ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

Σχολή Τεχνολογίας

Τμήμα Ψηφιακών Συστημάτων

Ανάπτυξη διαδικτυακής εφαρμογής ευρέσεως εργασίας

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Αδάμος Αλέξανδρος (ΑΜ: 3121178)

Επιβλέπων: ***Κουτσονικόλα Βασιλική, βαθμίδα***

ΛΑΡΙΣΑ, Ιούλιος 2025

UNIVERSITY OF THESSALY

School of Technology

Digital Systems Department

Development of a web application for online job searching

BACHELOR THESIS

Adamos Alexandros (RN: 3121178)

Supervisor: Koutsonikola Vasiliki*, position*

LARISSA, July 2025

Υπεύθυνη Δήλωση περί Ακαδημαϊκής Δεοντολογίας και Πνευματικών Δικαιωμάτων

Με πλήρη επίγνωση των συνεπειών του νόμου περί πνευματικών δικαιωμάτων, δηλώνω ρητά ότι η παρούσα πτυχιακή εργασία, καθώς και τα ηλεκτρονικά αρχεία και πηγαίοι κώδικες που αναπτύχθηκαν ή τροποποιήθηκαν στα πλαίσια αυτής της εργασίας, αποτελεί αποκλειστικά προϊόν προσωπικής μου εργασίας, δεν προσβάλλει κάθε μορφής δικαιώματα διανοητικής ιδιοκτησίας, προσωπικότητας και προσωπικών δεδομένων τρίτων, δεν περιέχει έργα/εισφορές τρίτων για τα οποία απαιτείται άδεια των δημιουργών/δικαιούχων και δεν είναι προϊόν μερικής ή ολικής αντιγραφής, οι πηγές δε που χρησιμοποιήθηκαν περιορίζονται στις βιβλιογραφικές αναφορές και μόνον, και πληρούν τους κανόνες της επιστημονικής παράθεσης. Τα σημεία όπου έχω χρησιμοποιήσει ιδέες, κείμενο, αρχεία ή/και πηγές άλλων συγγραφέων, αναφέρονται ευδιάκριτα στο κείμενο με την κατάλληλη παραπομπή και η σχετική αναφορά περιλαμβάνεται στο τμήμα των βιβλιογραφικών αναφορών με πλήρη περιγραφή.

Αναλαμβάνω πλήρως, ατομικά και προσωπικά, όλες τις νομικές και διοικητικές συνέπειες που δύναται να προκύψουν στην περίπτωση κατά την οποία αποδειχθεί, διαχρονικά, ότι η εργασία αυτή ή τμήμα της δεν μου ανήκει διότι είναι προϊόν λογοκλοπής.

Ο Δηλών

(Υπογραφή)

Αδάμος Αλέξανδρος

2/7/2025

Εγκρίνεται από την Επιτροπή Εξέτασης:

Επιβλέπων/πουσα **Κουτσονικόλα Βασιλική**
Βαθμίδα/ιδιότητα επιβλέποντα,
Τμήμα Ψηφιακών Συστημάτων, Πανεπιστήμιο Θεσσαλίας

Μέλος **Αβραμούλη Δήμητρα**
Βαθμίδα/ιδιότητα μέλους 1,
Τμήμα Ψηφιακών Συστημάτων, Πανεπιστήμιο Θεσσαλίας

Μέλος **Λαΐτσου Ελένη**
Βαθμίδα/ιδιότητα μέλους 2,
Τμήμα Ψηφιακών Συστημάτων, Πανεπιστήμιο Θεσσαλίας

Ημερομηνία έγκρισης: 02-07-2025

Περίληψη

Η εργασία αυτή έχει σκοπό να βοηθήσει τους ανθρώπους οι οποίοι αναζητούν εργασία να βρούν διαθέσιμες θέσεις εργασίας στις οποίες μπορούν να κάνουν αίτηση. Επιπλέον, οι εργοδότες μπορούν και εκείνοι αντίστοιχα να βρούν διαθέσιμους υποψήφιους για τις θέσεις εργασίας που διαχειρίζονται οι ίδιοι. Έτσι, η διαδικασία της πρόσληψης ενός νέου εργαζομένου γίνεται ευκολότερη.

Ο σκοπός αυτός επιτυγχάνεται με ένα Web Application, το JobSearch στο οποίο οι χρήστες δημιουργούν ένα λογαριασμό και δηλώνονται είτε ως υποψήφιοι για εργασία (candidates) είτε ως εργοδότες (employers) και στη συνέχεια μπορούν να έρθουν σε επικοινωνία μέσω μηνυμάτων.

Όταν κάποιος υποψήφιος γίνει αποδεκτός από κάποιον εργοδότη, παραμένει στην πλατφόρμα ως εργαζόμενος (employed) μέχρι να σταματήσει να εργάζεται στη συγκεκριμένη δουλειά όπου και ξαναγίνεται υποψήφιος.

Για την υλοποίηση αυτής της εφαρμογής χρησιμοποιήθηκε η βιβλιοθήκη React της JavaScript για το Front-End, Node.js και το framework Express.js για το Back-End, Socket.io για το σύστημα των μηνυμάτων και MySQL για την βάση δεδομένων.

Η εφαρμογή είναι λειτουργική και μπορεί να συμβάλλει θετικά στην προσπάθεια πολλών ανθρώπων να βρούν μια εργασία όπου να ταιριάζει στα ενδιαφέροντά τους, στην εκπαίδευσή τους, στον τόπο διαμονής τους αλλά και στο χρόνο που μπορούν να διαθέσουν.

Abstract

This work aims to help people looking for work find available jobs to which they can apply. In addition, employers can also find available candidates for the jobs they manage themselves. Thus, the process of hiring a new employee becomes easier.

This purpose is implemented with a Web Application, JobSearch, in which users create an account and register either as candidates for work (candidates) or as employers (employers) and then they can communicate via messages.

When a candidate is accepted by an employer, he remains on the platform as an employee (employed) until he stops working in the specific job at which point he becomes a candidate again.

For the implementation of this application, the React JavaScript library was used for the Front-End, Node.js and the Express.js framework for the Back-End, Socket.io for the messaging system and MySQL for the database.

The application is functional and can contribute positively to the efforts of many people to find a job that suits their interests, their education, their place of residence, and the time they have available.

Ευχαριστίες

Θα ήθελα να ευχαριστήσω πολύ όλους τους ανθρώπους που με στήριξαν καθ’ όλη την διάρκεια των σπουδών μου και με βοήθησαν να εξελιχθώ και να περάσω υπέροχα φοιτητικά χρόνια.

Αρχικά, τους γονείς μου οι οποίοι με στήριζαν σε κάθε μου απόφαση και με συμβούλευαν όταν το χρειαζόμουν. Δίχως εκείνους, δεν νομίζω πως θα είχα φτάσει μέχρι εδώ τόσο εύκολα οπότε τους ευχαριστώ πολύ για όλη αυτή τους την προσπάθεια.

Θα ήθελα επίσης να ευχαριστήσω πολύ τους καθηγητές μου και ειδικά την κυρία Κουτσονικόλα η οποία με βοήθησε σε ότι χρειάστηκα σε όλο το διάστημα της πτυχιακής μου. Οι καθηγητές μου, μου έμαθαν πολλά πράγματα και μου έδωσαν τα κατάλληλα ερεθίσματα ώστε να ασχοληθώ με την επιστήμη της πληροφορικής.

Τέλος, θα ήθελα να ευχαριστήσω όλους τους φίλους μου όπου έδωσαν ένα ωραίο χρώμα στις αναμνήσεις που κρατάω από τα φοιτητικά μου χρόνια και με βοήθησαν όταν το είχα ανάγκη.

Σας ευχαριστώ πολύ όλους.

Αδάμος Αλέξανδρος

2/7/2025

Περιεχόμενα

1 Εισαγωγή

Στη σημερινή εποχή όπου η ψηφιακή τεχνολογία έχει ενταχθεί πλήρως στην καθημερινότητα των ανθρώπων, δημιουργείται η ανάγκη για την προσαρμογή των λειτουργιών μιας κοινωνίας στα σύγχρονα πρότυπα με την χρήση των νέων τεχνολογιών.

Σε αυτό το πλαίσιο, κρίνεται απαραίτητος και ο μετασχηματισμός του τρόπου όπου τα άτομα εργάζονται αλλά και που ψάχνουν για εργασία. Τους δίνεται η δυνατότητα να εκμεταλλευτούν τα πλεονεκτήματα που δίνει η σύγχρονη τεχνολογία ώστε να καταλήξουν σε συμπεράσματα πιο σύντομα, πιο βέβαια και πιο αποτελεσματικά.

Φανταστείτε ότι δεν εργάζεστε την τρέχουσα χρονική περίοδο και ψάχνετε την επόμενή σας επαγγελματική απασχόληση. Δεν θα ήταν πολύ πιο εύκολη αυτή η αναζήτηση αν όλες οι διαθέσιμες θέσεις εργασίας ήταν συγκεντρωμένες σε μια ψηφιακή πλατφόρμα με όλες τις απαραίτητες πληροφορίες όπου θα μπορείτε να έρθετε απευθείας σε επικοινωνία με τους εργοδότες ώστε να καταλήξετε σε μια συμφωνία;

Πέρα από αυτό όμως, υπάρχει και ένας άλλος τρόπος να βελτιωθεί η διαδικασία προσλήψεων. Στο μοντέλο που περιγράφτηκε παραπάνω, ο εργοδότης μπορεί μόνο να περιμένει μέχρι κάποιος υποψήφιος να κάνει αίτηση ώστε να ξεκινήσει η επικοινωνία μεταξύ τους. Όμως θα ήταν καλύτερο και ο εργοδότης με τη σειρά του να μπορεί και εκείνος να προσεγγίσει έναν υποψήφιο πρώτος χωρίς να χρειάζεται να περιμένει να κάνει ο υποψήφιος την αίτηση.

Έτσι λοιπόν, δημιουργείται ένα περιβάλλον όπου και οι 2 πλευρές έχουν τη δυνατότητα να συμβάλλουν έμπρακτα στη διαδικασία της πρόσληψης και επιτυγχάνεται η καλύτερη σύνδεση μεταξύ τους. Αυτό ακριβώς το περιβάλλον προσπαθεί να δημιουργήσει η συγκεκριμένη εφαρμογή JobSearch όπου αναπτύχθηκε στα πλαίσια αυτής της πτυχιακή εργασίας.

Οι χρήστες όταν ανοίγουν την εφαρμογή για πρώτη φορά, καλούνται να δημιουργήσουν ένα λογαριασμό όπου παραθέτουν κάποιες πληροφορίες για εκείνους οι οποίες θα χρησιμοποιηθούν στο προφίλ τους. Επιπλέον, εκεί προσδιορίζουν τον ρόλο τους (υποψήφιος η εργοδότης).

Οι υποψήφιοι στη συνέχεια μπορούν να πλοηγηθούν στην εφαρμογή και να αναζητήσουν τις θέσεις εργασίας που επιθυμούν με τη χρήση φίλτρων. Όταν επιλέξουν, μπορούν να κάνουν κατευθείαν αίτηση σε αυτή, να την αποθηκεύσουν για μελλοντική προβολή, να στείλουν μήνυμα στον εργοδότη αλλά και να δουν περισσότερες πληροφορίες για την εταιρία και τη συγκεκριμένη δουλειά πριν αποφασίσουν να κάνουν κάποια ενέργεια.

Αντίστοιχα οι εργοδότες μπορούν να αναζητήσουν τους υποψηφίους με τα κριτήρια που επιθυμούν, δουν πληροφορίες για αυτούς όπως το βιογραφικό τους, την εκπαίδευσή τους και άλλα και να επικοινωνήσουν μαζί τους ώστε να δουν αν είναι κατάλληλοι για τη συγκεκριμένη θέση εργασίας.

Όσοι υποψήφιοι τελικά καταλήξουν σε συμφωνία με τους εργοδότες, γίνονται υπάλληλοι (employed) και μπορούν να εξερευνήσουν και νέες επιλογές θέσεων.

Μια ακόμα λειτουργία της εφαρμογής είναι το σύστημα των κριτικών. Κάθε εργαζόμενος η εργοδότης μπορεί να κάνει μια κριτική στην εταιρία που εργάζεται η στον εργαζόμενο που έχει στην θέση που διαχειρίζεται. Αυτό συμβάλλει στην καλύτερη αξιολόγησή τους.

Επιπλέον, στην εφαρμογή υπάρχει και ένα NLP σύστημα προτάσεων (recommendation system) το οποίο χρησιμοποιώντας στοιχεία από το προφίλ του κάθε χρήστη κάνει κάποιες προτάσεις ανάλογα με τον ρόλο του ατόμου. Σε υποψήφιους προτείνει δουλειές και σε εργοδότες προτείνει υποψηφίους.

Προτεινόμενη Διάρθρωση Πτυχιακής

Στα επόμενα κεφάλαια αναλύεται η σημαντικότητα της εφαρμογής μέσω σύγκρισης με ήδη υπάρχουσες εφαρμογές, οι τεχνολογίες που χρησιμοποιήθηκαν, η αρχιτεκτονική της, η χρήση της αλλά και τα συμπεράσματα και οι μελλοντικές επεκτάσεις της.

2 Τρέχουσα κατάσταση

Αυτή τη στιγμή υπάρχουν και αρκετές άλλες εφαρμογές οι οποίες συμβάλλουν στη διευκόλυνση της εύρεσης εργασίας. Παρακάτω θα παραθέσω μερικές και θα γίνει μια σύγκριση σε επίπεδο λειτουργιών με την εφαρμογή που παρουσιάζεται σε αυτή την εργασία.

2.1 Εμπορικές Εφαρμογές

Οι πιο χαρακτηριστικές εφαρμογές είναι οι εμπορικές, αυτές δηλαδή που μπορεί να γνωρίζετε και χωρις να τις έχετε χρησιμοποιήσει. Ίσως η πιο γνωστή από αυτές να είναι το Linkedin.

 [1]

Το Linkedin είναι μια πλατφόρμα κοινωνικής δικτύωσης η οποία στοχεύει στο να διασυνδέσει άτομα που θέλουν να δικτυωθούν επαγγελματικά. Η συγκεκριμένη πλατφόρμα προσφέρει τη δυνατότητα δημιουργίας ενός προφίλ χρήστη μέσω του οποίου θα προβληθούν και θα αναδείξουν τις ικανότητές τους. Κάθε χρήστης που ψάχνει για εργασία μπορεί να διασυνδεθεί με άλλα άτομα, να μπεί σε groups, να δει αναρτήσεις άλλων χρηστών που τον ενδιαφέρουν να δεί εκδηλώσεις οι οποίες είναι σχετικές με τον τομέα του ενδιαφέροντός του, να ψάξει για θέσεις εργασίας και να επικοινωνήσει με τους υπεύθυνους αυτών των θέσεων. Οι εργοδότες που μπαίνουν σε αυτή την πλατφόρμα

[1] Εικόνα 1: Linkedin logo.

μπορούν και εκείνοι να κάνουν περαιτέρω επαγγελματικές διασυνδέσεις αλλά και να ψάξουν για νέους εργαζόμενους ώστε να τους εντάξουν στο εταιρικό τους δυναμικό.[1]

Η εφαρμογή που έχει αναπτυχθεί στα πλαίσια αυτής της πτυχιακής δανείζεται κάποια στοιχεία από το Linkedin, όπως τη δυνατότητα προσαρμογής ενός προφίλ του χρήστη που λειτουργεί σαν μια έκθεση των σημαντικότερων προτερημάτων του, την δυνατότητα επικοινωνίας μέσω μηνυμάτων μεταξύ των 2 τύπων χρηστών αλλά και τη λογική του matchmaking μεταξύ τους.

