια αναφορά (Report). Οι αναφορές αφορούν το Task. Τα Reports των εθελοντών είναι ορατά και στους πληθοποριστές.



Εικόνα 32: Στιγμιότυπο από την εφαρμογή «UniCrowd – volunteer area» - Σύνταξη report.

4.5 Τεχνολογίες

Παρακάτω αναφέρονται οι τεχνολογίες που χρησιμοποιήθηκαν για την υλοποίηση του συστήματος.

- ASP.NET Core 3.1 – Model View Controller pattern
- SQL Server (με χρήση του Entity Framework Core – code first)
- MongoDB (με χρήση του Entity Framework Core)
- Ocelot – για την υλοποίηση του Gateway API
- JSON Web Token – για την ταυτοποίηση των χρηστών
- RabbitMQ και MassTransit – για την επικοινωνία μεταξύ των microservices
- Swagger – για την τεκμηρίωση των web services
- Bootstrap 4
- jQuery

Οι παραπάνω τεχνολογίες έχουν αναλυθεί σε προηγούμενα κεφάλαια.

4.6 Εργαλεία

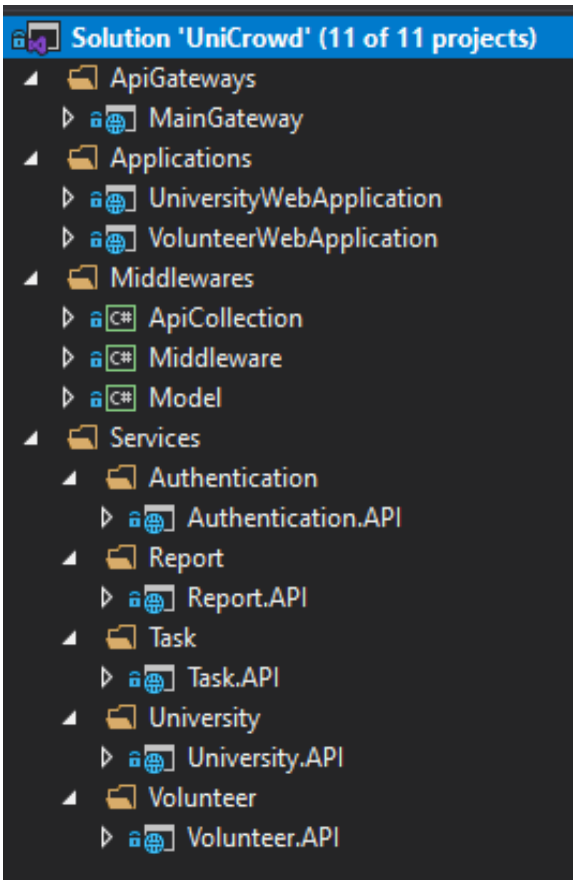
Παρακάτω αναφέρονται τα εργαλεία που χρησιμοποιήθηκαν για την υλοποίηση του συστήματος:

- Visual Studio 2019 (έκδοση 16.8.2)
- Microsoft SQL Server Management Studio (έκδοση 18.6)
- MongoDB Compass (έκδοση 1.23)
- Postman
- GitHub

4.7 Αρχιτεκτονική συστήματος

Ο κώδικας του συστήματος αποτελείται τέσσερα μέρη:

- Δύο εφαρμογές (ASP.NET Core Web Applications)
- Βιβλιοθήκες (class libraries)
- Microservices
- Gateway API

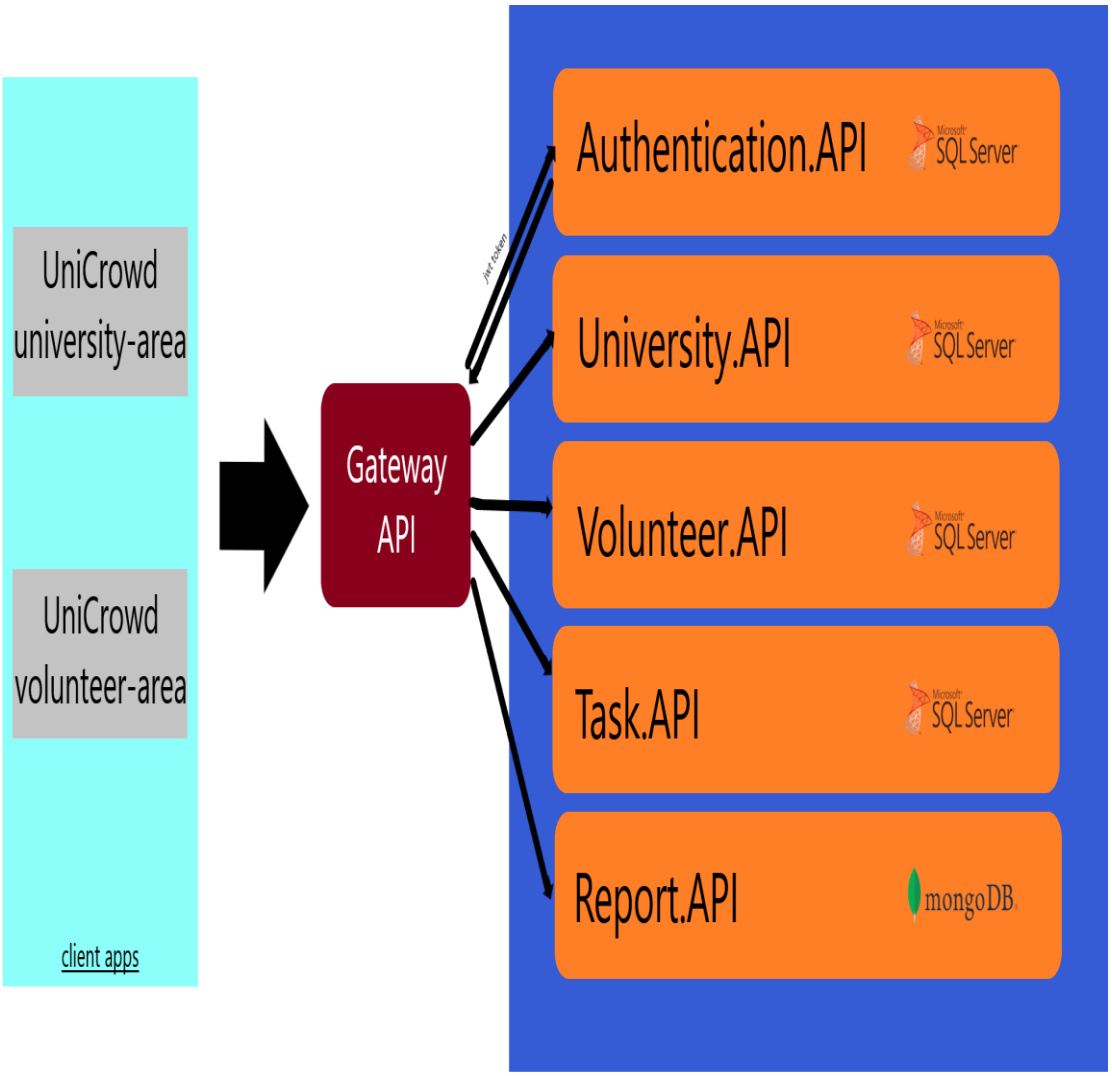


Εικόνα 33: Ο κώδικας του συστήματος χωρισμένος σε τέσσερα μέρη – Στιγμιότυπο από το Visual Studio.

Όπως αναφέρθηκε και παραπάνω, η πλατφόρμα έχει υιοθετήσει την αρχιτεκτονική των microservices. Τα microservices που υλοποιήθηκαν είναι:

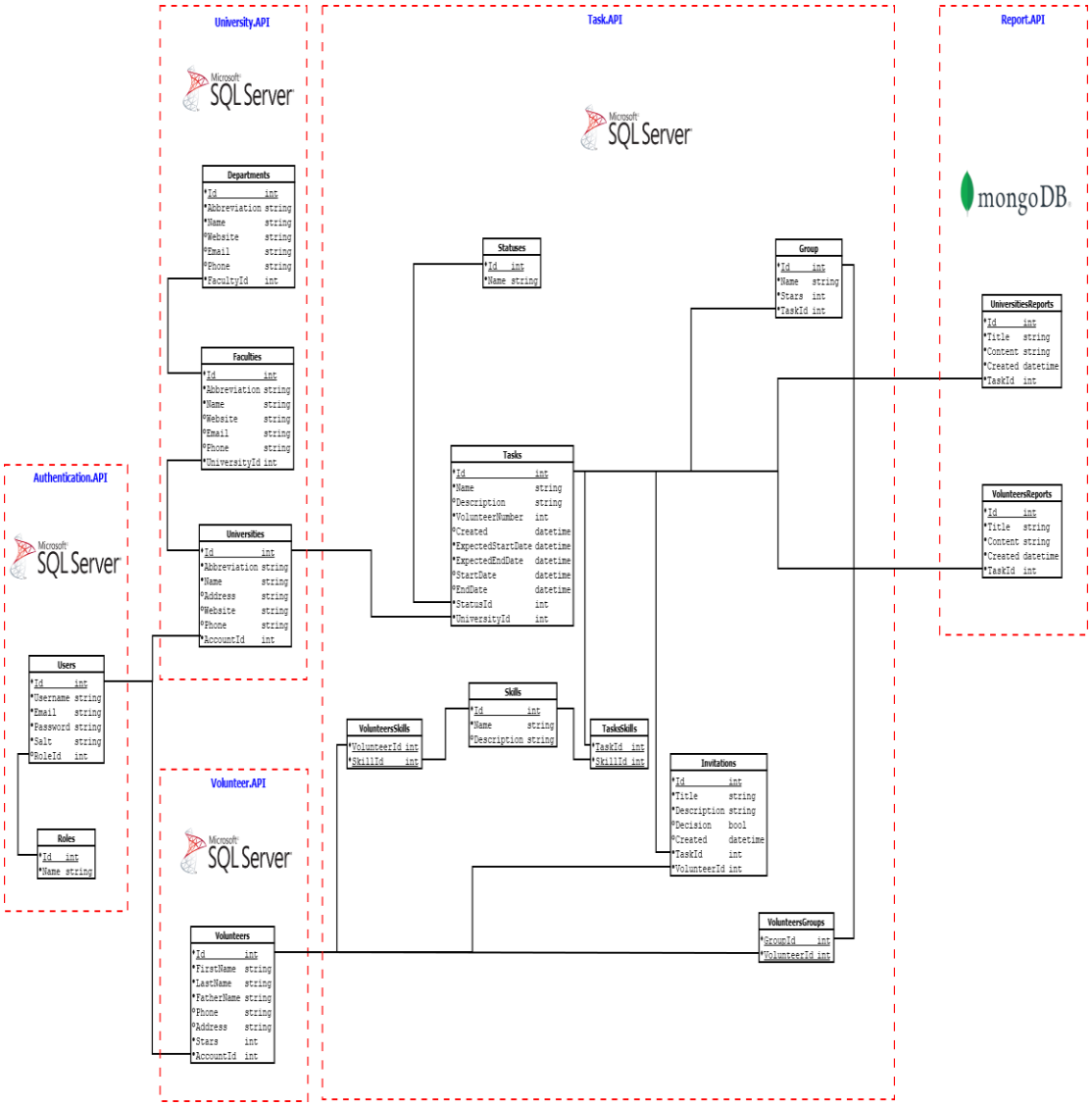
- Authentication.API
- University.API
- Volunteer.API
- Task.API
- Report.API

Ανάμεσα στις εφαρμογές και στα microservices, υπάρχει το gateway API. Το gateway API έχει υλοποιηθεί με το Ocelot.