Η κύρια διαφορά του Linkedin με την συγκεκριμένη εφαρμογή είναι ότι το Linkedin λειτουργεί σαν ένα υβρίδιο κοινωνικού δικτύου και εφαρμογής για εύρεση εργασίας δίνοντας έμφαση στην επαφή μεταξύ των χρηστών ώστε να δημιουργήσουν connections όπως λέγονται εκεί, ενώ η JobSearch είναι στοχευμένη καθαρά στην διαδικασία της εύρεσης εργασίας χωρίς άλλες λειτουργίες όπως τα groups, τα posts και άλλα. Αυτό κάνει την συγκεκριμένη εφαρμογή λιγότερο πολύπλοκη στη χρήση και πιο αποτελεσματική στον τομέα-στόχο της μιας και επικεντρώνεται στην ουσία της ιδέας και του προβλήματος που προσπαθεί να επιλύσει.

Μια ακόμα εφαρμογή η οποία είναι αρκετά δημοφιλής είναι το Indeed.

indeed[2]

Το Indeed, μοιάζει πολύ περισσότερο με την JobSearch καθώς και αυτό έχει λιγότερες λειτουργίες από το Linkedin και είναι πιο στοχευμένο στην εύρεση εργασίας. Στο Indeed οι χρήστες όταν φτιάχνουν τον λογαριασμό τους, διαλέγουν αν θέλουν να ψάξουν για εργασία η αν θέλουν να δημιοσιεύσουν μια θέση εργασίας ως εργοδότες. Οι υποψήφιοι στη συνέχεια μπορούν να ανεβάσουν ένα βιογραφικό ώστε να μπορούν να τους αναζητήσουν και οι εργοδότες και να δούν πληροφορίες για εκείνους. Επιπλέον μπορούν να απαντήσουν σε μερικές πολύ σύντομες ερωτήσεις ώστε η εφαρμογή να τους προτείνει θέσεις εργασίας με βάση τις προτιμήσεις τους. Τέλος, μπορούν να κάνουν κριτικές στις εταιρίες τις οποίες εργάζονται ώστε να τις δουν οι υποψήφιοι πριν κάνουν μια αίτηση. [2]

Μια ακόμη εφαρμογή η οποία μοιάζει με το JobSearch είναι το FlexJobs.

[2] Εικόνα 2: Indeed logo.

[3]

Το FlexJobs σαν χαρακτηριστικό του έχει ότι είναι μια πλατφόρμα για remote jobs, δηλαδή εργασία εξ' αποστάσεως. Αυτό είναι καλό για μια τέτοια πλατφόρμα, καθώς δίνει τη δυνατότητα για περισσότερη ανάπτυξη της διαδικασίας πρόσληψης μέσα στην πλατφόρμα διότι οι υποψήφιοι δεν χρειάζεται να κάνουν κάτι εκτός της πλατφόρμας για να έρθουν σε συμφωνία με τους εργοδότες. Η συγκεκριμένη πλατφόρμα επίσης έχει αρκετά άτομα προσωπικό που κάνουν διαλογή στις θέσεις εργασίας που βάζουν οι εργοδότες ώστε να τις φιλτράρει και να αποφευχθούν οι απάτες. Είναι από τις καλύτερες πλατφόρμες σε αυτόν τον τομέα. Η πλατφόρμα προσφέρει πιο ειδικά φίλτρα αναζήτησης καθώς και έναν μηχανισμό τύπου Save Job όπως και το JobSearch. [3]

Μια ακόμα πλατφόρμα που έχει κοινά χαρακτηριστικά με το JobSearch είναι το Job.com.

[4]

Το χαρακτηριστικό αυτής της πλατφόρμας είναι ότι για να ταιριάξει τους employers με τους candidates χρησιμοποιεί αλγορίθμους τεχνητής νοημοσύνης και μηχανικής μάθησης. Με αυτό τον τρόπο μπορούν να επεξεργαστούν καλύτερα τα διαθέσιμα δεδομένα για τους candidates και να γίνουν καλύτερες προτάσεις στους employers. Στο JobSearch το σύστημα των προτάσεων γίνεται με ένα πολύ απλό NLP μοντέλο και ενώ

[3] Εικόνα 3: Flexjobs Logo
[4] Εικόνα 4: Job.com Logo

δεν είναι όσο advanced είναι του Job.com, ακολουθεί την ίδια λογική και παράγει ένα ικανοποιητικό αποτέλεσμα. [4]

3 Τεχνολογίες

Στην υλοποίηση της εφαρμογής χρησιμοποιήθηκαν αρκετές τεχνολογίες. Εφόσον και γίνεται λόγος για Web εφαρμογή, χρειάστηκαν λύσεις για Front-End, Back-End αλλά και για την database.

3.1 Front-End

Αρχικά θα παρουσιαστούν οι τεχνολογίες του Front-End. Το βασικό UI και η κωδικοποίηση της σελίδας υλοποιήθηκαν στην React.js που είναι μια βιβλιοθήκη της Javascript για δημιουργία Front-End User Interface σε διακριτά κομμάτια που ονομάζονται Components. Χρησιμοποιήθηκε η React γιατί ειναι μια από τις πιο δημοφιλείς επιλογές για Front End development και υπάρχουν πολλές επιτυχημένες εφαρμογές που την χρησιμοποιούν όπως το Facebook, το Instagram, το Whatsapp και άλλες. [5] Μάλιστα, η Meta η εταιρία που έχει το Facebook και το Instagram διαχειρίζεται την React και τις νέες εκδόσεις της μαζί με κάποιους ιδιώτες developers και εταιρίες. [6] Οι λόγοι που τόσοι πολλοί developers χρησιμοποιούν την React είναι ότι επιτρέπει τη δημιουργία Javascript αντικειμένων (components) εύκολα και αποτελεσματικά. [5] Το render των αντικειμένων αυτών το κάνει μέσω VirtualDOM που είναι ένα δεύτερο εικονικό DOM που η React χρησιμοποιεί για να βρεί τον βέλτιστο δυνατό τρόπο να κάνει render το κανονικό DOM. [7] Επίσης, η ευελιξία της με τα Components και την δυνατότητα επαναχρησιμοποίησής τους την κάνει πολύ καλή και για large-scale εφαρμογές και γι' αυτό και επιλέγεται και από τόσες εταιρίες για μεγάλα projects.

Εκτός από την React για το σύστημα των μηνυμάτων χρησιμοποιήθηκε το Socket.io που είναι μια event-driven βιβλιοθήκη της Javascript που επιτρέπει την επικοινωνία μεταξύ ενος client και ενός server (στην προκειμένη περίπτωση, το Node.js). Στο Front-End διαχειρίζεται τη σύνδεση του WebSocket, την αποστολή των μηνυμάτων στο Back-End αλλά και το φόρτωμα των μηνυμάτων ώστε να φαίνονται στην εφαρμογή. [8]

Επιπλέον, όταν ο χρήστης κάνει login, δημιουργείται και αποθηκεύεται στο local storage της εφαρμογής ένα JWT (JSON Web Token) με τα στοιχεία του χρήστη το οποίο διαρκεί

24 ώρες. Όταν ο χρήστης ανοίγει την εφαρμογή, αυτή ελέγχει αν υπάρχει το web token στο storage πριν του ζητήσει να ξανακάνει Login. Με αυτό τον τρόπο ο χρήστης παραμένει για 24 ώρες συνδεδεμένος στην εφαρμογή, έχοντας πρόσβαση στις σελίδες της εφαρμογής, χωρίς να χρειάζεται να ξανασυνδεθεί στην εφαρμογή για αυτό το χρονικό διάστημα. [9]

3.2 Back-End

Στο Back-End η σημαντικότερη τεχνολογία που χρησιμοποιείται είναι το Node.js το οποίο είναι το περιβάλλον στο οποίο λειτουργεί ο server της εφαρμογής, δηλαδή η διασύνδεση μεταξύ της βάσης δεδομένων και της εφαρμογής. [10] Το Node.js είναι όπως και η React αντίστοιχα για το Front-End είναι μια πολύ δημοφιλής επιλογή ανάπτυξης Back-End λειτουργιών όπως ένας server. Κάποιες από τις πιο γνωστές εφαρμογές που χρησιμοποιούν το Node.js είναι η PayPal, το Linkedin, το Netflix και το Uber. [11] Οι εφαρμογές που το έχουν εμπιστευτεί για τη δημιουργία του Back-End τους το έχουν κάνει λόγω της ταχύτητάς του σε σχέση με άλλες επιλογές, λόγω της ικανότητάς του να λειτουργεί εξίσου καλα σε μικρού μεγέθους και μεγάλου μεγέθους εφαρμογές αλλά και λόγω του ότι δεν είναι ιδιαίτερα απαιτητικό σε υπολογιστικούς πόρους. [10]

Επιπλέον, χρησιμοποιήθηκε το framework Express.js του Node.js το οποίο είναι το πιο δημοφιλές framework για τη δημιουργία RESTful APIs με το Node.js. Απλοποιεί αρκετά τις διαδικασίες του Node και έτσι μειώθηκε ο χρόνος που χρειάστηκε το development του Back-End. [12] Με αυτό και το Node δημιουργήθηκαν τα CRUD Operations του Back-End που επιτρέπουν την επικοινωνία μεταξύ της βάσης δεδομένων και του client. [13]

Για το σύστηματα των μηνυμάτων στο Back-End χρησιμοποιήθηκε το Socket.io με το οποίο δημιουργήθηκε το Socket.io server στον οποίο στέλνονται τα μηνύματα από το Front-End, ο server τα βάζει στη βάση δεδομένων και τα στέλνει στους παραλήπτες. Επίσης καταγράφει τους χρήστες που είναι online και τους συνδέει μέσω του WebSocket. [8]

Στο Back-End επίσης, ο server ελέγχει το JSON Web Token που έχει δημιουργηθεί ώστε να το επαληθεύσει και να βεβαιωθεί ότι είναι αυθεντικό. Αυτό γινέται στο Login POST route των users χρησιμοποιώντας το JWT Secret Key. Για περισσότερη ασφάλεια επίσης, χρησιμοποιείται η βιβλιοθήκη bcrypt.js με την οποία δημιουργήθηκε μια

συνάρτηση για hashing και salting του password, ώστε να αποθηκεύεται με πιο ασφαλή τρόπο στη βάση δεδομένων. [9]

Τέλος, στο Back-End υπάρχει και ένα σύστημα NLP τεχνητής νοημοσύνης που κάνει προτάσεις υποψηφίων στους εργοδότες και δουλειές στους υποψηφίους. Χρησιμοποιείται η βιβλιοθήκη TensorflowJS και το μοντέλο Universal Sentence Encoder. Ο τρόπος που κάνει τις προτάσεις στους χρήστες είναι με το να παίρνει τους υποψηφίους και τις θέσεις εργασίας και να φτιάχνει ένα κείμενο για το καθένα με τις πληροφορίες που έχουν βάλει στην πλατφόρμα. Στη συνέχεια συγκρίνει αυτά τα δύο κείμενα και βγάζει ένα σκόρ ανάλογα με το πόσο ομοιότητα έχουν. Το μοντέλο αυτό δεν κάνει απλή σύγκριση των λέξεων, καταλαβαίνει και το νόημά τους και έτσι τις συγκρίνει με μεγαλύτερη ακρίβεια. Τα σκόρ που βγάζει είναι από 0-10, όμως προτάσεις με σκορ 0 παραλείπονται και δεν χρησιμοποιούνται.

3.3 Βάση Δεδομένων

Για την βάση δεδομένων χρησιμοποιήθηκε MySQL. Η MySQL είναι ένα σύστημα διαχείρισης βάσεων δεδομένων σχεσιακού μοντέλου. [14] Είναι από τις πιο δημοφιλείς επιλογές καθώς χρησιμοποιείται από το Facebook, το Booking.com και άλλα. [15] Αυτό σημαίνει πως τα δεδομένα είναι οργανωμένα σε πίνακες (σχέσεις) οι οποίοι συνδέονται μεταξύ τους κάνοντας τα δεδομένα πιο εύκολα διαχειρίσιμα. [16] Η MySQL μπήκε στο πρόγραμμα μέσω του WAMP Server που είναι ένα solution stack που έχει εγκατεστημένη την MySQL. Έτσι το Back-End στέλνει τις CRUD εντολές στο WAMP και παίρνει δεδομένα από την database που έχει δημιουργηθεί εκεί. [17]

3.4 Άλλες Παρατηρήσεις

Για την ανάπτυξη της εφαρμογής χρησιμοποιήθηκαν και κάποιες εφαρμογές τεχνητής νοημοσύνης που βοήθησαν στην συγγραφή του κώδικα. Από αυτές, κυρίως χρησιμοποιήθηκε η εφαρμογή ChatGPT και σε μικρότερο επίπεδο το Github Copilot και το TabNine AI.

Οι τεχνολογίες που χρησιμοποιήθηκαν για το Front-End και το Back-End είναι μέρος των πιο δημοφιλών Web Development stacks, όπως το MERN, το MEAN και άλλα. Ένα από τα σημαντικά πλεονεκτήματα αυτών των Stacks και συνεπώς και της συγκεκριμένης εφαρμογής είναι ότι και το Front και το Back End είναι υλοποιημένα σε Javascript. Αυτό

βοηθάει στην καλύτερη επικοινωνία μεταξύ τους αλλά και επιπλέον έκανε το development της εφαρμογής πιο ομαλό μιας και δεν χρειάστηκαν άλλες γλώσσες.

4 Αρχιτεκτονική Εφαρμογής

Αφού αναλύθηκαν οι τεχνολογίες που χρησιμοποιήθηκαν, τώρα θα παρατεθεί η αρχιτεκτονική της εφαρμογής, δηλαδή ο τρόπος που οι σελίδες της επικοινωνούν μεταξύ τους και ανταλλάσσουν δεδομένα. Επίσης, θα εξηγηθεί και η αρχιτεκτονική της βάσης δεδομένων.

4.1 Επικοινωνία σελίδων

Το κομμάτι της επικοινωνίας των σελίδων αναφέρεται στο πως οι σελίδες της web εφαρμογής επιτρέπουν την πλοήγηση από τη μια στην άλλη αλλά και το πως ανταλλάσσουν δεδομένα ώστε το interface να εμφανίζει τα σωστά αποτελέσματα. Η πλοήγηση στις σελίδες στην React υλοποιείται μέσω του useNavigate() hook και του React Router (react-router-dom package). Το React Router είναι μια βιβλιοθήκη για την δημιουργία routes, δηλαδή μέσων πλοήγησης στην εφαρμογή χωρίς την ανάγκη για refresh στην σελίδα, δημιουργώντας έτσι μια Single-page εφαρμογή. Σε αυτά τα routes κουμπώνει το Link, που είναι μια βελτιωμένη έκδοση του <a href> της HTML και το useNavigate hook. Τα hooks στην React είναι συναρτήσεις που επιτρέπουν την είσοδο και τις αλλαγές στην εσωτερική μνήμη της (State) και σε άλλες λειτουργίες της. Το useNavigate hook συγκεκριμένα χρησιμοποιεί τα Routes που έχουν δηλωθεί μέσω του React Router και έτσι επιτρέπει την πλοήγηση στα routes μέσω buttons η και άλλων προγραμματιστικών εργαλείων.

Τα routes στα οποία μπορεί να πλοηγηθεί ο χρήστης ανά πάσα χρονική στιγμή διαφέρουν ανάλογα με το role του user. Όπως φαίνεται στις παρακάτω εικόνες, το navbar για τους candidates είναι διαφορετικό σε σχέση με των employers καθώς οι candidates βλέπουν links προς το route Jobs και οι employers προς το route Candidates.

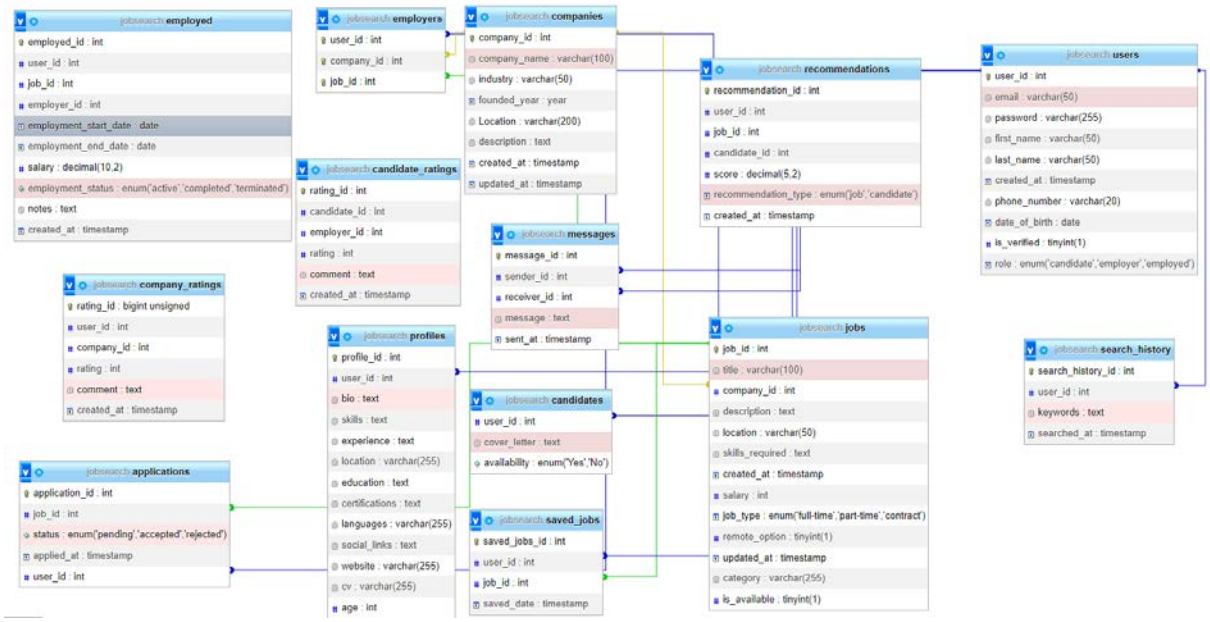
Επιπλέον υπάρχει στην εφαρμογή η λογική του Conditional Rendering. Αυτό σημαίνει πως κάποιες σελίδες έχουν διαφορετική συμπεριφορά ανάλογα με τα δεδομένα του χρήστη που παίρνουν από την βάση δεδομένων. Ένα παράδειγμα για αυτό είναι το Home Page της εφαρμογής το οποίο αλλάζει ανάλογα με το role του user αλλά και με το

αν είναι Logged In η όχι. Κατά συνέπεια, εφόσον αλλάζει το περιεχόμενο των σελιδών, αλλάζουν και τα links μεταξύ τους.

4.2 Αρχιτεκτονική Βάσης Δεδομένων

Η βάση δεδομένων είναι υλοποιημένη σε MySQL. Οι πίνακες είναι στη μορφή InnoDB. Παρακάτω φαίνονται κάποιοι ενδεικτικοί πίνακες ώστε να παρουσιαστεί η αρχιτεκτονική τους:

Για να αναδειχθεί καλύτερα η αρχιτεκτονική της βάσης δεδομένων, θα χρειαστεί το ERD. Το ERD η αλλιώς Entity-Relation Diagram είναι ένα διάγραμμα που δείχνει τις οντότητες και τις σχέσεις μεταξύ τους σε μια βάση δεδομένων.



[5]

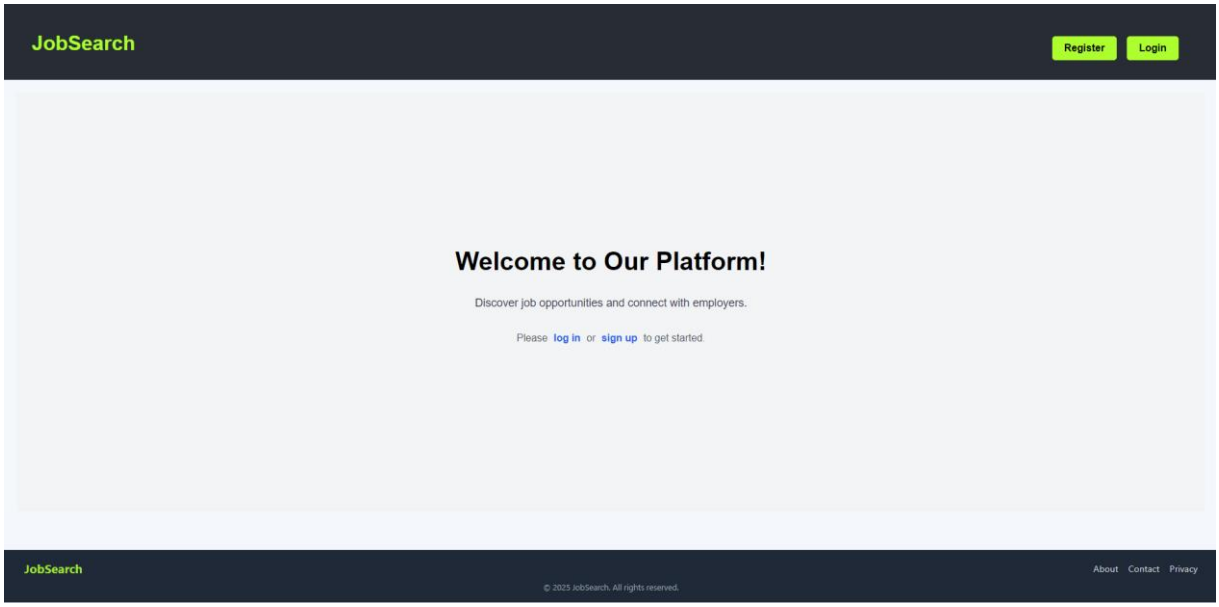
[5] Εικόνα 5: Διάγραμμα Σχέσεων-Οντοτήτων της βάσης δεδομένων

4.3 Επικοινωνία σελίδων με βάση δεδομένων

Η επικοινωνία μεταξύ του Front-End και της βάσης δεδομένων γίνεται με τη βοήθεια ενός RESTful API. Αυτό δίνει κάποιες εντολές στη βάση δεδομένων, συγκεκριμένα Create, Read, Update, Delete οι οποίες κάνουν αλλαγές στα δεδομένα και τα στέλνουν στο Front-End ώστε να προβληθούν. Οι εντολές Create γίνονται με το endpoint POST, οι Read με το GET, οι UPDATE με το PUT και οι Delete με το DELETE.

5 Η εφαρμογή

Η εφαρμογή JobSearch, όταν ο χρήστης μπαίνει για πρώτη φορά, τον στέλνει στη σελίδα HomeNotLoggedIn όπου είναι το Home Page για τους χρήστες οι οποίοι δεν έχουν κάνει Login στην εφαρμογή.



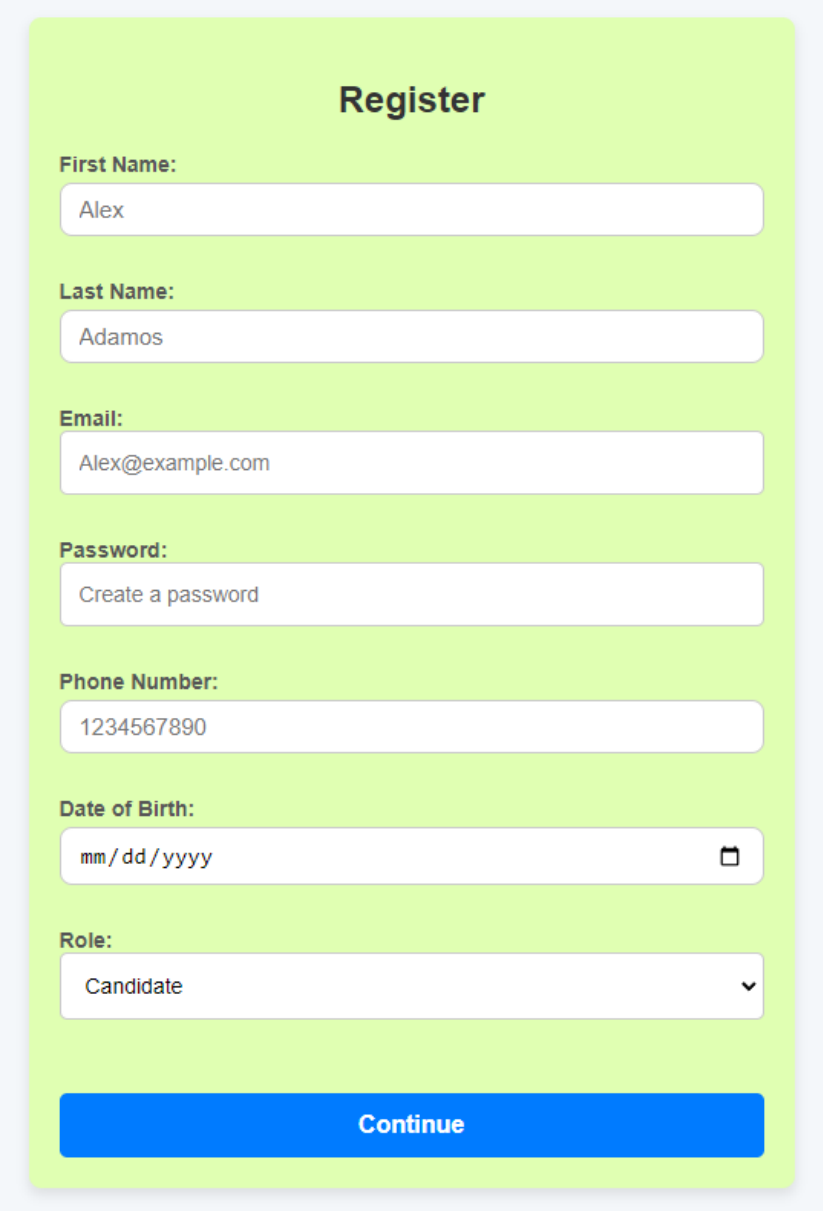
6

Στο Header της εφαρμογής υπάρχει στα αριστερά το Logo και στα δεξιά ένα κουμπί για Register, δηλαδή δημιουργία λογαριασμό στην εφαρμογή και ένα για Login, δηλαδή σύνδεση στην εφαρμογή με λογαριασμό που έχει δημιουργηθεί. Στο footer φαίνεται ξανα το Logo στα αριστερά και ένα Copyright text στη μέση του.

Το HomeNotLoggedIn page περιέχει ένα μήνυμα καλωσορίσματος προς τους χρήστες της εφαρμογής και τους προτρέπει να κάνουν sign up (register) η log in στην εφαρμογή.

[6] Εικόνα 6: Αρχική σελίδα (Home Page)

Η φόρμα register της εφαρμογής αρχικά ξεκινάει έτσι:



[7]

Ο χρήστης καλείται να βάλει το όνομα και το επίθετό του, το email του, τον κωδικό πρόσβασής του, το τηλέφωνό του, την ημερομηνία γέννησής του και τον ρόλο του. Έπειτα ανάλογα με τον ρόλο που θα επιλέξει ο user του δίνει και την ανάλογη συνέχεια της φόρμας ώστε να συμπληρώσει και τα υπόλοιπα στοιχεία του προφίλ τους.

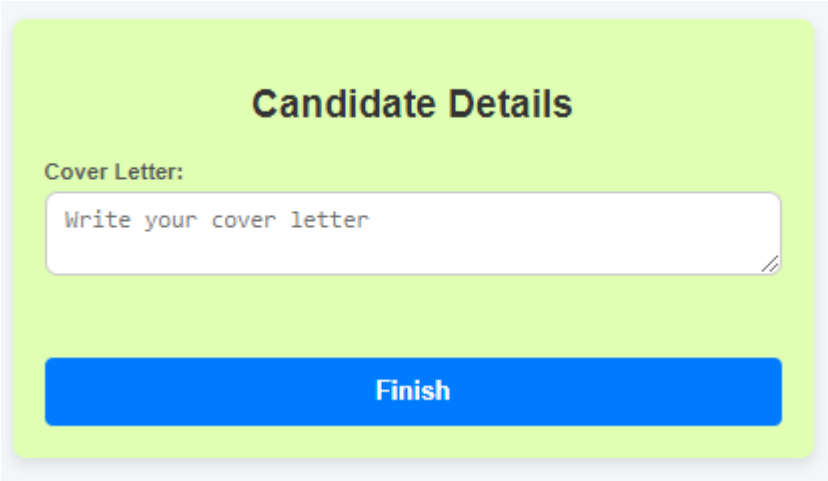
[7] Εικόνα 7: Φόρμα εγγραφής

Το register για τους candidates συνεχίζει με αυτό το form όπου ο χρήστης συμπληρώνει την ηλικία του, μια σύντομη περιγραφή του βιογραφικού του, τις δεξιότητές του, την εργασιακή του εμπειρία, την τοποθεσία του, την εκπαίδευσή του, τις πιστοποιήσεις τους, τις γλώσσες που γνωρίζει, το link του βιογραφικού του, τα links των κοινωνικών δικτύων του αλλά και το site-portfolio του:

[8]

Στη συνέχεια, όταν πατήσει το next, τον προτρέπει να συμπληρώσει και το cover letter:

[8] Εικόνα 8: Συνέχεια φόρμας εγγραφής

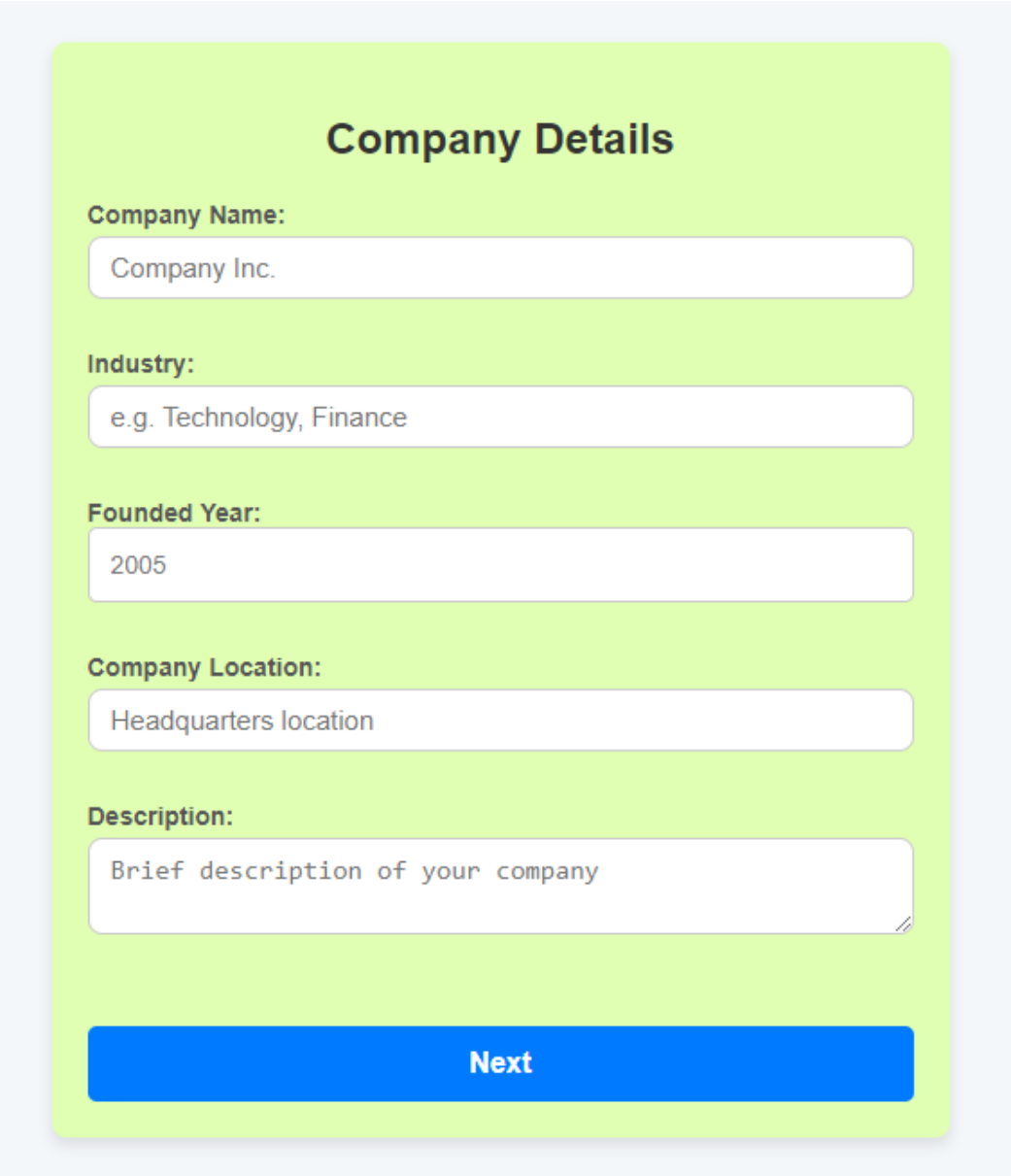


[9]

και ύστερα έχει ολοκληρώσει την διαδικασία της εγγραφής του.