Εικόνα 34: Η αρχιτεκτονική του συστήματος.

Κάθε microservice συντηρεί τη δικιά του βάση δεδομένων.



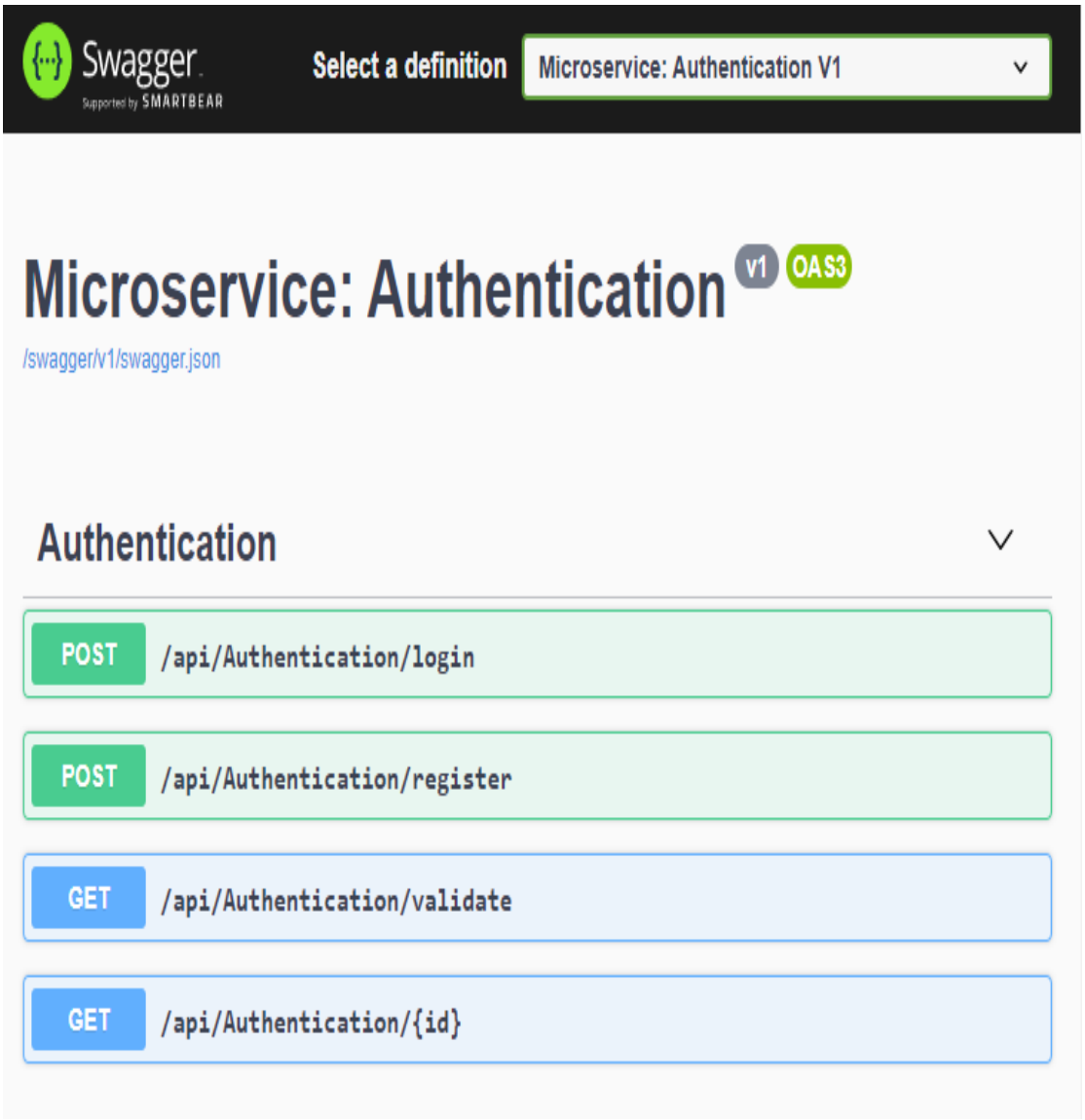
Εικόνα 35: Οι πέντε βάσεις δεδομένων της πλατφόρμας.

Αναλυτικά οι βάσεις δεδομένων του συστήματος:

- UniCrowd-AuthenticationDB (SQL Server) - Πίνακες: Users, Roles.
- UniCrowd-UniversityDB (SQL Server) - Πίνακες: Universities, Faculties, Departments.
- UniCrowd-VolunteerDB (SQL Server) - Πίνακες: Volunteers.
- UniCrowd-TaskDB (SQL Server) - Πίνακες: Tasks, Statuses, Groups, Invitations, Skills, TasksSkills, VolunteersGroups, VolunteersSkills.
- UniCrowd-ReportDB (MongoDB) – Πίνακες: UniversitiesReports, VolunteersReports.

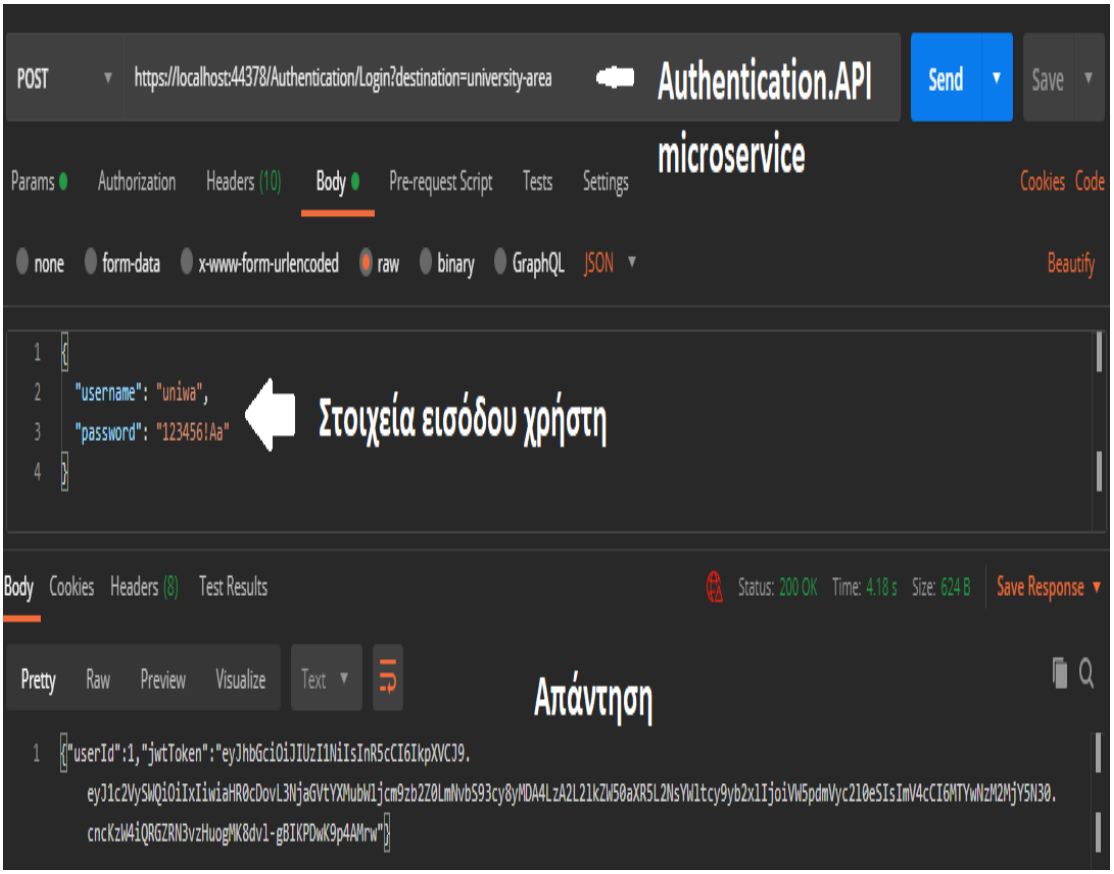
4.7.1 Authentication.API

Η ταυτοποίηση και η εγγραφή των χρηστών γίνεται μέσω του microservice «Authentication.API». Ο διαχωρισμός των ρόλων στις εφαρμογές γίνεται μέσω JSON Web Tokens (JWT). Τα JSON Web Tokens μπορούν να παρέχουν αρκετή πληροφορία, που να βοηθάει τις εφαρμογές να ταυτοποιεί χρήστες.



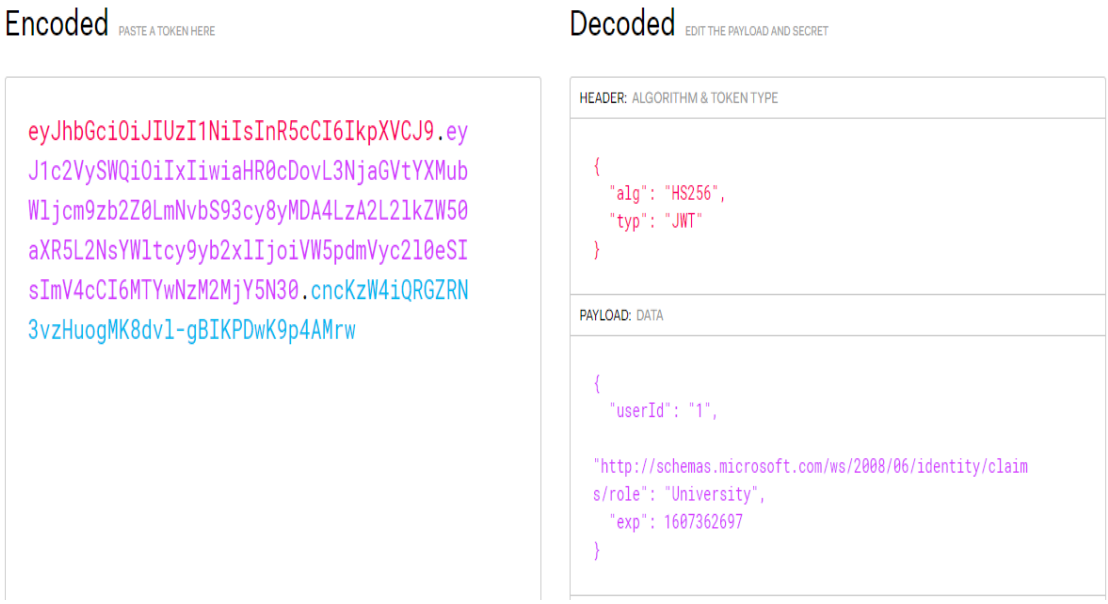
Εικόνα 35: Τα endpoints του «Authentication.API» - Στιγμιότυπο από το Swagger.

Όταν ένας χρήστης πραγματοποιεί είσοδο στο σύστημα, καλείται το microservice «Authentication.API». Εφόσον τα στοιχεία είναι σωστά, το microservice θα επιστρέψει ένα μοναδικό JWT.