Εαν ο user επιλέξει role employer, μετά τη φόρμα για το προφίλ, καλείται να βάλει και στοιχεία για την εταιρία του αλλά και για την δουλειά για την οποία ψάχνει εργαζόμενους. Όσον αφορά την εταιρία οι πληροφορίες που ζητούνται είναι το όνομά της, ο κλάδος στον οποίο δραστηριοποιείται, η χρονολογία ίδρυσής της, η τοποθεσία της και μια περιγραφή της δραστηριότητάς της ώστε να τη δεί ο υποψήφιος και να τον κάνει να ενδιαφερθεί.

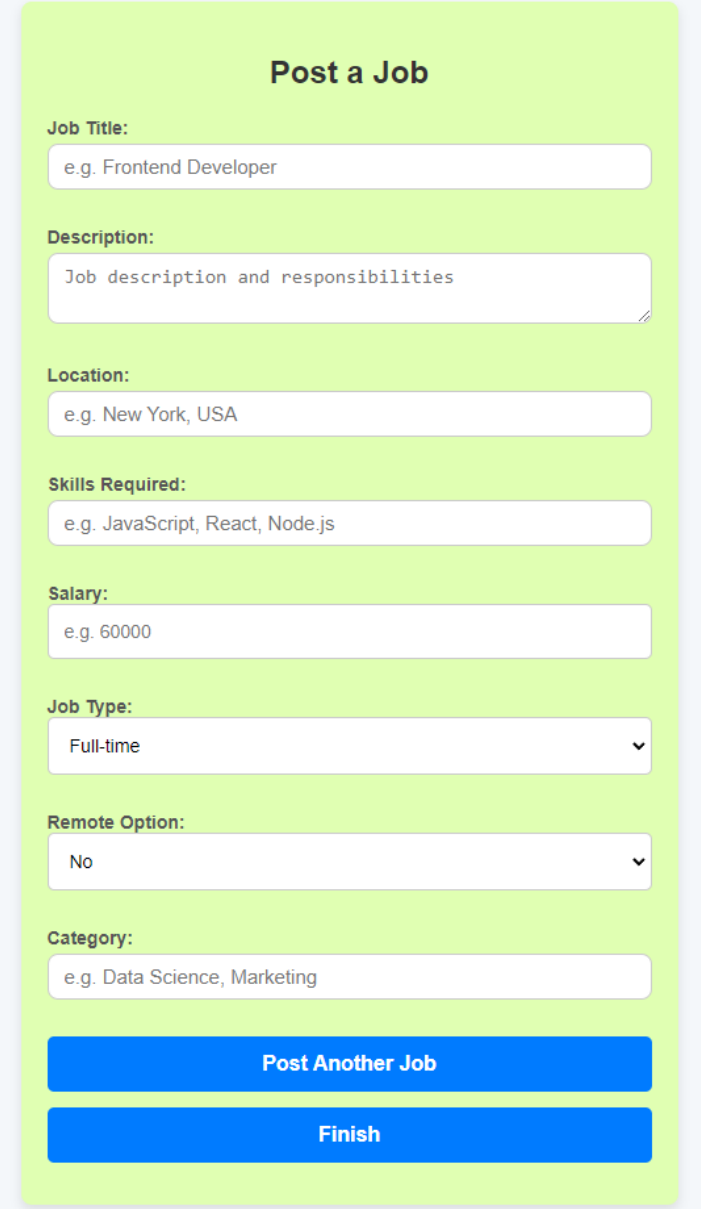
[9] Εικόνα 9: Φόρμα εγγραφής υποψηφίων



[10]

Και τα στοιχεία της δουλειάς, ο τίτλος της, μια σύντομη περιγραφή, η τοποθεσία της, οι απαραίτητες γνώσεις, ο μισθός της, ο τύπος της, η δυνατότητα εργασίας από απόσταση και η κατηγορία της η οποία χρησιμοποιείται στην αναζήτηση και στα φίλτρα:

[10] Εικόνα 10: Δήλωση στοιχείων εταιρίας στη φόρμα εγγραφής των εργοδοτών

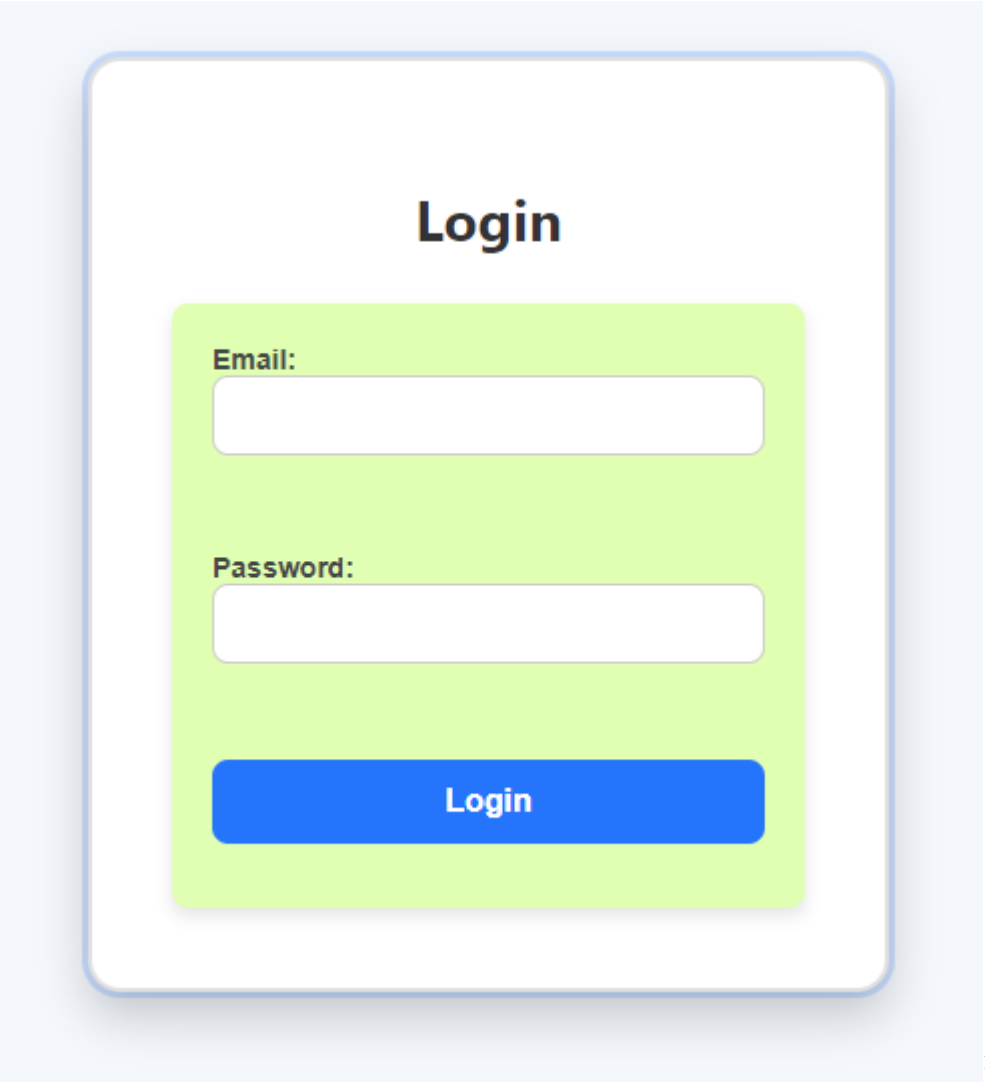


[11]

Ο χρήστης μπορεί επίσης πατώντας το κουμπί Post Another Job να αναρτήσει και μια δεύτερη θέση εργασίας στην πλατφόρμα. Μετά από αυτό το βήμα της δημιουργίας του Job listing, όταν πατήσει το finish έχει ολοκληρώσει την εγγραφή του. Αμέσως μετά την εγγραφή του χρήστη, ξεκινάει η διαδικασία της δημιουργίας των προτάσεων για τον χρήστη. Αυτή η διαδικασία είναι background task και χρειάζεται κάποιο χρόνο μέχρι να δημιουργηθούν όλες οι προτάσεις.

[11] Εικόνα 11: Δήλωση θέσης εργασίας στη φόρμα εγγραφής των εργοδοτών

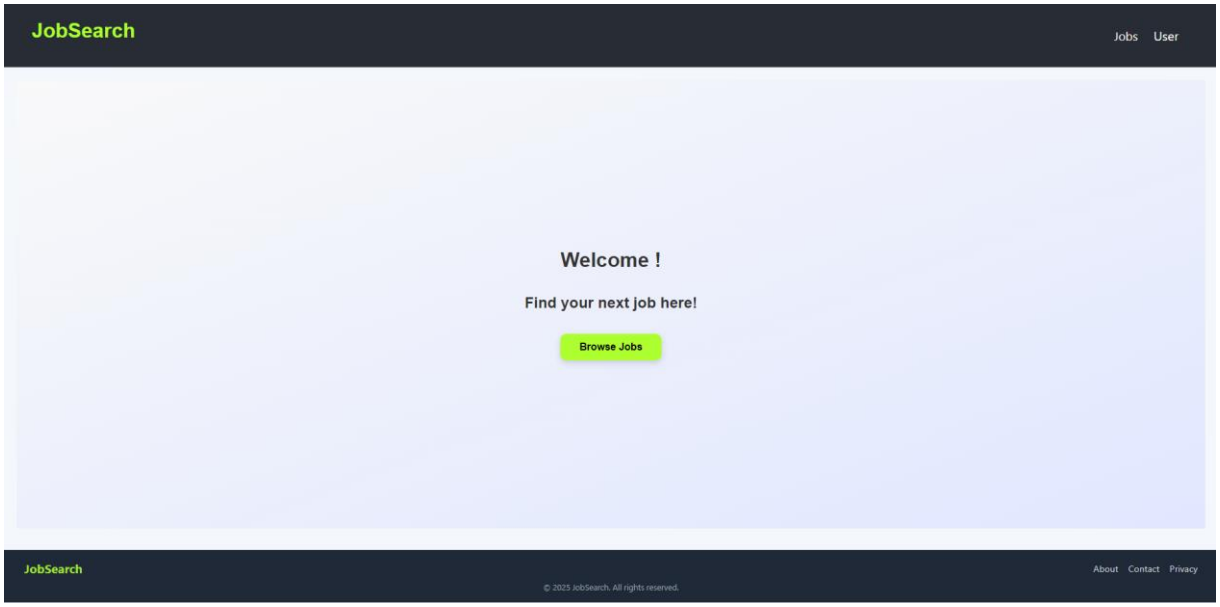
Το Login της εφαρμογής:



[12]

[12] Φόρμα σύνδεσης στην πλατφόρμα

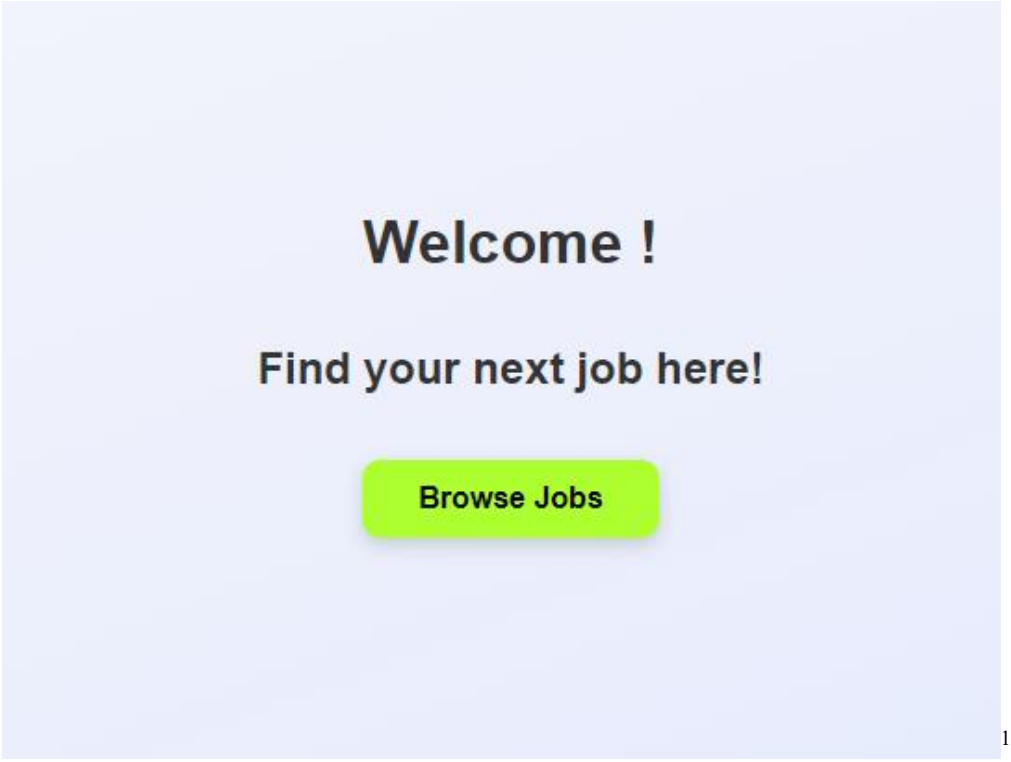
Όταν ο χρήστης κάνει εγγραφή η login στην εφαρμογή με επιτυχία, θα τον κατευθύνει η εφαρμογή στην σελίδα Home. Αυτό θα είναι και το Home page που θα του εμφανίζεται για όσο παραμείνει logged in στην εφαρμογή μέσω του web token που δημιουργείται.