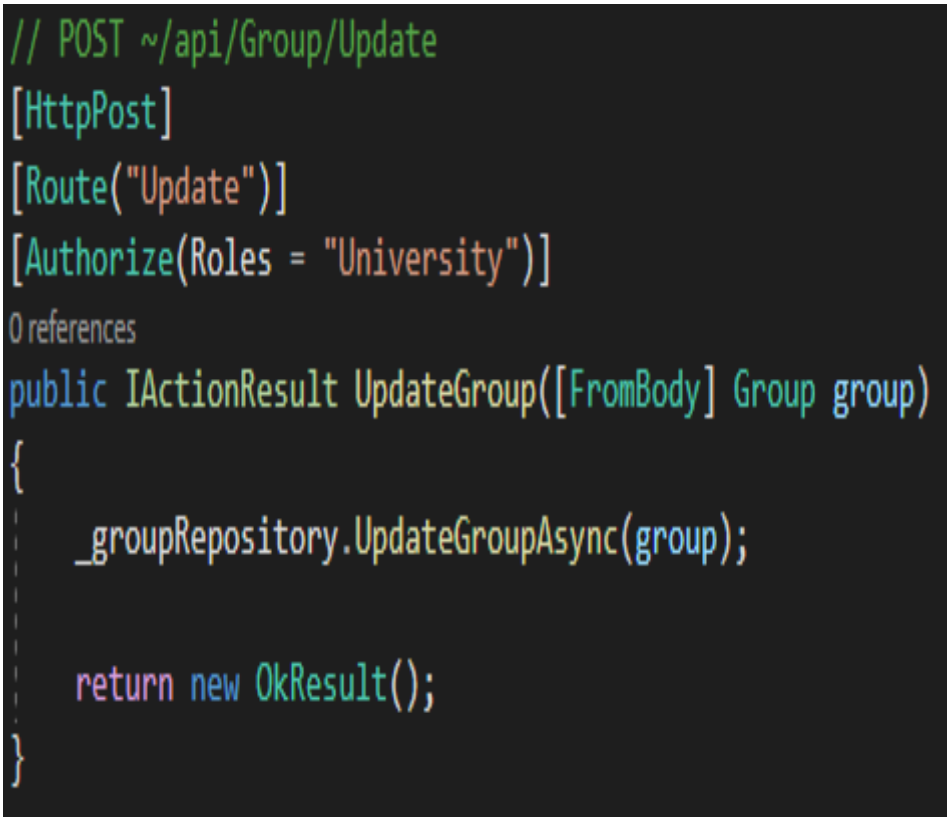
Εικόνα 36: Είσοδος στο σύστημα μέσω του microservice «Authentication.API» - Στιγμιότυπο από το Postman.

Το συγκεκριμένο JWT περιέχει την εξής πληροφορία: α) τον κωδικό του χρήστη (userId) και β) το ρόλο του εκάστοτε χρήστη (πληθοποριστής ή εθελοντής).



Εικόνα 37: Η πληροφορία που κρατάει ένα JWT.

Κατά την περιήγηση του χρήστη στην εφαρμογή, το JWT φυλάγεται σε SESSION μεταβλητή. Έτσι η εφαρμογή ξέρει ανά πάσα στιγμή αν ο χρήστης έχει το δικαίωμα να εκτελέσει μια συγκεκριμένη λειτουργία.



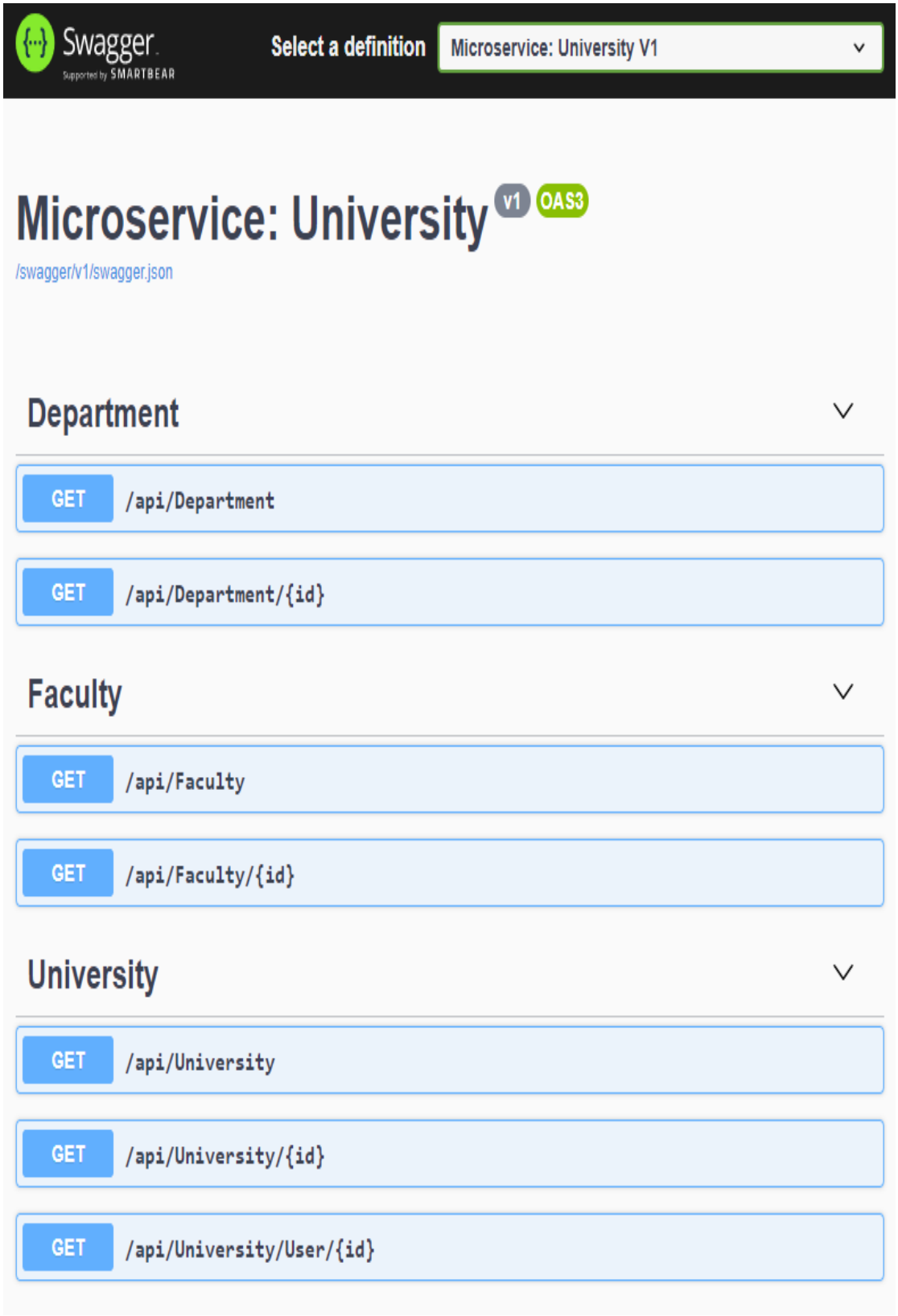
Εικόνα 38: Η εφαρμογή ελέγχει αν ο χρήστης έχει δικαίωμα να εκτελέσει μια λειτουργία- Στιγμιότυπο από το Visual Studio.

Για παράδειγμα, μόνο οι πληθοποριστές έχουν δικαίωμα να διαχειρίζονται τα Groups των Tasks.

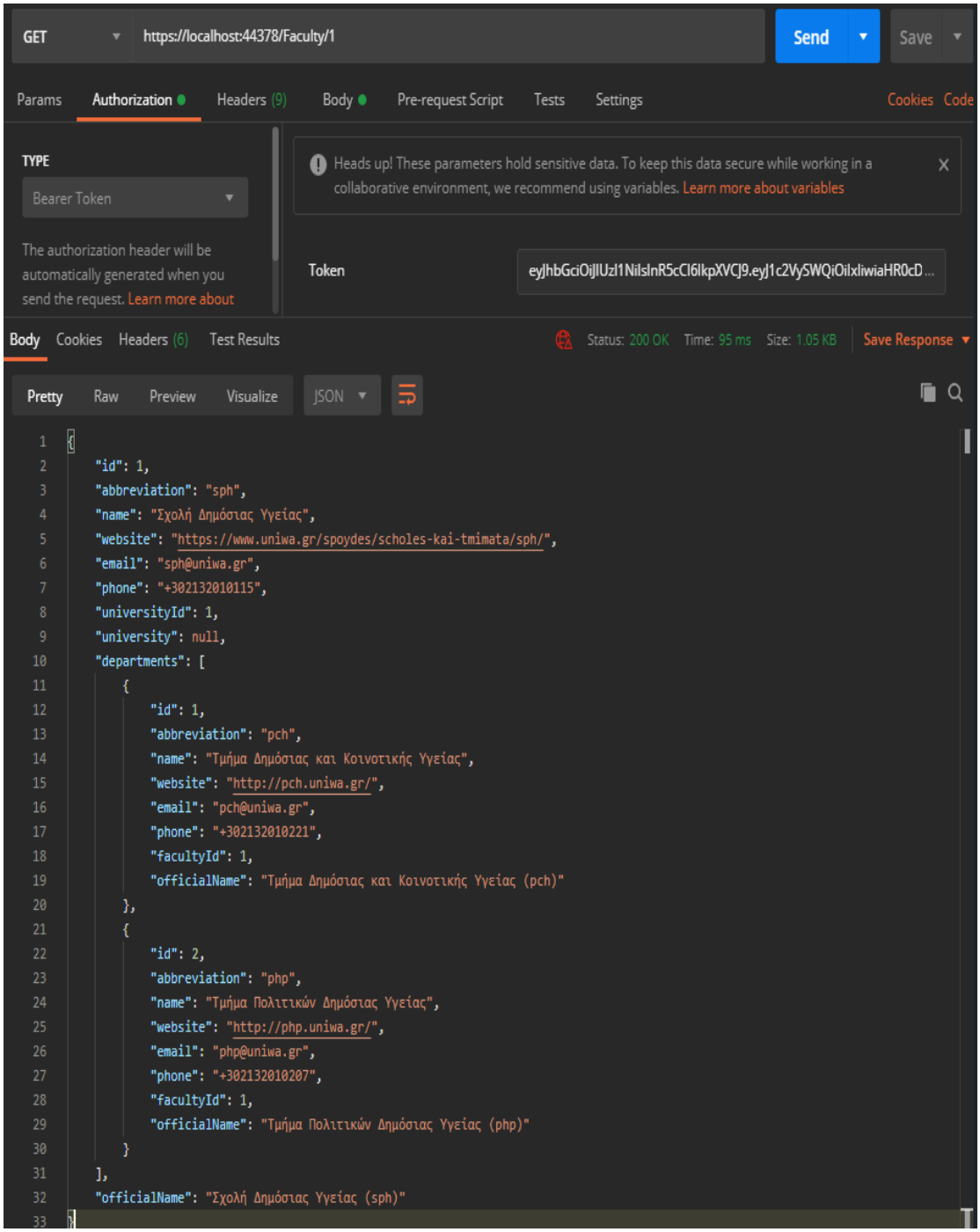
Το «Authentication.API» συντηρεί την SQL Server βάση δεδομένων «UniCrowd-AuthenticationDB». Η διαχείριση της βάσης δεδομένων γίνεται με το Entity Framework Core.

4.7.2 University.API

Το microservice «University.API» παρέχει πληροφορίες για τα καταχωρημένα στο σύστημα πανεπιστήμια.



Εικόνα 39: Τα endpoints του «University.API» - Στιγμιότυπο από το Swagger.

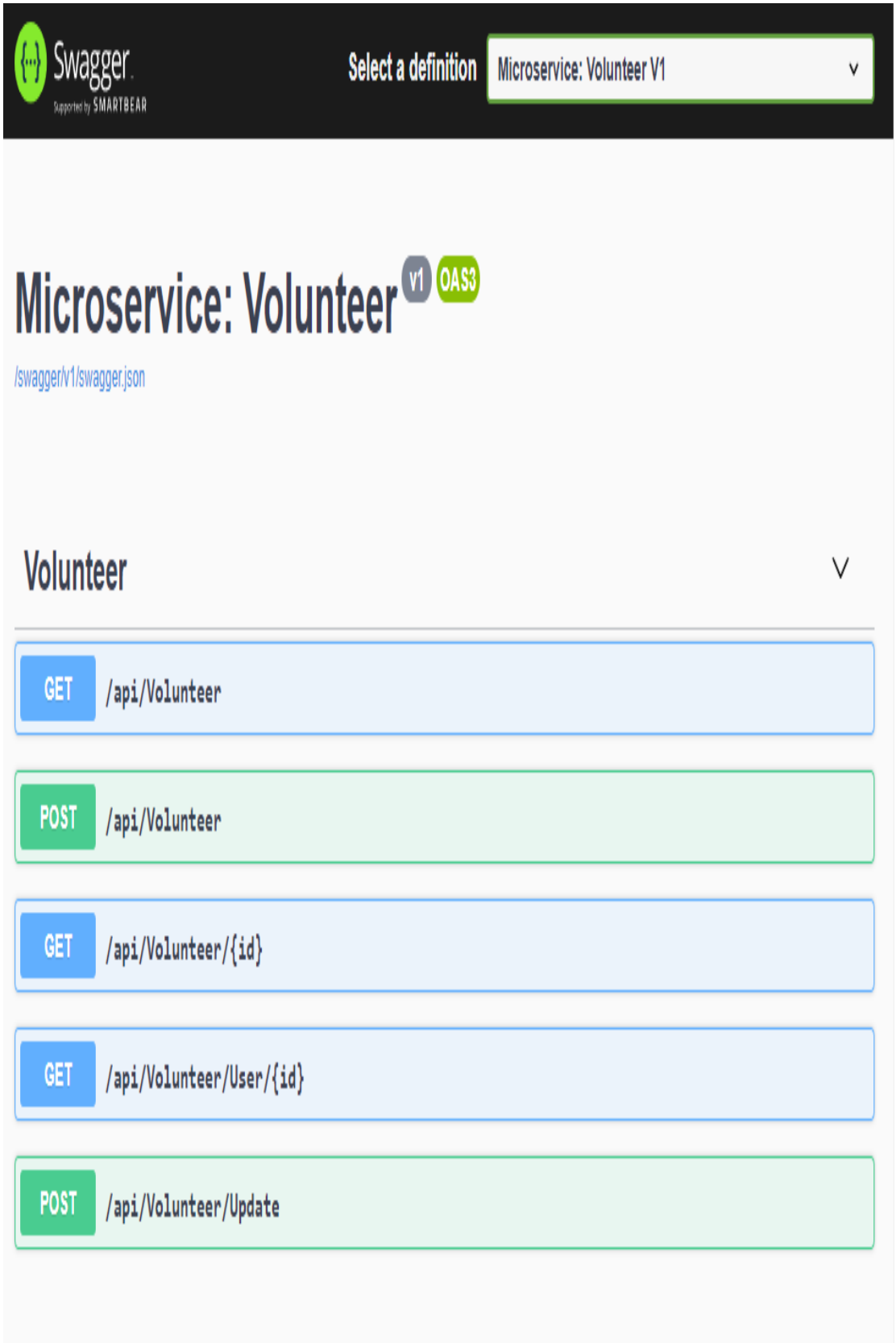


Εικόνα 40: Κλήση του microservice «University.API» - Στιγμιότυπο από το Postman.

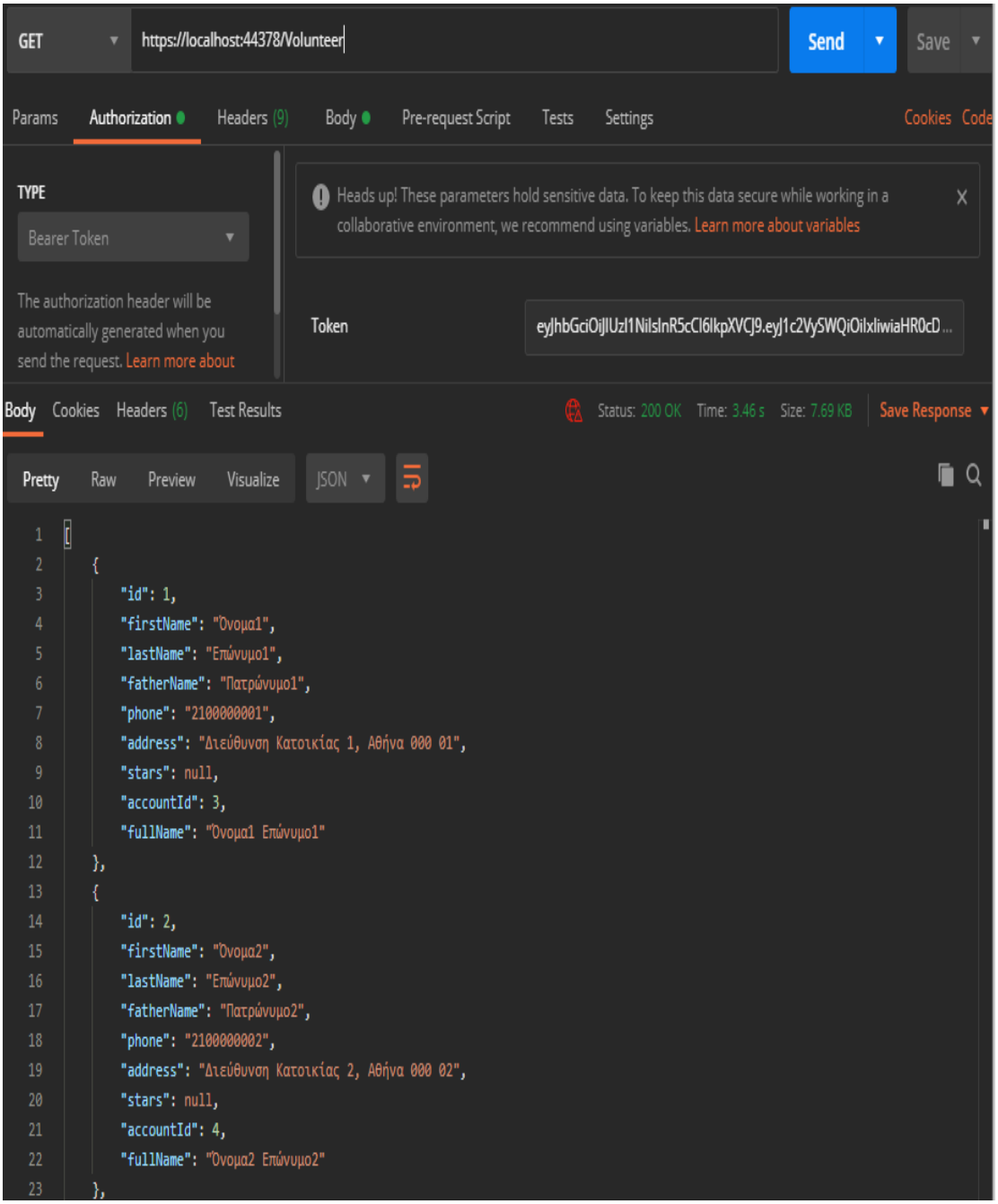
Το «University.API» συντηρεί την SQL Server βάση δεδομένων «UniCrowd-UniversityDB». Η διαχείριση της βάσης δεδομένων γίνεται με το Entity Framework Core.

4.7.3 Volunteer.API

Το microservice «Volunteer.API» παρέχει πληροφορίες για τους καταχωρημένους στο σύστημα εθελοντές.



Εικόνα 41: Τα endpoints του «Volunteer.API» - Στιγμιότυπο από το Swagger.

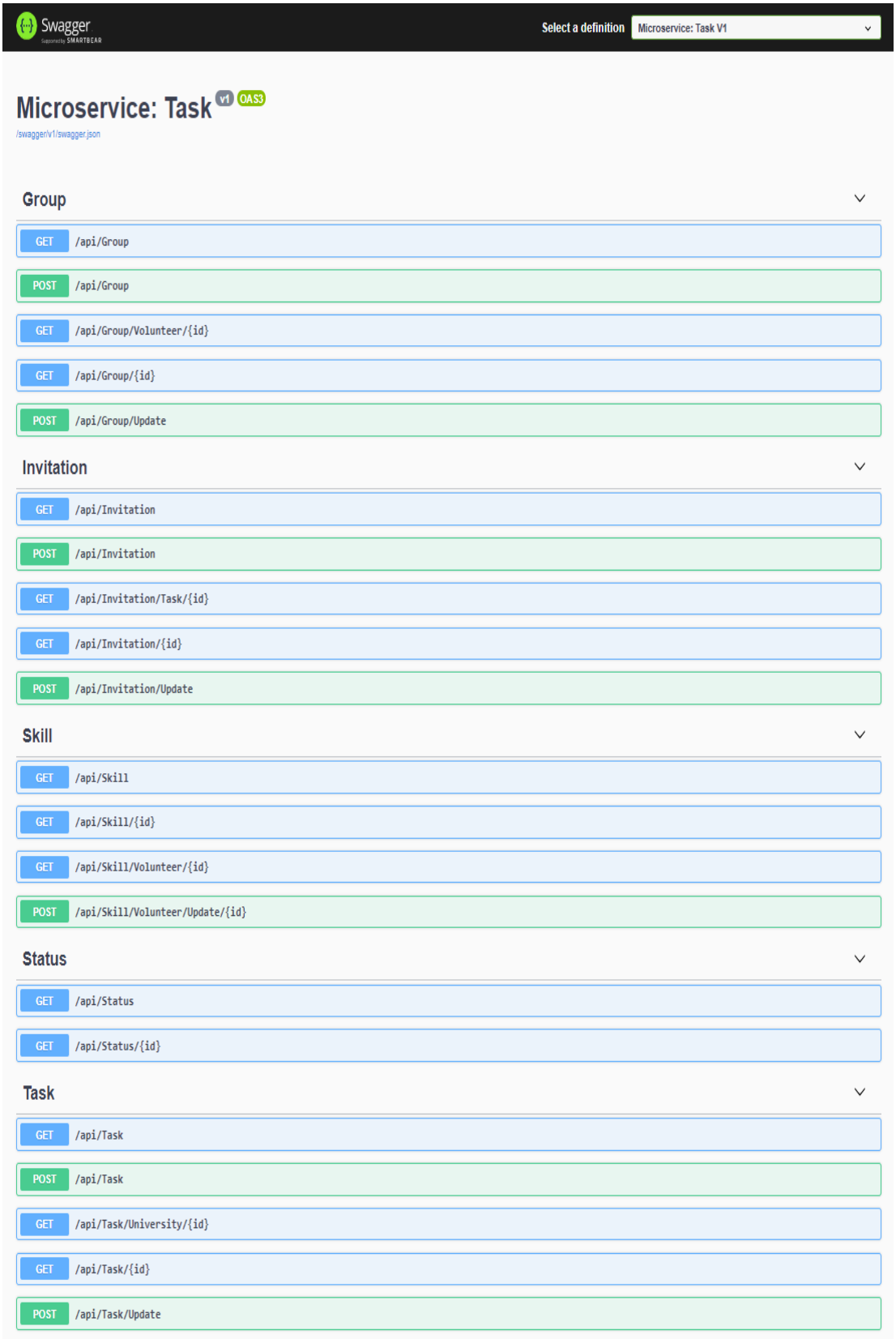


Εικόνα 42: Κλήση του microservice «Volunteer.API» - Στιγμιότυπο από το Postman.