13

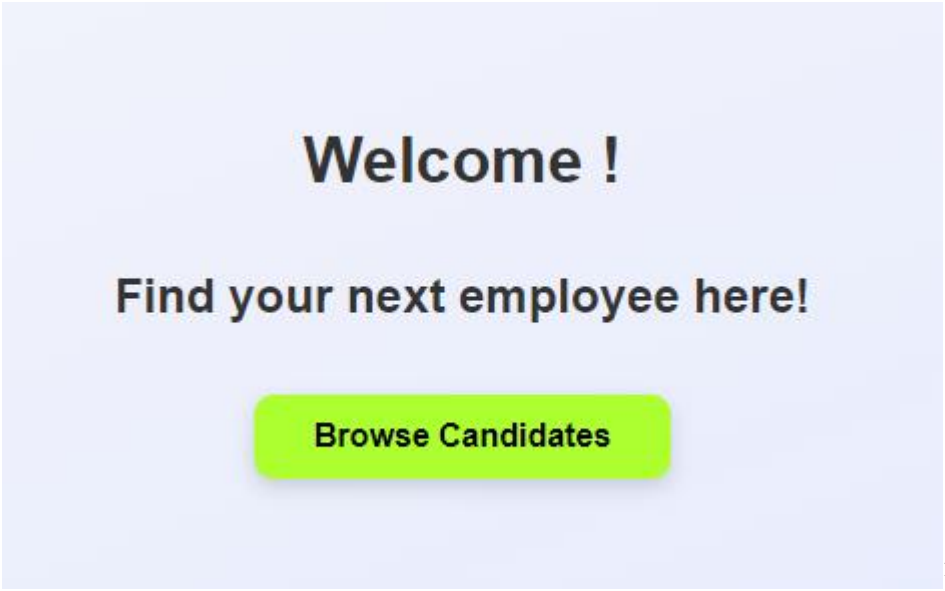
Από αυτή τη σελίδα μπορεί να πλοηγηθεί σε όλη την υπόλοιπη εφαρμογή. Ανάλογα με το role του user, του εμφανίζεται και διαφορετικό μήνυμα στο Home page. Στους candidates δείχνει κουμπί που κάνει navigate στη σελίδα Jobs

[13] Αρχική σελίδα της εφαρμογής για τους συνδεδεμένους χρήστες



[14]

ενώ στους employers δείχνει κουμπί για navigation στη σελίδα Candidates.



[15]

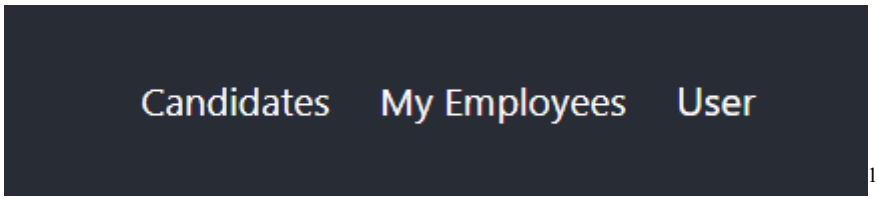
[14] Εικόνα 14: Κουμπί Browse Jobs
[15] Εικόνα 15: Κουμπί Browse Candidates

Επιπλέον, το navbar που βρίσκεται στο header έχει αλλάξει πλέον και δείχνει τις διαθέσιμες σελίδες στις οποίες μπορεί να πλοηγηθεί ο user.

Candidates navbar:

[16]

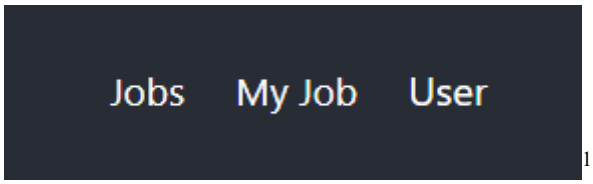
Employers navbar:



[17]

Οι employed, όσοι δηλαδή έχουν δουλειά, έχουν επίσης την σελίδα My Job.

Employed navbar:

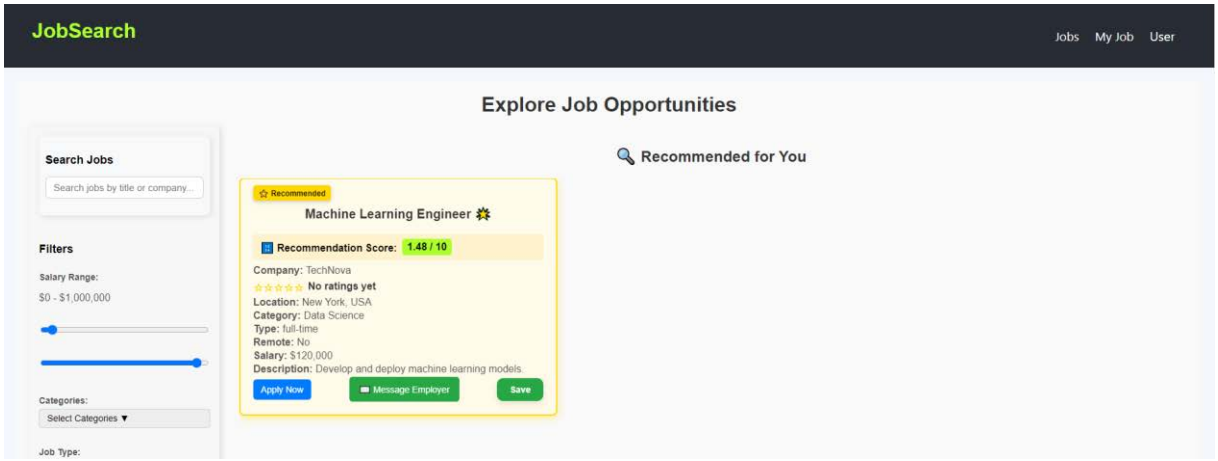


[18]

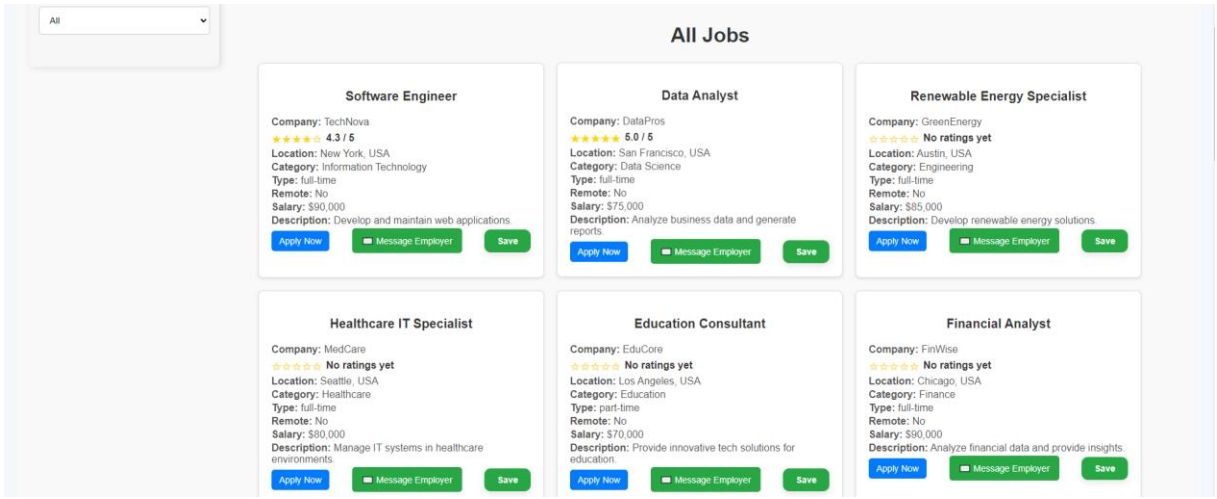
Στη σελίδα Jobs οι candidates μπορούν να δούν τις διαθέσιμες δουλειές που υπάρχουν και πληροφορίες για αυτές καθώς και να αναζητήσουν δουλειές βάσει κριτηρίων αλλά και αναζήτησης μέσω κειμένου. Επιπλέον η εφαρμογή βγάζει και προτεινόμενες δουλειές στον χρήστη με βάση τα προσόντα του.

Η σελίδα Jobs:

[16] Εικόνα 16: Candidates Navbar
[17] Εικόνα 17: Employers navbar
[18] Εικόνα 18: Employed navbar



19

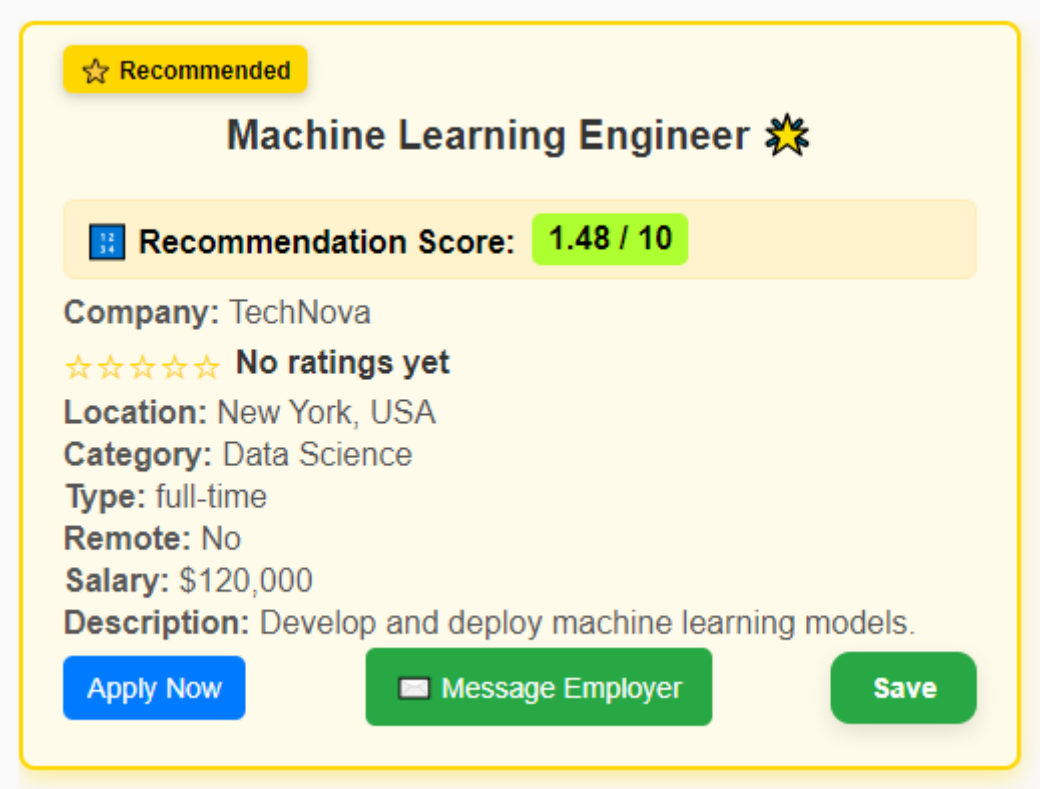


20

Όταν βρουν μια δουλειά που τους ενδιαφέρει μπορούν να κάνουν αίτηση σε αυτή τη δουλειά ώστε να τους δούν οι εργοδότες και να επικοινωνήσουν μαζί τους, μπορούν να στείλουν μήνυμα στον εργοδότη που είναι υπεύθυνος για αυτή τη θέση εργασίας, να αποθηκεύσουν την θέση εργασίας ώστε να μπορούν να την ξαναδούν σε μελλοντική στιγμή αλλά και πατωντας πάνω στο job listing να δούν περισσότερες πληροφορίες για την δουλεία, τον εργοδότη που έχει βάλει την δουλειά στην εφαρμογή και την εταιρία που έχει την δουλειά.

[19] Εικόνα 19: Σελίδα Jobs
[20] Εικόνα 20: Συνέχεια σελίδας Jobs

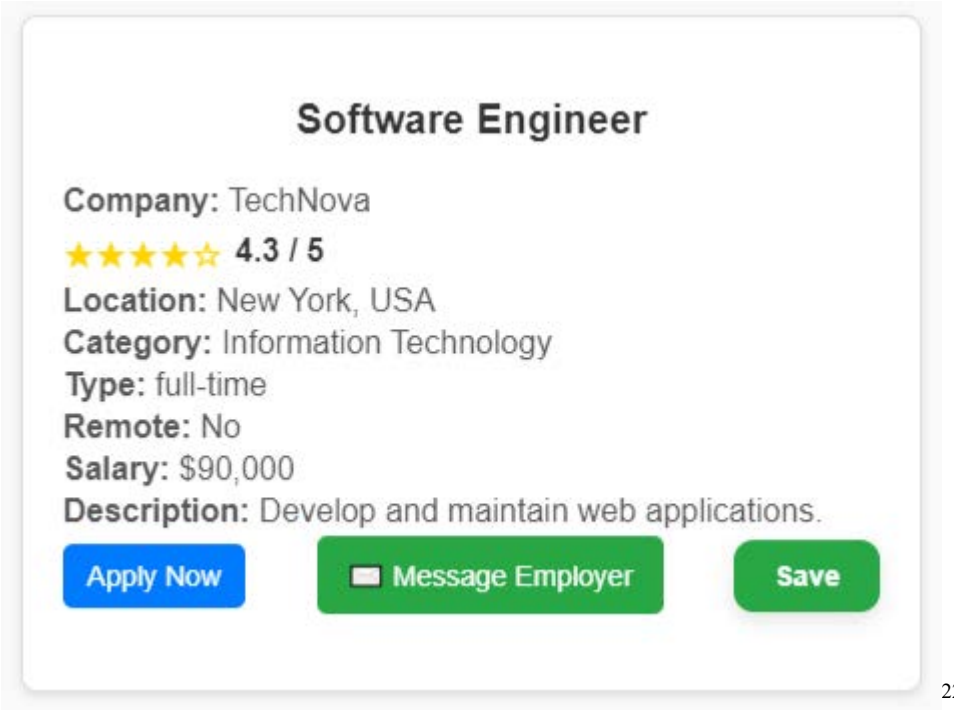
Παράδειγμα προτεινόμενης δουλειάς:



[21]

Παράδειγμα διαθέσιμης δουλειάς:

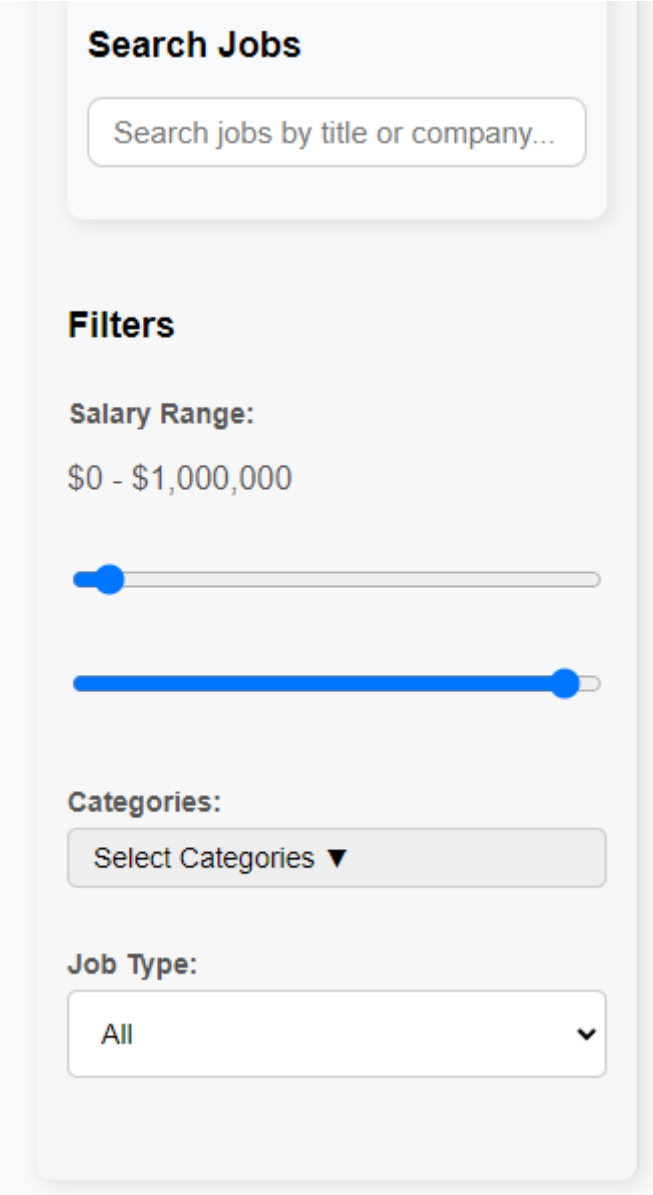
[21] Εικόνα 21: Παράδειγμα προτεινόμενης θέσης εργασίας



[22]

Τα διαθέσιμα φίλτρα και η αναζήτηση μέσω κειμένου:

[22] Εικόνα 22: Παράδειγμα διαθέσιμης δουλειάς

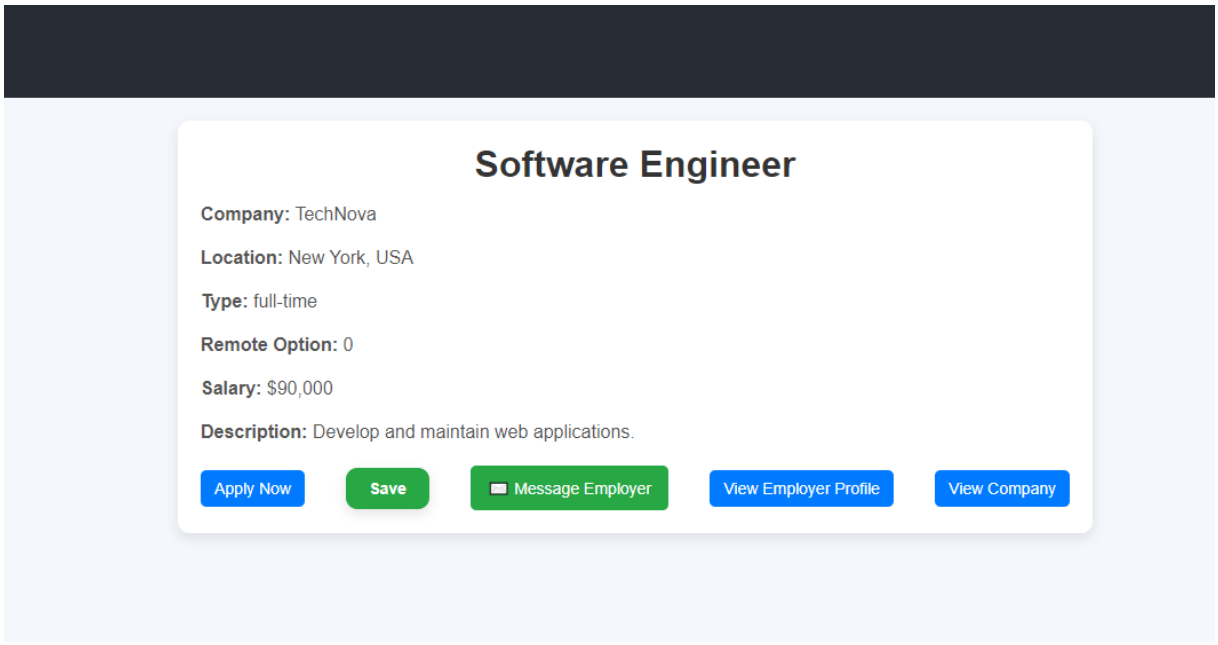


[23]

Τα φίλτρα που μπορούν να χρησιμοποιήσουν οι χρήστες είναι το εύρος του ετήσιου μισθού, οι κατηγορίες της δουλειάς, δηλαδή ο τομέας τους (Engineering, Marketing etc.), 25 στο σύνολο και η επιλογή του αν είναι Full-Time η Part-Time η Internship.

[23] Εικόνα 23: Φίλτρα θέσεων εργασίας

Παράδειγμα προβολής λεπτομερειών μιας δουλειάς όταν ο χρήστης πατάει πάνω στο job listing:

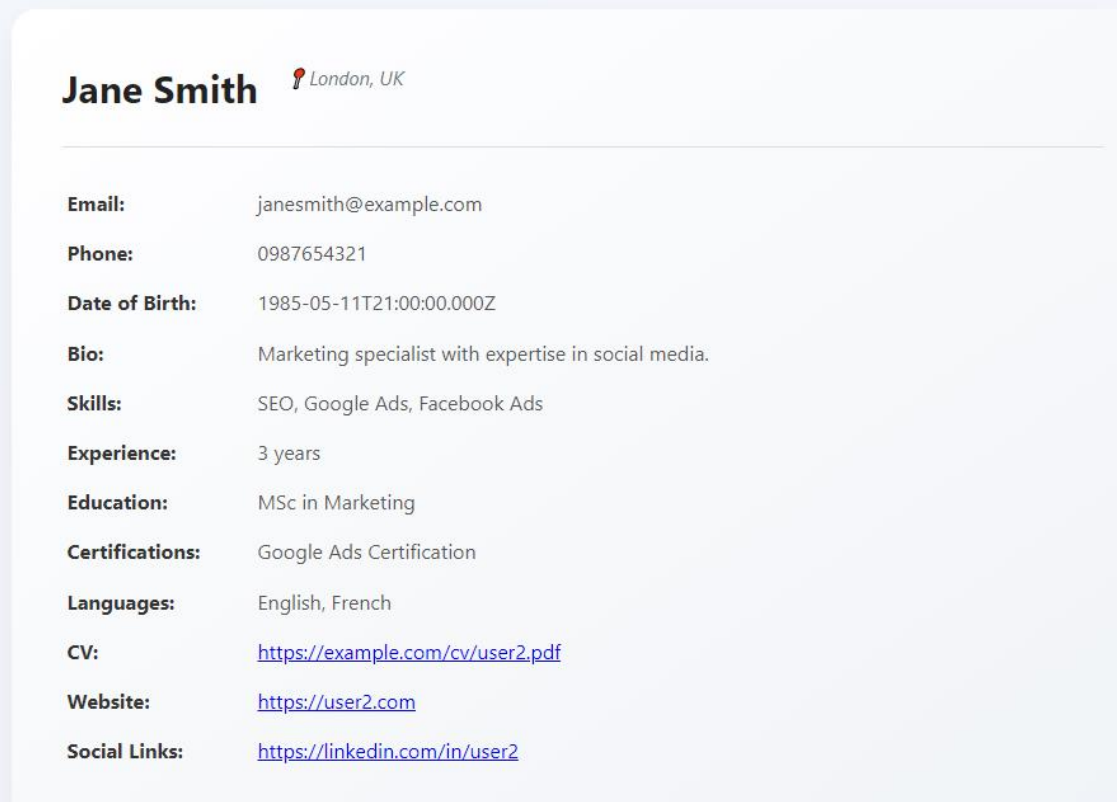


[24]

Εκτός από τα κουμπιά που υπάρχουν και στο Jobs, υπάρχουν και τα κουμπιά View Employer Profile και View Company.

Παράδειγμα View Employer Profile (η σελίδα Profile απλά για άλλον χρήστη):

[24] Εικόνα 24: Σελίδα με περισσότερες επιλογές για μια θέση εργασίας

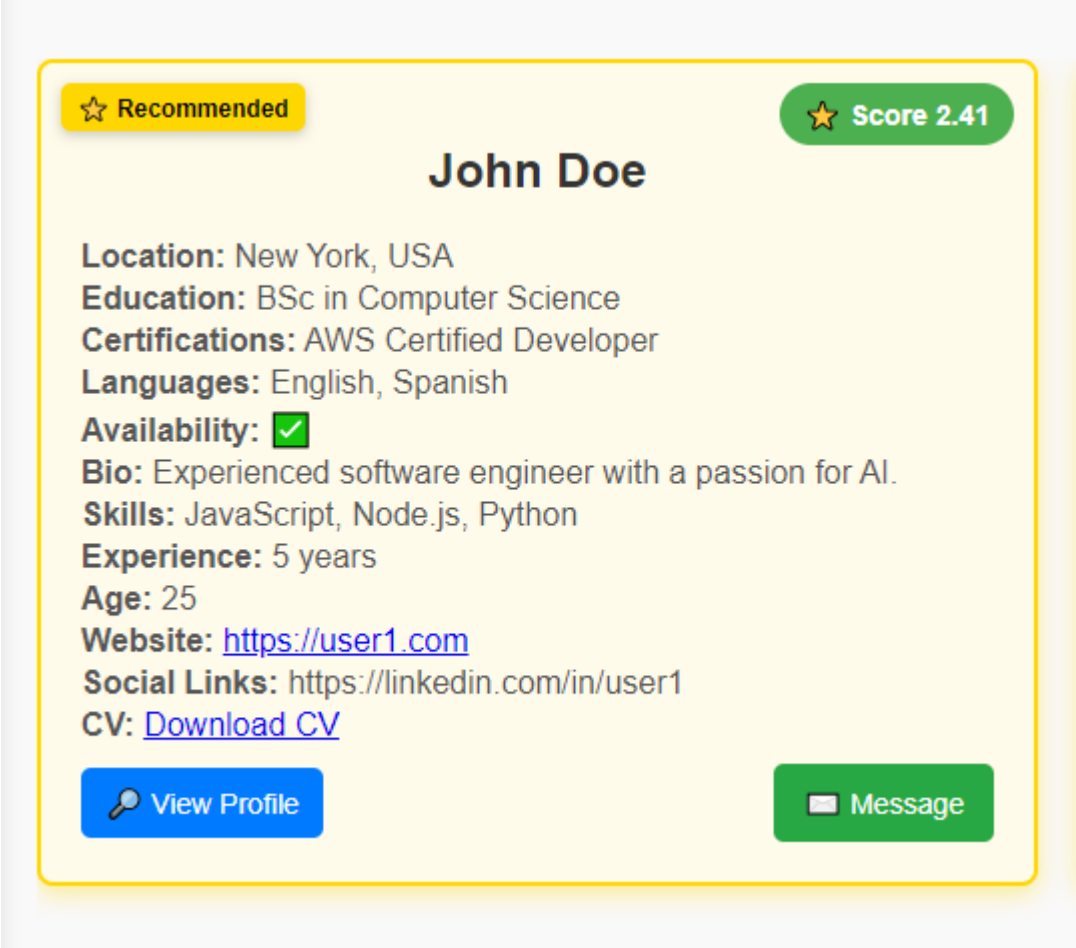


[25]

Στη σελίδα Candidates οι Employers μπορούν να δούν υποψηφίους ώστε να τους στείλουν μήνυμα για να δουλέψουν στις δουλειές που έχουν αναρτήσει. Επίσης μπορούν να δούν τους προτεινόμενους από την εφαρμογή Candidates για κάθε μια από τις δουλειές. Η αναζήτηση μπορεί να γίνει ευκολότερη με τη χρήση φίλτρων και αναζήτησης μέσω κειμένου. Τα φίλτρα είναι οι προτάσεις για κάθε δουλειά, η τοποθεσία του υποψηφίου και η διαθεσιμότητά του.

Παράδειγμα Recommended Candidate:

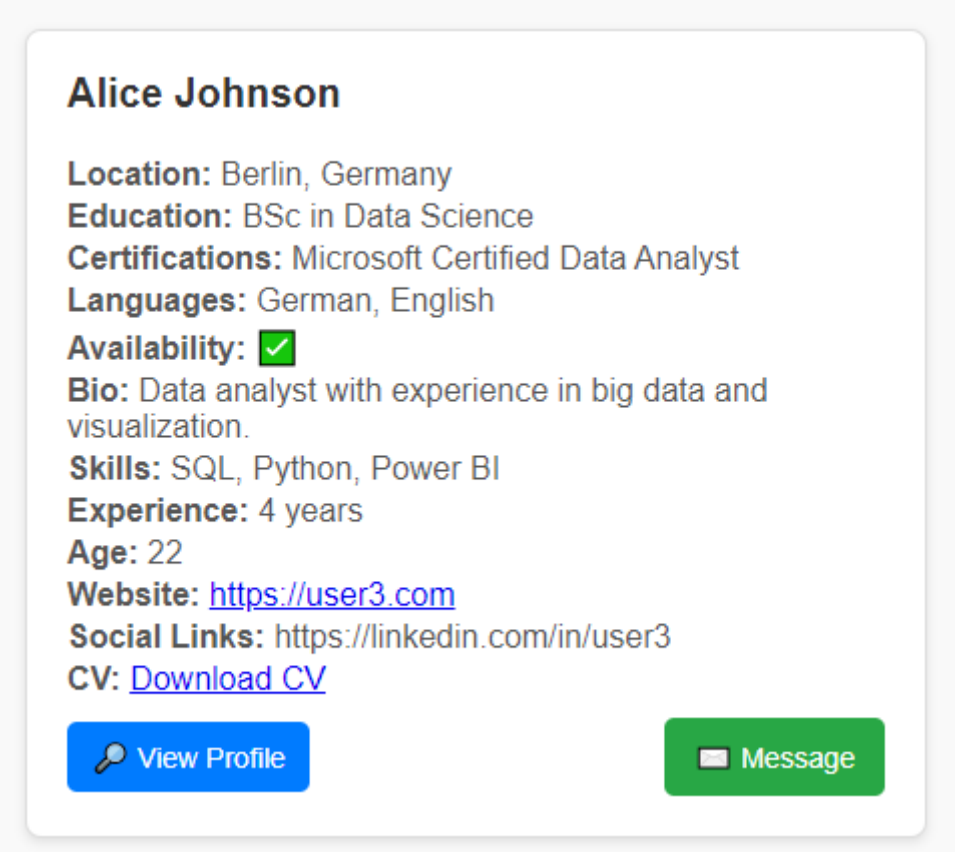
[25] Εικόνα 25: Σελίδα View Employer Profile



[26]

Παράδειγμα Candidate:

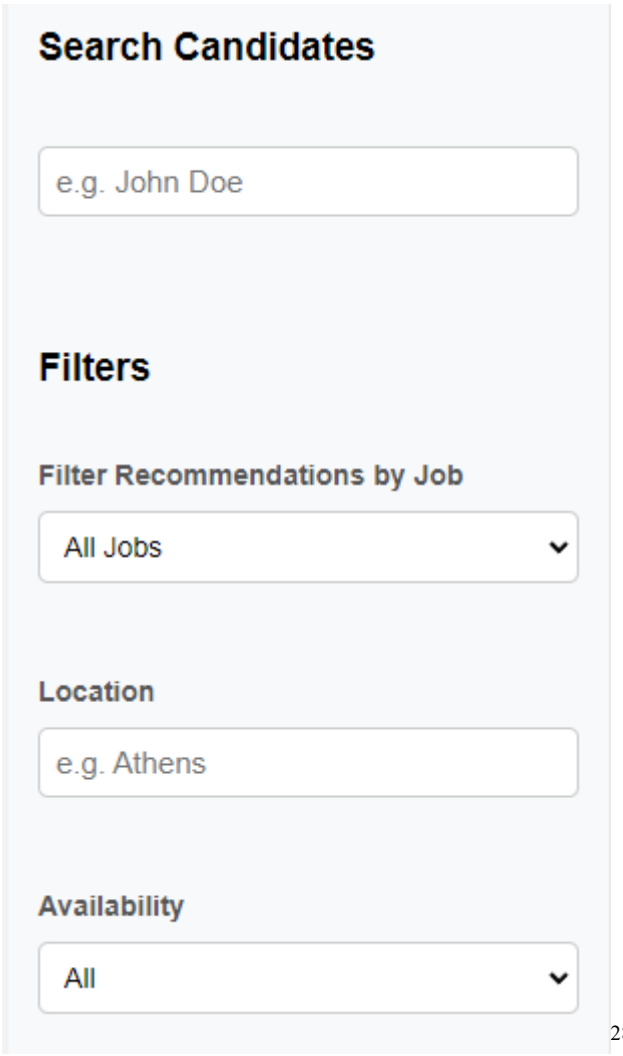
[26] Εικόνα 26: παράδειγμα πρότασης υποψηφίου



[27]

[27] Εικόνα 27: Παράδειγμα υποψηφίου

Τα φίλτρα των Candidates και η αναζήτηση:



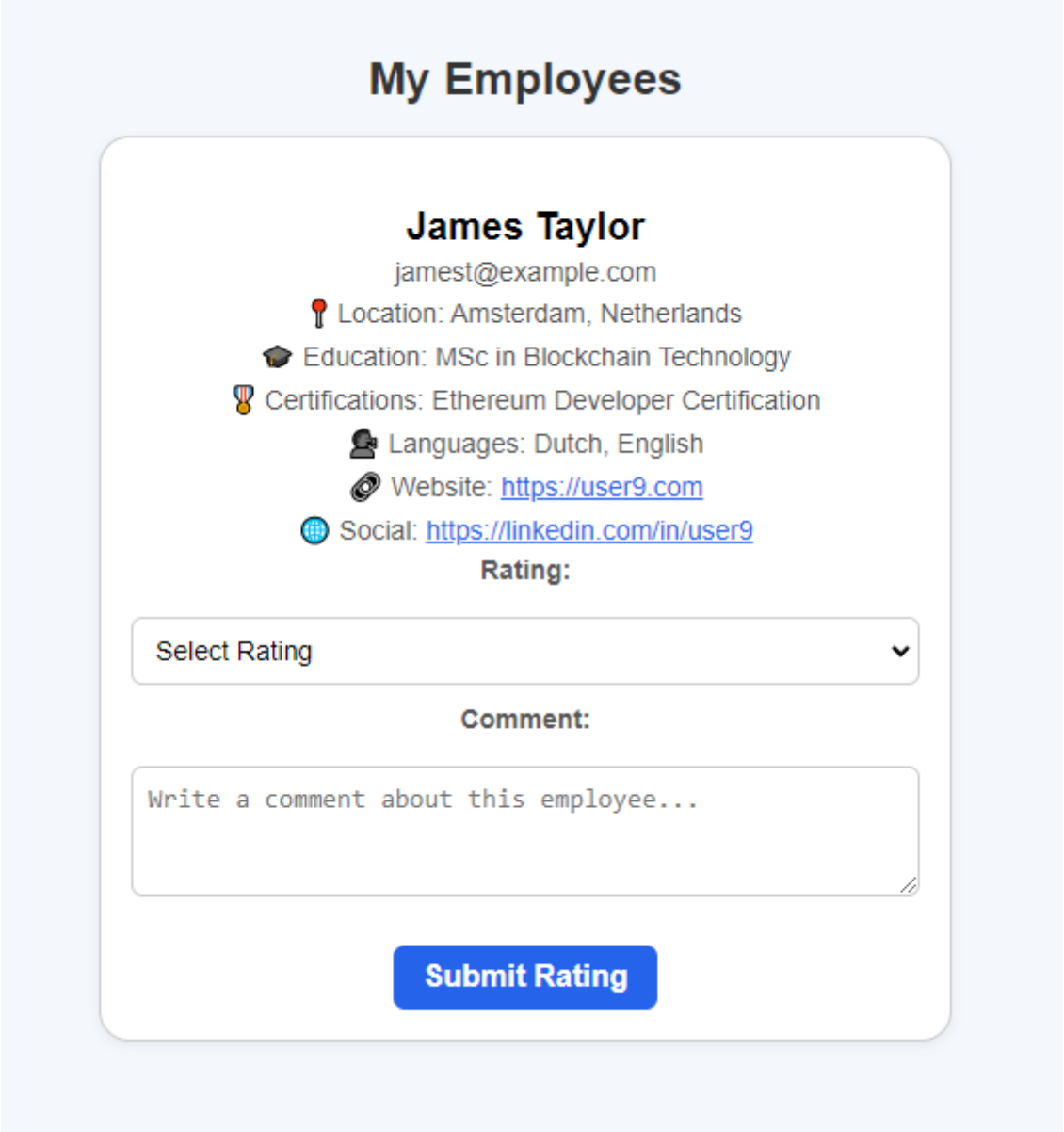
[28]

Τα φίλτρα εδώ είναι το Filter Recommendations by Job όπου ο χρήστης μπορεί να διαλέξει για ποιά από τις δουλειές που είναι εργοδότης θέλει να του εμφανίσει Recommendations, η τοποθεσία του Candidate αλλά και η διαθεσιμότητά του.

[28] Εικόνα 28: Φίλτρα υποψηφίων

Οι employers έχουν επίσης την σελίδα My Employees, στην οποία μπορούν να δούν τους employees που δουλεύουν για αυτούς και να τους κάνουν μια κριτική. Στην κριτική επίσης μπορούν να γράψουν και ένα σχόλιο για τον employee.

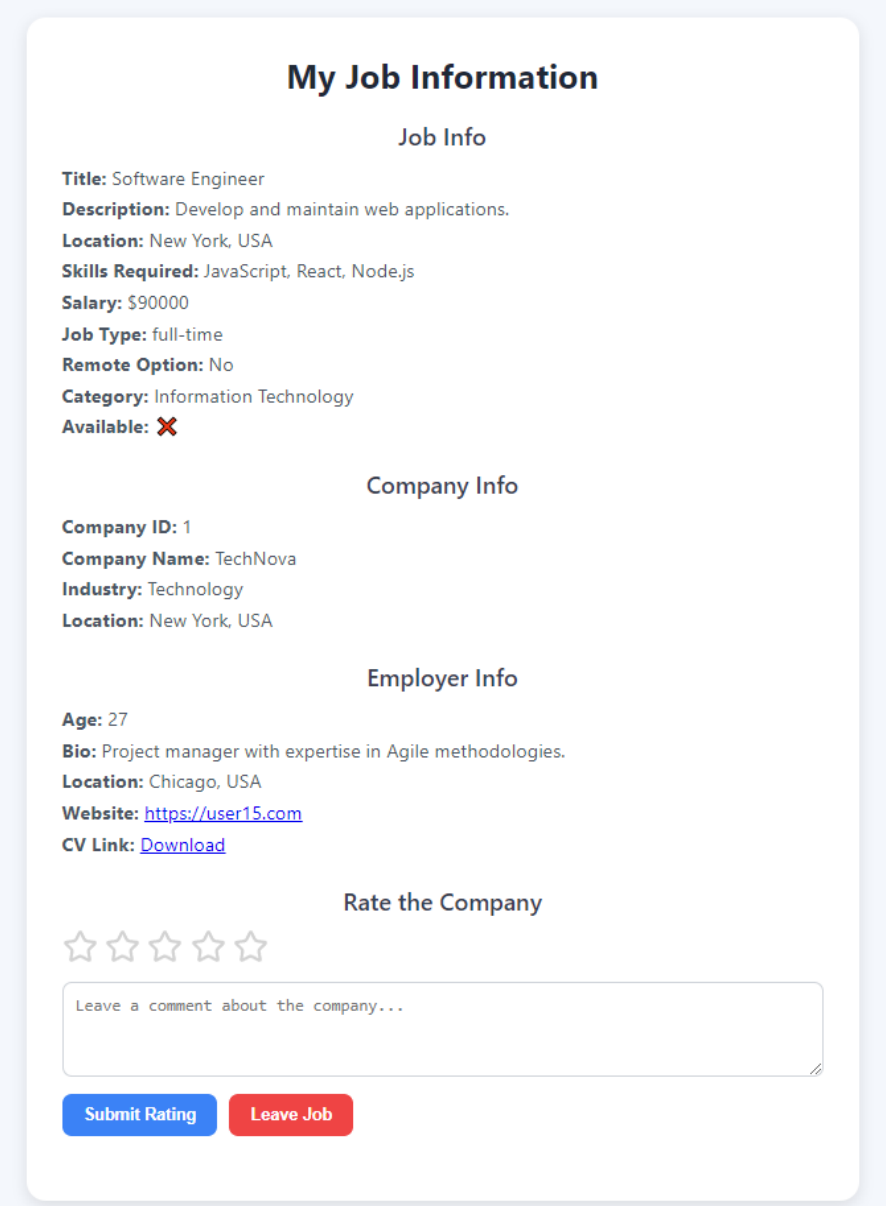
Παράδειγμα σελίδας My Employees με 1 employee:



29

[29] Εικόνα 29: Σελίδα My Employees

Οι employed, όσοι δηλαδή έχουν δουλειά, έχουν επίσης την σελίδα My Job, στην οποία μπορούν να δούν τη δουλειά τους, την εταιρία για την οποία δουλεύουν αλλά και τον εργοδότη τους και να τους κάνουν μια κριτική. Στην κριτική επίσης μπορούν να γράψουν και ένα σχόλιο για την δουλειά, την εταιρία και τον εργοδότη. Πατώντας το κουμπί Leave Job, ο εργαζόμενος ξαναγίνεται υποψήφιος για εργασία.



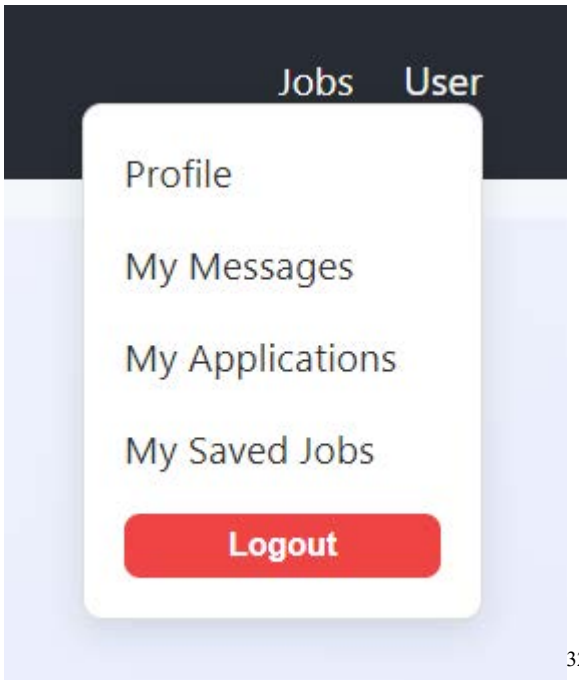
[30]

Στο navbar υπάρχει και το User button το οποίο όταν το πατούν οι χρήστες βγαίνει ένα dropdown menu.

[30] Εικόνα 30: Σελίδα My Job

[31]

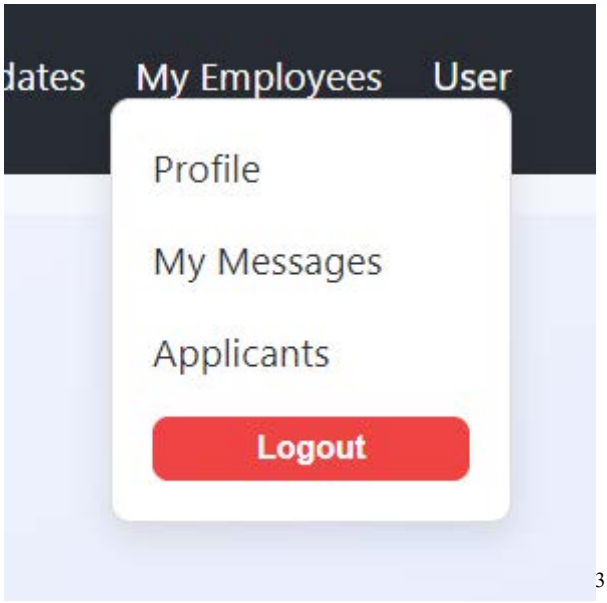
Στους candidates το user menu έχει αυτά τα στοιχεία:



[32]

Στους employers το user menu έχει αυτά τα στοιχεία:

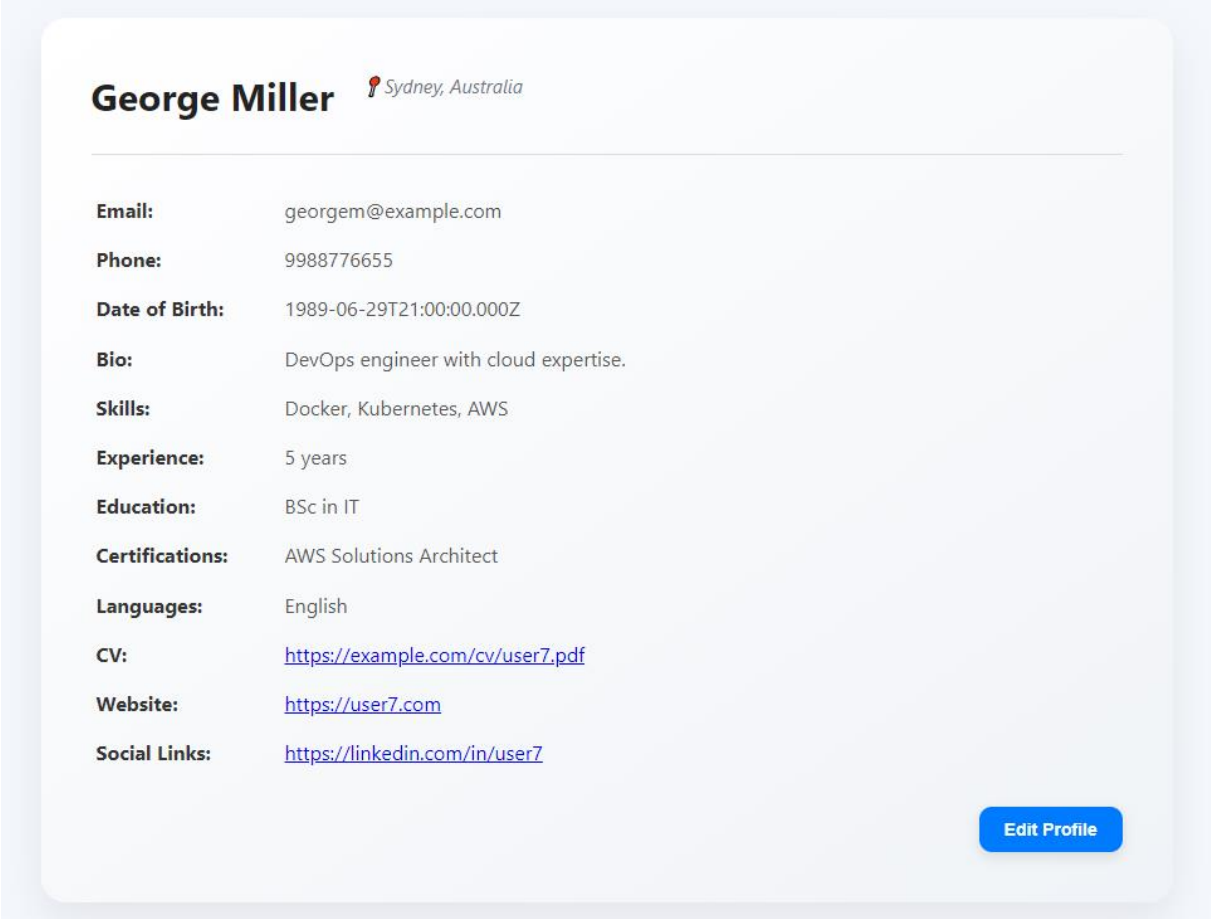
[31] Εικόνα 31: User Dropdown Menu
[32] Εικόνα 32: Το Dropdown Menu των υποψηφίων



[33]

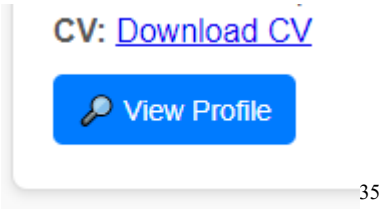
Η σελίδα Profile δείχνει το προφίλ του χρήστη καθώς και κάποιες σημαντικές πληροφορίες για αυτόν όπως η εκπαίδευσή του, η εργασιακή του εμπειρία, τα social links του και άλλα. Ο χρήστης εκτός από το να δεί τις πληροφορίες του μπορεί και να τις επεξεργαστεί με τη χρήση του Edit Profile button.