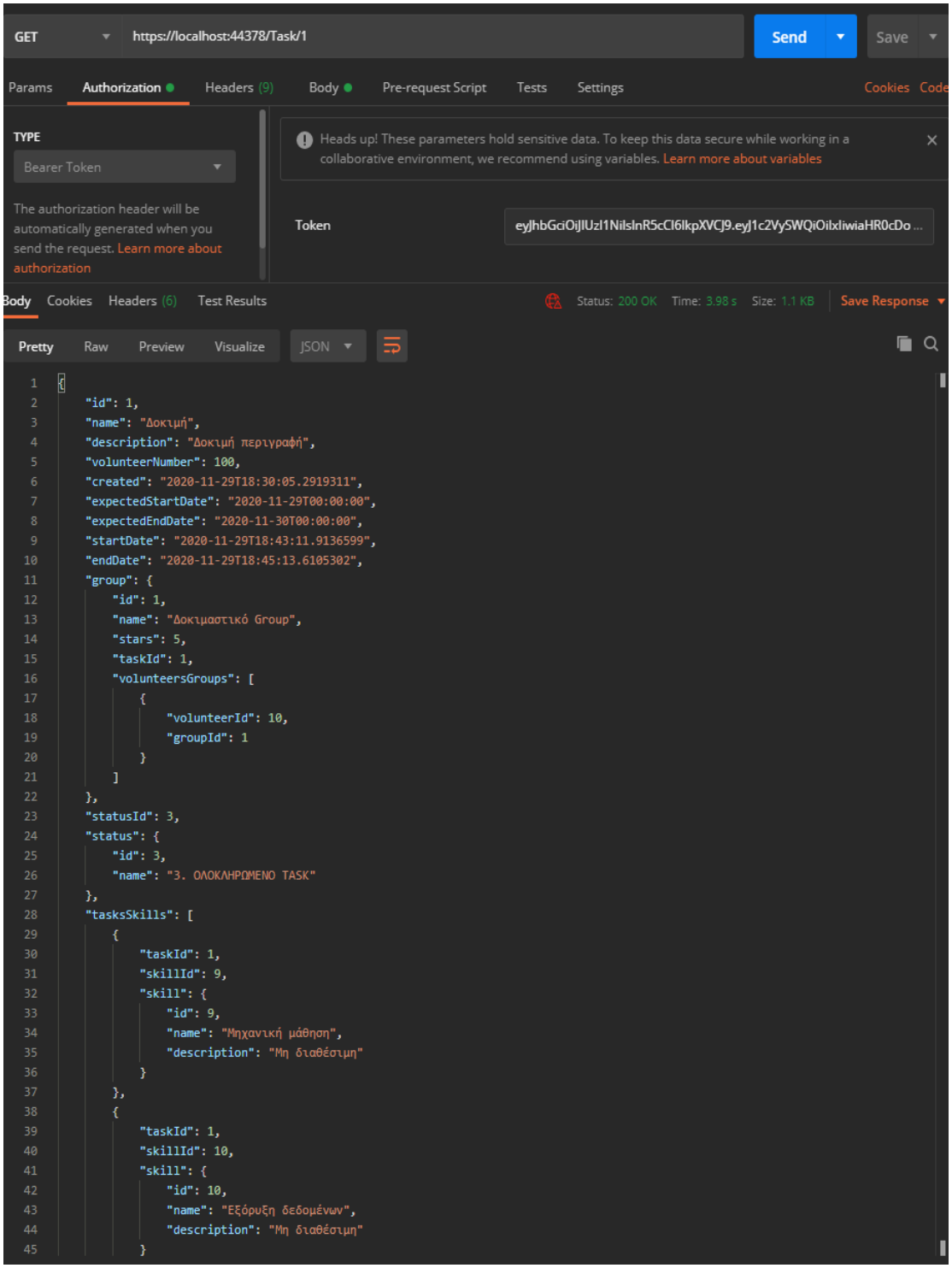
Το «Volunteer.API» συντηρεί την SQL Server βάση δεδομένων «UniCrowd-VolunteerDB». Η διαχείριση της βάσης δεδομένων γίνεται με το Entity Framework Core.

4.7.4 Task.API

Το «Task.API» αποτελεί το μεγαλύτερο microservice του συστήματος. Παρέχει πληροφορίες για τα Tasks, Groups, προσκλήσεις και δεξιότητες.



Εικόνα 43: Τα endpoints του «Task.API» - Στιγμιότυπο από το Swagger.

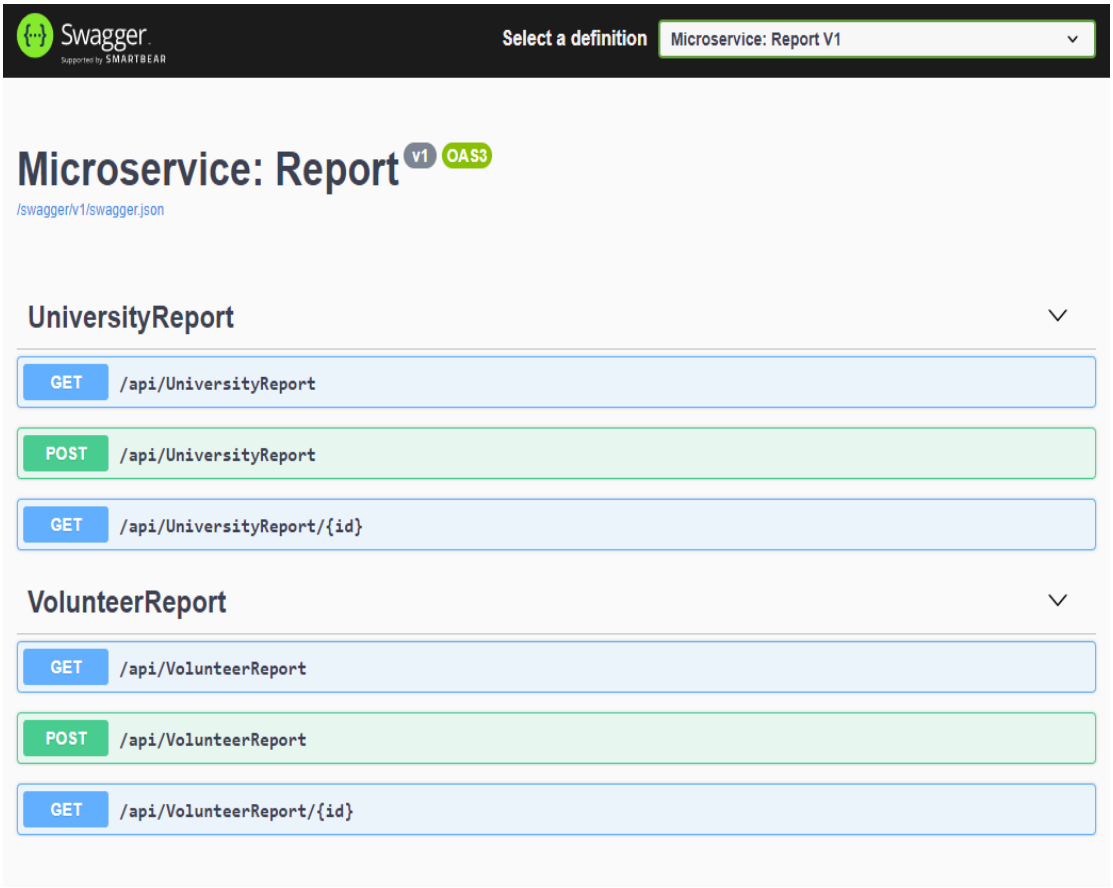


Εικόνα 44: Κλήση του microservice «Task.API» - Στιγμιότυπο από το Postman.

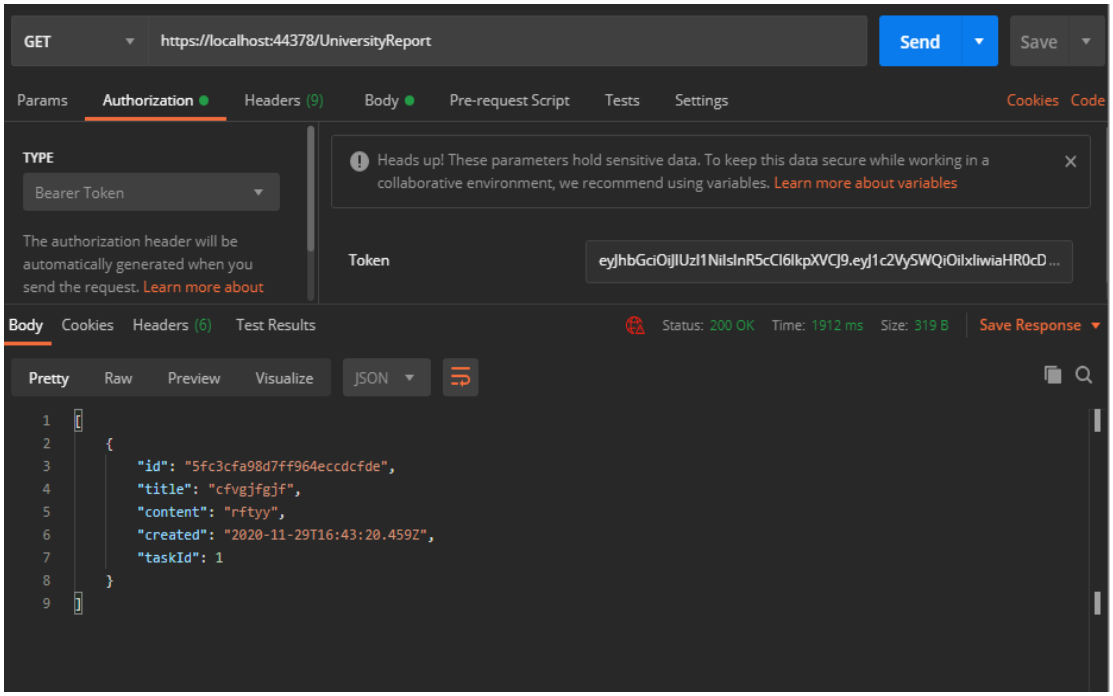
Το «Task.API» συντηρεί την SQL Server βάση δεδομένων «UniCrowd-TaskDB». Η διαχείριση της βάσης δεδομένων γίνεται με το Entity Framework Core.

4.7.5 Report.API

Το microservice «Report.API» παρέχει πληροφορίες για τα Reports των πληθοποριστών και των εθελοντών.



Εικόνα 45: Τα endpoints του «Report.API» - Στιγμιότυπο από το Swagger.



Εικόνα 46: Κλήση του microservice «Report.API» - Στιγμιότυπο από το Postman.

Το «Report.API» συντηρεί την MongoDB βάση δεδομένων «UniCrowd-ReportDB». Η διαχείριση της βάσης δεδομένων γίνεται με το Entity Framework Core.

5 Συμπεράσματα

Ο πληθοπορισμός ενθαρρύνει τόσο τους φοιτητές όσο και τους καθηγητές στη διαδικασία της συνδημιουργίας και της ανταλλαγής γνώσεων.

Δυστυχώς, η νομοθεσία και η γραφειοκρατία κάνουν δύσκολη την εφαρμογή ορισμένων τεχνικών πληθοπορισμού (π.χ. crowdfunding) στην Ανώτατη Εκπαίδευση.

Δεν υπάρχουν πλατφόρμες που να υποστηρίζουν όλες τις τεχνικές πληθοπορισμού.

Ο σχεδιασμός και η υλοποίηση της πλατφόρμας διήρκησε περίπου τέσσερις μήνες. Οι σύγχρονες τεχνολογίες που χρησιμοποιήθηκαν, μείωσαν σημαντικά το χρόνο υλοποίησης. Η αρχιτεκτονική του λογισμικού έχει σχεδιαστεί έτσι, ώστε να το κάνει συντηρήσιμο και επεκτάσιμο.

Η πλατφόρμα «UniCrowd» που αναπτύχθηκε, δε διαφέρει σε πολλά με μια πλατφόρμα πληθοπορισμού γενικού περιεχομένου.

6 Μελλοντικές επεκτάσεις

Παρακάτω αναφέρονται ορισμένες προτάσεις, που μπορούν να βελτιώσουν την πλατφόρμα «UniCrowd»:

Προσθήκη περισσότερων λειτουργιών. Για παράδειγμα, δυνατότητα επισύναψης αρχείων σε κάθε Task.

Βελτίωση του user interface. Μικρές αλλαγές που μπορεί να βοηθήσουν στο να βελτιωθεί η εμπειρία του χρήστη.

Ανάπτυξη νέας εφαρμογής για κινητά τηλέφωνα (mobile applications). Η αρχιτεκτονική των microservices που έχει χρησιμοποιηθεί, θα μειώσει σημαντικά το χρόνο ανάπτυξης. Η Microsoft προσφέρει τεχνολογίες για ανάπτυξη mobile εφαρμογών (Xamarin Forms).

7 Βιβλιογραφία

[1] Enrique Estellés-Arolas, Raúl Navarro-Giner and Fernando González-Ladrón-de-Guevara, 2015, “Crowdsourcing Fundamentals: Definition and Typology, Advances in Crowdsourcing”,

chapter 3, pp. 33-48, Springer, ISBN 978-3-319-18341-1 (eBook), DOI 10.1007/978-3-319-18341-1.

[2] Roberto Llorente and Maria Morant, 2015, “Crowdsourcing in Higher Education”, chapter 7, pp. 87-95, Springer, ISBN 978-3-319-18341-1 (eBook), DOI 10.1007/978-3-319-18341-1.

[3] Prester, Julian; Schlagwein, Daniel; and Cecez-Kecmanovic, Dubravka, 2019, “Crowdsourcing for education: literature review, conceptual framework, and research agenda”, In Proceedings of the 27th European Conference on Information Systems (ECIS), Stockholm & Uppsala, Sweden, June 8-14, 2019.

[4] Michael Anderson, 2011, “Crowdsourcing Higher Education: A Design Proposal for Distributed Learning”, MERLOT Journal of Online Learning and Teaching.

[5] Jeff Howe, 2006, “The rise of crowdsourcing”, Wired magazine, 14(6), pp.1-4.

[6] Ankit Sharma, 2010, “Crowdsourcing Critical Success Factor Model”, Working Paper 1 – 2010.

[7] Microsoft, 2020, “ASP.NET documentation”, https://docs.microsoft.com/en-us/aspnet/core/, [πρόσβαση 30/10/20].

[8] Microsoft, 2020, “Entity Framework Core”, https://docs.microsoft.com/en-us/ef/core/, [πρόσβαση 30/10/20].

[9] Microsoft, 2019, “What is SQL Server Management Studio (SSMS)?”, https://docs.microsoft.com/en-us/sql/ssms/sql-server-management-studio-ssms?view=sql-server-ver15, [πρόσβαση 30/10/20].

[10] MongoDB, 2020, “Get started with MongoDB”, https://docs.mongodb.com/, [πρόσβαση 10/11/20].

[11] Microsoft, 2020, “.NET Microservices: Architecture for Containerized .NET Applications”, https://docs.microsoft.com/en-us/dotnet/architecture/microservices/, [πρόσβαση 30/09/20].

[12] Microsoft, 2020, “Why use a microservices approach to building applications”, https://docs.microsoft.com/en-us/azure/service-fabric/service-fabric-overview-microservices, [πρόσβαση 30/10/20].

[13] Docker, 2020, “Docker documentation”, https://docs.docker.com/, [πρόσβαση 30/09/20].

[14] Microsoft, 2020, “Implement API Gateways with Ocelot”, https://docs.microsoft.com/en-us/dotnet/architecture/microservices/multi-container-microservice-net-applications/implement-api-gateways-with-ocelot, [πρόσβαση 30/09/20].

[15] Wolfgang Ofner, 2020, “RabbitMQ in an ASP .NET Core 3.1 Microservice”, https://www.programmingwithwolfgang.com/rabbitmq-in-an-asp-net-core-3-1-microservice/, [πρόσβαση 30/09/20].

[16] jQuery, 2020, “jQuery documentation”, https://api.jquery.com/, [πρόσβαση 30/10/20].

[17] Bootstrap, 2020, “Build fast, responsive sites with Bootstrap”, https://getbootstrap.com/, [πρόσβαση 30/09/20].

Παράρτημα

UniCrowd\Model\University\University.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.Text;


namespace Model
{
    // University.API

    public class University
    {
        public int Id { get; set; }

        [Display(Name = "Συντομογραφία")]
        public string Abbreviation { get; set; }

        [Display(Name = "Όνομασία")]
        public string Name { get; set; }

        [Display(Name = "Διεύθυνση")]
        public string Address { get; set; }

        [Display(Name = "Ιστοσελίδα")]
        public string Website { get; set; }

        [Display(Name = "Τηλέφωνο")]
        public string Phone { get; set; }

        [Display(Name = "Σχολές")]
        public List<Faculty> Faculties { get; set; }

        [Display(Name = "Λογαριασμός")]
        public int AccountId { get; set; }

        [Display(Name = "Επίσημη ονομασία")]
        public string OfficialName
```

```
        {
            get
            {
                return Name + " (" + Abbreviation + ")";
            }
        }
    }
}
```

UniCrowd\University.API\Controllers\UniversityController.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
using University.API.Data;
using University.API.Services;

namespace University.API.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    [Authorize]
    public class UniversityController : ControllerBase
    {
        private readonly IUniversityRepository _universityRepository;

        public UniversityController(ApplicationDbContext context)
        {
            _universityRepository = new
UniversityRepository(context);
        }

        // GET api/University
        [HttpGet]
        public ActionResult<IEnumerable<Model.University>>
GetUniversities()
        {
            return _universityRepository.GetUniversities().ToList();
        }

        // GET api/University/{id}
        [HttpGet("{id}")]
        public ActionResult<Model.University> GetUniversity(int id)
        {
            return _universityRepository.GetUniversity(id);
        }

        // GET api/University/User/{id}
        [HttpGet("User/{id}")]
        public ActionResult<Model.University>
GetUniversityByUserId([FromRoute] int id)
        {
            return _universityRepository.GetUniversityByUserId(id);
        }
    }
}
```

UniCrowd\University.API\Repositories\Interfaces\IUniversityRepository.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Model;

namespace University.API.Services
{
    public interface IUniversityRepository
    {
        IEnumerable<Model.University> GetUniversities();

        Model.University GetUniversity(int universityId);

        Model.University GetUniversityByUserId(int userId);

        void AddUniversity(Model.University university);

        void UpdateUniversity(Model.University university);

        void DeleteUniversity(int universityId);
    }
}
```

UniCrowd\University.API\Repositories\UniversityRepository.cs

```
using Microsoft.EntityFrameworkCore;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using University.API.Data;
using Model;

namespace University.API.Services
{
    public class UniversityRepository : IUniversityRepository
    {
        private readonly ApplicationDbContext _context;

        public UniversityRepository(ApplicationDbContext context)
        {
            _context = context;
        }

        public IEnumerable<Model.University> GetUniversities()
        {
            return _context
                .Universities
                .Include(u => u.Faculties)
                .ToList();
        }

        public Model.University GetUniversity(int universityId)
        {
            return _context.Universities
                .Include(u => u.Faculties)
                .Where(u => u.Id == universityId)
                .FirstOrDefault();
        }
```

```
        public Model.University GetUniversityByUserId(int userId)
        {
            return _context.Universities
                .Include(u => u.Faculties)
                .Where(u => u.AccountId == userId)
                .FirstOrDefault();
        }
    }
}
```

UniCrowd\ApiCollection\UniversityApi.cs

```
using System;
using System.Collections.Generic;
using System.Collections.Specialized;
using System.Linq;
using System.Net.Http;
using System.Threading.Tasks;
using ApiCollection.Infrastructure;
using ApiCollection.Interfaces;
using Model;

namespace ApiCollection
{
    public class UniversityApi : BaseHttpClientWithFactory,
IUniversityApi
    {
        public UniversityApi(IHttpClientFactory factory)
            : base(factory)
        {
        }

        public async Task<List<University>> GetUniversities(string
jwtToken)
        {
            NameValueCollection authorization = new
NameValueCollection
            {
                { "Authorization", "Bearer " + jwtToken }
            };

            var message = new
HttpRequestBuilder("https://localhost:44378")
                            .SetPath("/University")
                            .HttpMethod(HttpMethod.Get)
                            .Headers(authorization)
                            .GetHttpMessage();

            return await SendRequest<List<University>>(message);
        }

        public async Task<University> GetUniversity(int universityId,
string jwtToken)
        {
            NameValueCollection authorization = new
NameValueCollection
            {
                { "Authorization", "Bearer " + jwtToken }
            };
```

```
            var message = new
HttpRequestBuilder("https://localhost:44378")
                              .SetPath("/University")
                              .AddToPath(universityId.ToString())
                              .HttpMethod(HttpMethod.Get)
                              .Headers(authorization)
                              .GetHttpMessage();

            return await SendRequest<University>(message);
        }
    }
}
```

UniCrowd\UniversityWebApplication\Views\University\Details.cshtml

```
@model Model.University

@{
    ViewData["Title"] = "Προφίλ Πανεπιστημίου";
    Layout = "_Layout";
}

<h1>Προφίλ Πανεπιστημίου</h1>

<hr class="basic" />

<div class="form-group row">
    <div class="col-4">
        <label asp-for="OfficialName"></label>
    </div>
    <div class="col-8">
        <input asp-for="OfficialName" class="form-control" disabled
/>
    </div>
</div>

<div class="form-group row">
    <div class="col-4">
        <label asp-for="Address"></label>
    </div>
    <div class="col-8">
        <input asp-for="Address" class="form-control" disabled />
    </div>
</div>

<div class="form-group row">
    <div class="col-4">
        <label asp-for="Website"></label>
    </div>
    <div class="col-8">
        <input asp-for="Website" class="form-control" disabled />
    </div>
</div>

<div class="form-group row">
    <div class="col-4">
        <label asp-for="Phone"></label>
    </div>
    <div class="col-8">
        <input asp-for="Phone" class="form-control" disabled />
    </div>
```

```
</div>

@{
    if (Model.Faculties.Count > 0)
    {
        <hr class="basic" />

        <h2>Σχολές:</h2>
        <ul>
            @foreach (var item in Model.Faculties)
            {
                <li>
                    <a class="nav-link text-light" asp-area="" asp-
controller="Faculty" asp-action="Details" asp-route-
id="@item.Id">@item.OfficialName</a>
                </li>
            }
        </ul>
    }
}

<div class="form-group row">
    <div class="col-4">
        <a asp-controller="Task" asp-action="University" asp-route-
id="@Model.Id" class="btn btn-basic" role="button">Tasks
Πανεπιστημίου</a>
    </div>
    <div class="col-8">
    </div>
</div>

<hr class="basic" />

<a class="btn btn-basic" href="javascript:history.go(-1)"
role="button">Πίσω</a>

@section scripts
{
    @{
        <script>
            $(document).ready(function () {
            });
        </script>
    }
}
```

UniCrowd\MainGateway\ocelot.Development.json

```
{
  "Routes": [
    // Authentication.API
    {
      "DownstreamPathTemplate": "/api/Authentication/login",
      "DownstreamScheme": "https",
      "DownstreamHostAndPorts": [
        {
          "Host": "localhost",
          "Port": 44319 // Authentication.API SSL Port
        }
      ],
      "UpstreamPathTemplate": "/Authentication/login",
```

```
      "UpstreamHttpMethod": [ "POST" ]
    },
    {
      "DownstreamPathTemplate": "/api/Authentication/register",
      "DownstreamScheme": "https",
      "DownstreamHostAndPorts": [
        {
          "Host": "localhost",
          "Port": 44319 // Authentication.API SSL Port
        }
      ],
      "UpstreamPathTemplate": "/Authentication/register",
      "UpstreamHttpMethod": [ "POST" ]
    },
    {
      "DownstreamPathTemplate":
"/api/Authentication/validate?userId={userId}&jwtToken={jwtToken}",
      "DownstreamScheme": "https",
      "DownstreamHostAndPorts": [
        {
          "Host": "localhost",
          "Port": 44319 // Authentication.API SSL Port
        }
      ],
      "UpstreamPathTemplate":
"/Authentication/validate/{userId}/{jwtToken}",
      "UpstreamHttpMethod": [ "GET" ]
    },
    {
      "DownstreamPathTemplate": "/api/Authentication/{id}",
      "DownstreamScheme": "https",
      "DownstreamHostAndPorts": [
        {
          "Host": "localhost",
          "Port": 44319 // Authentication.API SSL Port
        }
      ],
      "UpstreamPathTemplate": "/Authentication/{id}",
      "UpstreamHttpMethod": [ "GET" ],
      "AuthenticationOptions": {
        "AuthenticationProviderKey": "Bearer",
        "AllowedScopes": []
      }
    },
    // University.API
    {
      "DownstreamPathTemplate": "/api/University",
      "DownstreamScheme": "https",
      "DownstreamHostAndPorts": [
        {
          "Host": "localhost",
          "Port": 44382 // University.API SSL Port
        }
      ],
      "UpstreamPathTemplate": "/University",
      "UpstreamHttpMethod": [ "GET" ],
      "AuthenticationOptions": {
        "AuthenticationProviderKey": "Bearer",
        "AllowedScopes": []
      }
    },
```

```
    {
      "DownstreamPathTemplate": "/api/University/{id}",
      "DownstreamScheme": "https",
      "DownstreamHostAndPorts": [
        {
          "Host": "localhost",
          "Port": 44382 // University.API SSL Port
        }
      ],
      "UpstreamPathTemplate": "/University/{id}",
      "UpstreamHttpMethod": [ "GET" ],
      "AuthenticationOptions": {
        "AuthenticationProviderKey": "Bearer",
        "AllowedScopes": []
      }
    },
    {
      "DownstreamPathTemplate": "/api/University/User/{id}",
      "DownstreamScheme": "https",
      "DownstreamHostAndPorts": [
        {
          "Host": "localhost",
          "Port": 44382 // University.API SSL Port
        }
      ],
      "UpstreamPathTemplate": "/University/User/{id}",
      "UpstreamHttpMethod": [ "GET" ],
      "AuthenticationOptions": {
        "AuthenticationProviderKey": "Bearer",
        "AllowedScopes": []
      }
    },
    {
      "DownstreamPathTemplate": "/api/Faculty",
      "DownstreamScheme": "https",
      "DownstreamHostAndPorts": [
        {
          "Host": "localhost",
          "Port": 44382 // University.API SSL Port
        }
      ],
      "UpstreamPathTemplate": "/Faculty",
      "UpstreamHttpMethod": [ "GET" ],
      "AuthenticationOptions": {
        "AuthenticationProviderKey": "Bearer",
        "AllowedScopes": []
      }
    },
    {
      "DownstreamPathTemplate": "/api/Faculty/{id}",
      "DownstreamScheme": "https",
      "DownstreamHostAndPorts": [
        {
          "Host": "localhost",
          "Port": 44382 // University.API SSL Port
        }
      ],
      "UpstreamPathTemplate": "/Faculty/{id}",
      "UpstreamHttpMethod": [ "GET" ],
      "AuthenticationOptions": {
        "AuthenticationProviderKey": "Bearer",
```

```
      "AllowedScopes": []
    }
  },
  {
    "DownstreamPathTemplate": "/api/Department",
    "DownstreamScheme": "https",
    "DownstreamHostAndPorts": [
      {
        "Host": "localhost",
        "Port": 44382 // University.API SSL Port
      }
    ],
    "UpstreamPathTemplate": "/Department",
    "UpstreamHttpMethod": [ "GET" ],
    "AuthenticationOptions": {
      "AuthenticationProviderKey": "Bearer",
      "AllowedScopes": []
    }
  },
  {
    "DownstreamPathTemplate": "/api/Department/{id}",
    "DownstreamScheme": "https",
    "DownstreamHostAndPorts": [
      {
        "Host": "localhost",
        "Port": 44382 // University.API SSL Port
      }
    ],
    "UpstreamPathTemplate": "/Department/{id}",
    "UpstreamHttpMethod": [ "GET" ],
    "AuthenticationOptions": {
      "AuthenticationProviderKey": "Bearer",
      "AllowedScopes": []
    }
  },
  // Volunteer.API
  {
    "DownstreamPathTemplate": "/api/Volunteer",
    "DownstreamScheme": "https",
    "DownstreamHostAndPorts": [
      {
        "Host": "localhost",
        "Port": 44399 // Volunteer.API SSL Port
      }
    ],
    "UpstreamPathTemplate": "/Volunteer",
    "UpstreamHttpMethod": [ "GET", "POST" ],
    "AuthenticationOptions": {
      "AuthenticationProviderKey": "Bearer",
      "AllowedScopes": []
    }
  },
  {
    "DownstreamPathTemplate": "/api/Volunteer/{id}",
    "DownstreamScheme": "https",
    "DownstreamHostAndPorts": [
      {
        "Host": "localhost",
        "Port": 44399 // Volunteer.API SSL Port
      }
    ],
```

```
      "UpstreamPathTemplate": "/Volunteer/{id}",
      "UpstreamHttpMethod": [ "GET" ],
      "AuthenticationOptions": {
        "AuthenticationProviderKey": "Bearer",
        "AllowedScopes": []
      }
    },
    {
      "DownstreamPathTemplate": "/api/Volunteer/Update",
      "DownstreamScheme": "https",
      "DownstreamHostAndPorts": [
        {
          "Host": "localhost",
          "Port": 44399 // Volunteer.API SSL Port
        }
      ],
      "UpstreamPathTemplate": "/Volunteer/Update",
      "UpstreamHttpMethod": [ "POST" ],
      "AuthenticationOptions": {
        "AuthenticationProviderKey": "Bearer",
        "AllowedScopes": []
      }
    },
    // Task.API
    {
      "DownstreamPathTemplate": "/api/Task",
      "DownstreamScheme": "https",
      "DownstreamHostAndPorts": [
        {
          "Host": "localhost",
          "Port": 44347 // Task.API SSL Port
        }
      ],
      "UpstreamPathTemplate": "/Task",
      "UpstreamHttpMethod": [ "GET", "POST" ],
      "AuthenticationOptions": {
        "AuthenticationProviderKey": "Bearer",
        "AllowedScopes": []
      }
    },
    {
      "DownstreamPathTemplate": "/api/Task/Update",
      "DownstreamScheme": "https",
      "DownstreamHostAndPorts": [
        {
          "Host": "localhost",
          "Port": 44347 // Task.API SSL Port
        }
      ],
      "UpstreamPathTemplate": "/Task/Update",
      "UpstreamHttpMethod": [ "GET", "POST" ],
      "AuthenticationOptions": {
        "AuthenticationProviderKey": "Bearer",
        "AllowedScopes": []
      }
    },
    {
      "DownstreamPathTemplate": "/api/Task/University/{id}",
      "DownstreamScheme": "https",
      "DownstreamHostAndPorts": [
        {
```

```
          "Host": "localhost",
          "Port": 44347 // Task.API SSL Port
        }
      ],
      "UpstreamPathTemplate": "/Task/University/{id}",
      "UpstreamHttpMethod": [ "GET" ],
      "AuthenticationOptions": {
        "AuthenticationProviderKey": "Bearer",
        "AllowedScopes": []
      }
    },
    {
      "DownstreamPathTemplate": "/api/Task/{id}",
      "DownstreamScheme": "https",
      "DownstreamHostAndPorts": [
        {
          "Host": "localhost",
          "Port": 44347 // Task.API SSL Port
        }
      ],
      "UpstreamPathTemplate": "/Task/{id}",
      "UpstreamHttpMethod": [ "GET" ],
      "AuthenticationOptions": {
        "AuthenticationProviderKey": "Bearer",
        "AllowedScopes": []
      }
    },
    {
      "DownstreamPathTemplate": "/api/Skill",
      "DownstreamScheme": "https",
      "DownstreamHostAndPorts": [
        {
          "Host": "localhost",
          "Port": 44347 // Task.API SSL Port
        }
      ],
      "UpstreamPathTemplate": "/Skill",
      "UpstreamHttpMethod": [ "GET" ],
      "AuthenticationOptions": {
        "AuthenticationProviderKey": "Bearer",
        "AllowedScopes": []
      }
    },
    {
      "DownstreamPathTemplate": "/api/Skill/{id}",
      "DownstreamScheme": "https",
      "DownstreamHostAndPorts": [
        {
          "Host": "localhost",
          "Port": 44347 // Task.API SSL Port
        }
      ],
      "UpstreamPathTemplate": "/Skill/{id}",
      "UpstreamHttpMethod": [ "GET" ],
      "AuthenticationOptions": {
        "AuthenticationProviderKey": "Bearer",
        "AllowedScopes": []
      }
    },
    {
      "DownstreamPathTemplate": "/api/Skill/Volunteer/{id}",
```

```
      "DownstreamScheme": "https",
      "DownstreamHostAndPorts": [
        {
          "Host": "localhost",
          "Port": 44347 // Task.API SSL Port
        }
      ],
      "UpstreamPathTemplate": "/Skill/Volunteer/{id}",
      "UpstreamHttpMethod": [ "GET" ],
      "AuthenticationOptions": {
        "AuthenticationProviderKey": "Bearer",
        "AllowedScopes": []
      }
    },
    {
      "DownstreamPathTemplate": "/api/Skill/Volunteer/Update/{id}",
      "DownstreamScheme": "https",
      "DownstreamHostAndPorts": [
        {
          "Host": "localhost",
          "Port": 44347 // Task.API SSL Port
        }
      ],
      "UpstreamPathTemplate": "/Skill/Volunteer/Update/{id}",
      "UpstreamHttpMethod": [ "POST" ],
      "AuthenticationOptions": {
        "AuthenticationProviderKey": "Bearer",
        "AllowedScopes": []
      }
    },
    {
      "DownstreamPathTemplate": "/api/Status",
      "DownstreamScheme": "https",
      "DownstreamHostAndPorts": [
        {
          "Host": "localhost",
          "Port": 44347 // Task.API SSL Port
        }
      ],
      "UpstreamPathTemplate": "/Status",
      "UpstreamHttpMethod": [ "GET" ],
      "AuthenticationOptions": {
        "AuthenticationProviderKey": "Bearer",
        "AllowedScopes": []
      }
    },
    {
      "DownstreamPathTemplate": "/api/Status/{id}",
      "DownstreamScheme": "https",
      "DownstreamHostAndPorts": [
        {
          "Host": "localhost",
          "Port": 44347 // Task.API SSL Port
        }
      ],
      "UpstreamPathTemplate": "/Status/{id}",
      "UpstreamHttpMethod": [ "GET" ],
      "AuthenticationOptions": {
        "AuthenticationProviderKey": "Bearer",
        "AllowedScopes": []
      }
```

```
    },
    {
      "DownstreamPathTemplate": "/api/Invitation",
      "DownstreamScheme": "https",
      "DownstreamHostAndPorts": [
        {
          "Host": "localhost",
          "Port": 44347 // Task.API SSL Port
        }
      ],
      "UpstreamPathTemplate": "/Invitation",
      "UpstreamHttpMethod": [ "GET", "POST" ],
      "AuthenticationOptions": {
        "AuthenticationProviderKey": "Bearer",
        "AllowedScopes": []
      }
    },
    {
      "DownstreamPathTemplate": "/api/Invitation/Task/{id}",
      "DownstreamScheme": "https",
      "DownstreamHostAndPorts": [
        {
          "Host": "localhost",
          "Port": 44347 // Task.API SSL Port
        }
      ],
      "UpstreamPathTemplate": "/Invitation/Task/{id}",
      "UpstreamHttpMethod": [ "GET" ],
      "AuthenticationOptions": {
        "AuthenticationProviderKey": "Bearer",
        "AllowedScopes": []
      }
    },
    {
      "DownstreamPathTemplate": "/api/Invitation/{id}",
      "DownstreamScheme": "https",
      "DownstreamHostAndPorts": [
        {
          "Host": "localhost",
          "Port": 44347 // Task.API SSL Port
        }
      ],
      "UpstreamPathTemplate": "/Invitation/{id}",
      "UpstreamHttpMethod": [ "GET" ],
      "AuthenticationOptions": {
        "AuthenticationProviderKey": "Bearer",
        "AllowedScopes": []
      }
    },
    {
      "DownstreamPathTemplate": "/api/Invitation/Update",
      "DownstreamScheme": "https",
      "DownstreamHostAndPorts": [
        {
          "Host": "localhost",
          "Port": 44347 // Task.API SSL Port
        }
      ],
      "UpstreamPathTemplate": "/Invitation/Update",
      "UpstreamHttpMethod": [ "GET", "POST" ],
      "AuthenticationOptions": {
```

```
        "AuthenticationProviderKey": "Bearer",
        "AllowedScopes": []
      }
    },
    {
      "DownstreamPathTemplate": "/api/Group",
      "DownstreamScheme": "https",
      "DownstreamHostAndPorts": [
        {
          "Host": "localhost",
          "Port": 44347 // Task.API SSL Port
        }
      ],
      "UpstreamPathTemplate": "/Group",
      "UpstreamHttpMethod": [ "GET" ],
      "AuthenticationOptions": {
        "AuthenticationProviderKey": "Bearer",
        "AllowedScopes": []
      }
    },
    {
      "DownstreamPathTemplate": "/api/Group/Volunteer/{id}",
      "DownstreamScheme": "https",
      "DownstreamHostAndPorts": [
        {
          "Host": "localhost",
          "Port": 44347 // Task.API SSL Port
        }
      ],
      "UpstreamPathTemplate": "/Group/Volunteer/{id}",
      "UpstreamHttpMethod": [ "GET" ],
      "AuthenticationOptions": {
        "AuthenticationProviderKey": "Bearer",
        "AllowedScopes": []
      }
    },
    {
      "DownstreamPathTemplate": "/api/Group/{id}",
      "DownstreamScheme": "https",
      "DownstreamHostAndPorts": [
        {
          "Host": "localhost",
          "Port": 44347 // Task.API SSL Port
        }
      ],
      "UpstreamPathTemplate": "/Group/{id}",
      "UpstreamHttpMethod": [ "GET" ],
      "AuthenticationOptions": {
        "AuthenticationProviderKey": "Bearer",
        "AllowedScopes": []
      }
    },
    {
      "DownstreamPathTemplate": "/api/Group/Update",
      "DownstreamScheme": "https",
      "DownstreamHostAndPorts": [
        {
          "Host": "localhost",
          "Port": 44347 // Task.API SSL Port
        }
      ],
```

```
      "UpstreamPathTemplate": "/Group/Update",
      "UpstreamHttpMethod": [ "GET", "POST" ],
      "AuthenticationOptions": {
        "AuthenticationProviderKey": "Bearer",
        "AllowedScopes": []
      }
    },
    // Report.API
    {
      "DownstreamPathTemplate": "/api/UniversityReport",
      "DownstreamScheme": "https",
      "DownstreamHostAndPorts": [
        {
          "Host": "localhost",
          "Port": 44316 // Report.API SSL Port
        }
      ],
      "UpstreamPathTemplate": "/UniversityReport",
      "UpstreamHttpMethod": [ "GET", "POST" ],
      "AuthenticationOptions": {
        "AuthenticationProviderKey": "Bearer",
        "AllowedScopes": []
      }
    },
    {
      "DownstreamPathTemplate": "/api/UniversityReport/{id}",
      "DownstreamScheme": "https",
      "DownstreamHostAndPorts": [
        {
          "Host": "localhost",
          "Port": 44316 // Report.API SSL Port
        }
      ],
      "UpstreamPathTemplate": "/UniversityReport/{id}",
      "UpstreamHttpMethod": [ "GET" ],
      "AuthenticationOptions": {
        "AuthenticationProviderKey": "Bearer",
        "AllowedScopes": []
      }
    },
    {
      "DownstreamPathTemplate": "/api/VolunteerReport",
      "DownstreamScheme": "https",
      "DownstreamHostAndPorts": [
        {
          "Host": "localhost",
          "Port": 44316 // Report.API SSL Port
        }
      ],
      "UpstreamPathTemplate": "/VolunteerReport",
      "UpstreamHttpMethod": [ "GET", "POST" ],
      "AuthenticationOptions": {
        "AuthenticationProviderKey": "Bearer",
        "AllowedScopes": []
      }
    },
    {
      "DownstreamPathTemplate": "/api/VolunteerReport/{id}",
      "DownstreamScheme": "https",
      "DownstreamHostAndPorts": [
        {
```

```
          "Host": "localhost",
          "Port": 44316 // Report.API SSL Port
        }
      ],
      "UpstreamPathTemplate": "/VolunteerReport/{id}",
      "UpstreamHttpMethod": [ "GET" ],
      "AuthenticationOptions": {
        "AuthenticationProviderKey": "Bearer",
        "AllowedScopes": []
      }
    }
  ],
  "GlobalConfiguration": {
    "BaseUrl": "https://localhost:44378" // MainGateway SSL Port
  }
}
```

UniCrowd\MainGateway\Startup.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Authentication.JwtBearer;
using Microsoft.AspNetCore.Builder;
using Microsoft.AspNetCore.Hosting;
using Microsoft.AspNetCore.Http;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;
using Microsoft.IdentityModel.Tokens;
using Ocelot.DependencyInjection;
using Ocelot.Middleware;
using Ocelot.Cache.CacheManager;
using Middleware;
using System.Configuration;
using Microsoft.Extensions.Configuration;

namespace MainGateway
{
    public class Startup
    {
        public Startup(IConfiguration configuration)
        {
            Configuration = configuration;
        }

        public IConfiguration Configuration { get; }

        // 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)
        {
            services.AddControllers();

            services
                .AddOcelot(Configuration)
                .AddCacheManager(configCacheBuilder =>
                {
```

```
                    configCacheBuilder.WithDictionaryHandle();
                });

                var jwtSection = Configuration.GetSection("jwt");
                var jwtOptions = jwtSection.Get<JwtOptions>();
                var key = Encoding.UTF8.GetBytes(jwtOptions.Secret);

                services.AddAuthentication(option => {
                    option.DefaultAuthenticateScheme =
JwtBearerDefaults.AuthenticationScheme;
                    option.DefaultChallengeScheme =
JwtBearerDefaults.AuthenticationScheme;
                }).AddJwtBearer(options => {
                    options.RequireHttpsMetadata = false;
                    options.SaveToken = true;
                    options.TokenValidationParameters = new
TokenValidationParameters
                {
                    IssuerSigningKey = new SymmetricSecurityKey(key),
                    ValidateIssuerSigningKey = true,
                    ValidateIssuer = false,
                    ValidateAudience = false
                };
            });

            //====================================

            services.AddCors(options =>
            {
                options.AddPolicy("CorsPolicy",
                    builder => builder.AllowAnyOrigin()
                        .AllowAnyMethod()
                        .AllowAnyHeader());
            });
        }

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

            app.UseMiddleware<RequestResponseLoggingMiddleware>();
            app.UseCors("CorsPolicy");

            app.UseAuthentication();
            app.UseOcelot().Wait();
        }
    }
}
```