[33] Εικόνα 33: Το Dropdown Menu των εργοδοτών



[34]

Αυτή η σελίδα χρησιμοποιείται και από τις σελίδες οι οποίες θέλουν να δείξουν το προφίλ κάποιου άλλου χρήστη πέρα από αυτού που είναι συνδεδεμένος. Πατώντας το View Profile



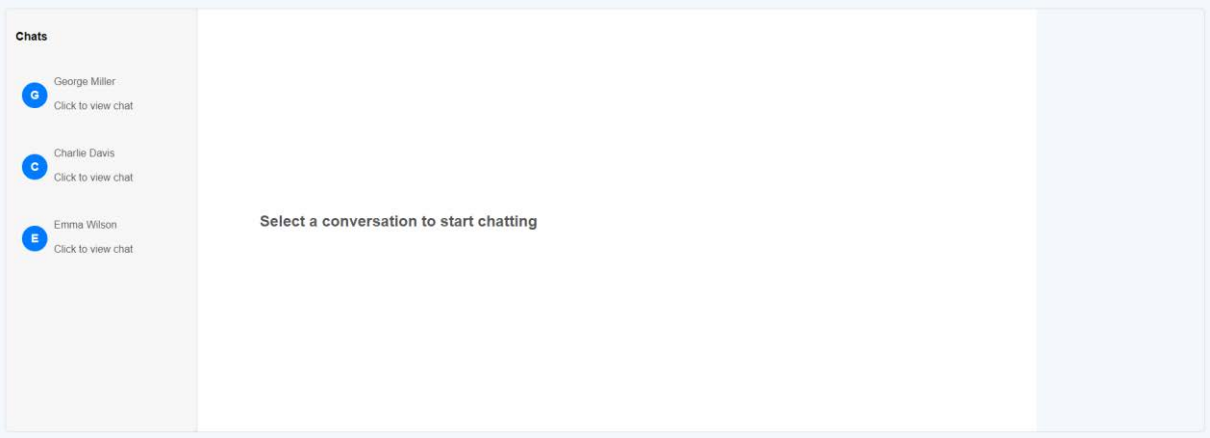
[35]

είτε στους Candidates είτε στους Employers ο χρήστης κατευθύνεται σε μια τέτοια σελίδα με τα στοιχεία του χρήστη του οποίου το προφίλ επιθυμεί να δεί.

Η σελίδα Messages δείχνει τις τρέχουσες και παλαιότερες συνομιλίες του χρήστη με άλλους χρήστες της εφαρμογής. Το chat είναι real-time και λειτουργεί μέσω του socket-

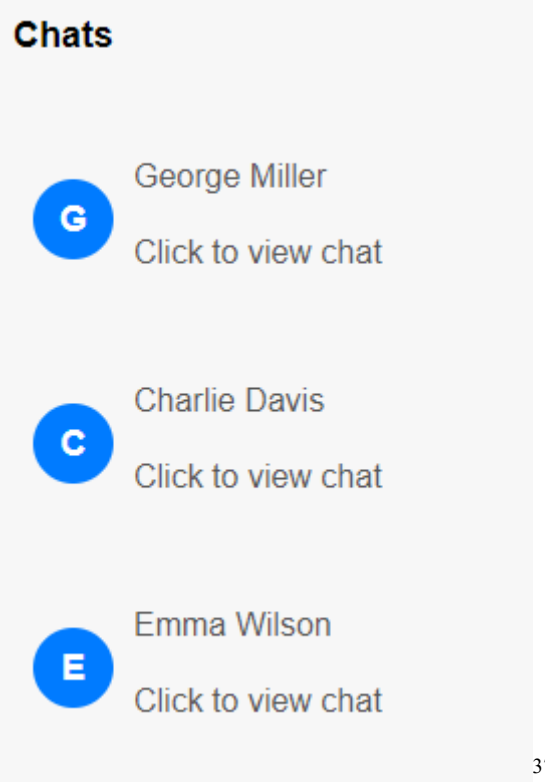
[34] Εικόνα 34: Η σελίδα Profile
[35] Εικόνα 35: Κουμπί View Profile

io server. Με αυτό τον τρόπο το chat επιτρέπει την ταυτόχρονη συνομιλία του χρήστη με πολλούς άλλους χρήστες και γίνεται καλύτερη η διαδικασία πρόσληψης, καθώς μπορεί να εξερευνήσει περισσότερες επιλογές.



[36]

Πατώντας ένα από αυτά τα buttons η εφαρμογή δείχνει τη συνομιλία με τον συγκεκριμένο χρήστη.



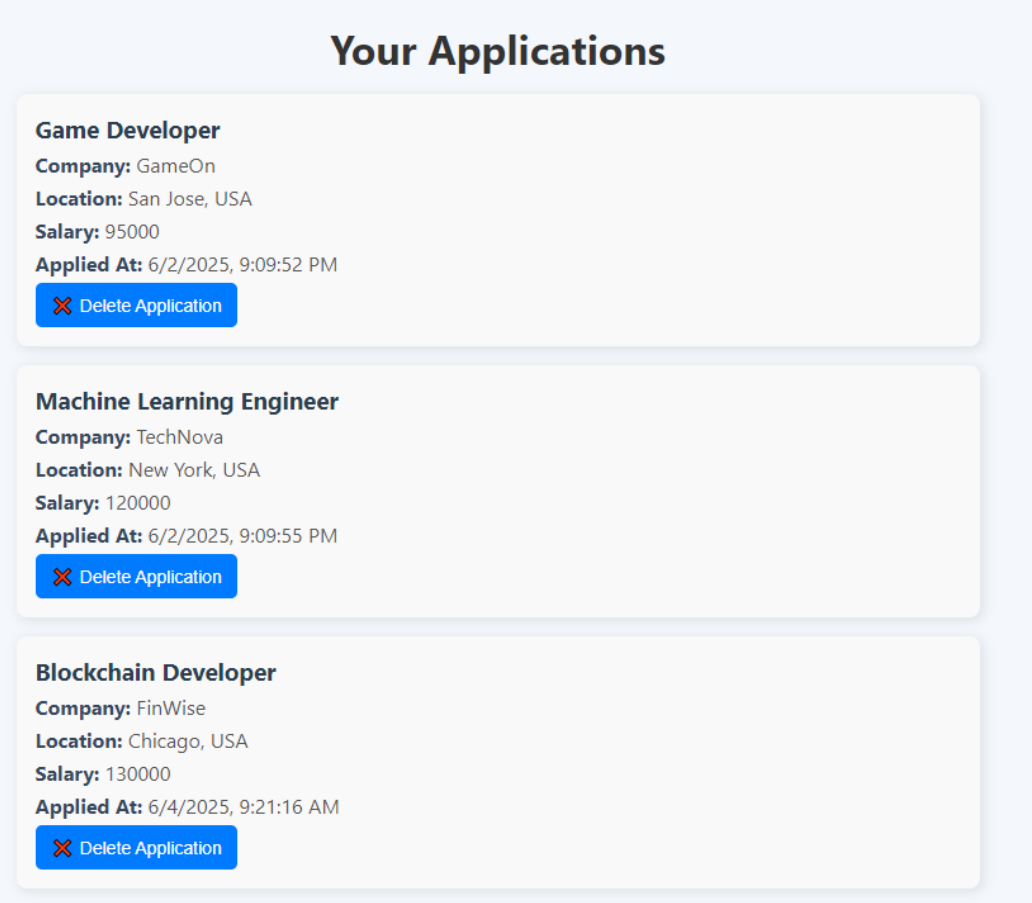
[37]

[36] Εικόνα 36: Σελίδα Messages με τις συνομιλίες του χρήστη με άλλους χρήστες
[37] Εικόνα 37: Χρήστες με τους οποίους έχει συνομιλήσει ο user

Παράδειγμα συνομιλίας δύο χρηστών:

[38]Η σελίδα My Applications δείχνει τις αιτήσεις που έχει κάνει ο χρήστης σε θέσεις εργασίας. Εκεί φαίνονται στοιχεία της δουλειάς και του εργοδότη. Ο κύριος σκοπός της σελίδας αυτής είναι να μπορεί ο χρήστης να δεί γρήγορα τις αιτήσεις του καθώς και να διαγράψει κάποια αν το επιθυμεί.

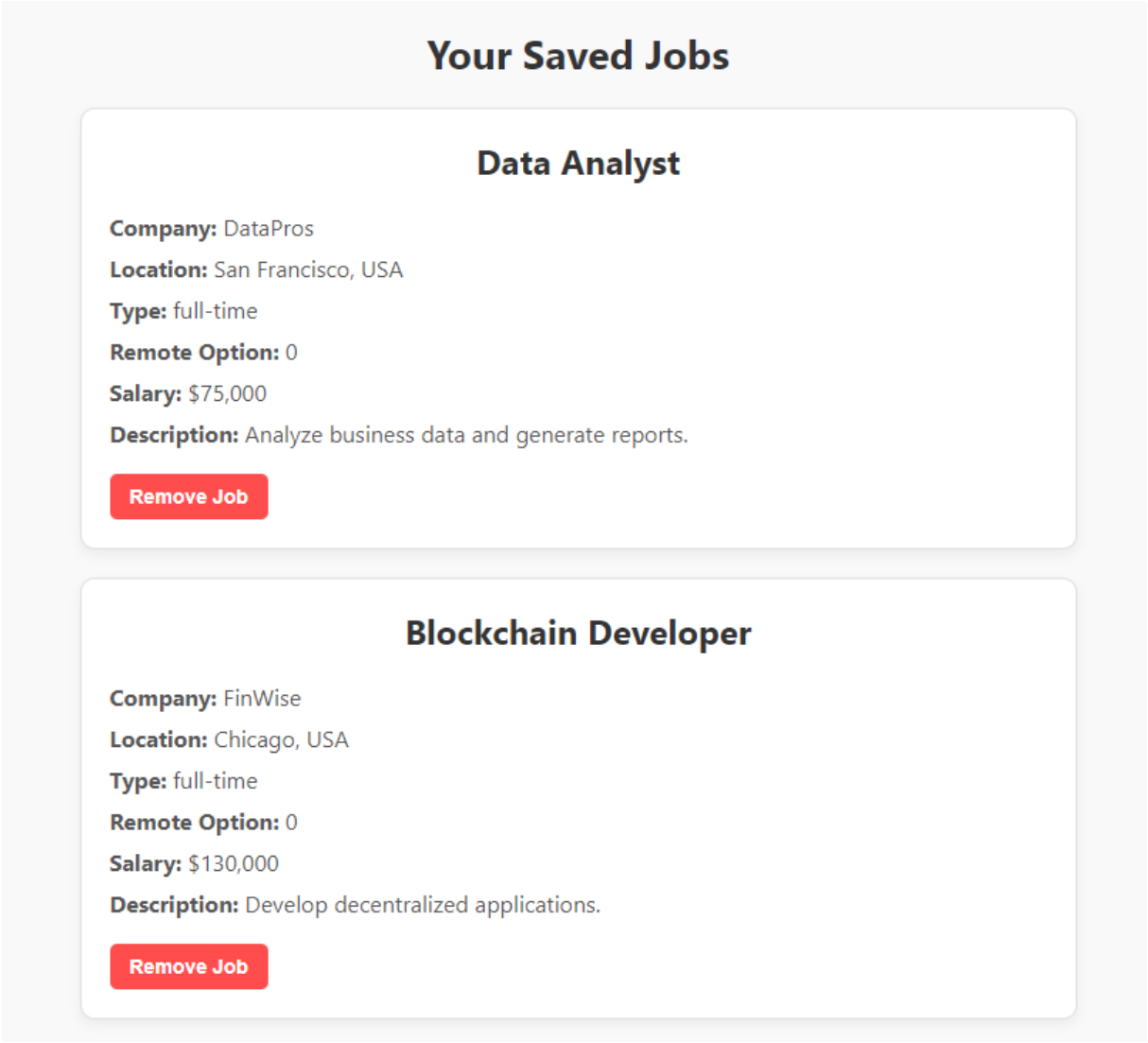
[38] Εικόνα 38: Παράδειγμα συνομιλίας δύο χρηστών



[39]

Η σελίδα My Saved Jobs δείχνει τις αποθηκευμένες δουλειές του χρήστη, δηλαδή αυτές στις οποίες έχει πατήσει save ώστε να τις ξαναδεί αργότερα.

[39] Εικόνα 39: Σελίδα My Applications που δείχνει τις αιτήσεις που έχει κάνει ο χρήστης



40

[40] Εικόνα 40: Σελίδα My Saved Jobs που έχει τις αποθηκευμένες δουλειές του χρήστη

Η σελίδα Applicants επιτρέπει στους employers να δούν ποιοί έχουν κάνει αιτήσεις στις δουλειές τους. Από εκεί μπορούν να κάνουν και αποδοχή του υποψηφίου και να γίνει εργαζόμενός τους η να τον απορρίψουν.

[41]

[41] Εικόνα 41: Σελίδα Applicants που δείχνει τους χρήστες που έχουν κάνει αίτηση σε δουλειά του εργοδότη

6 Συμπεράσματα

Η ανάπτυξη του προγράμματος έγινε σε διάστημα περίπου 6-7 μηνών.

Συνολικά η εργασία αυτή δημιουργήθηκε για να βοηθήσει στην επίλυση του προβλήματος της εύρεσης εργασίας στην σύγχρονη ψηφιακή εποχή. Το πρόγραμμα βασίστηκε στα κύρια σημεία που αποτελούν την διαδικασία της εύρεσης εργασίας και πρότεινε ένα εύχρηστο και γρήγορο τρόπος διασύνδεσης μεταξύ υποψηφίων και εργοδοτών.

Συνοπτικά, η εμπειρία μου με την ανάπτυξη αυτής της εφαρμογής ήταν πολύ θετική. Έμαθα πολλά καινούρια πράγματα και έκανα ότι καλύτερο μπορούσα ώστε να δημιουργήσω ένα καλό αποτέλεσμα.

Αν ξαναέκανα την πτυχιακή σήμερα με την εμπειρία και τις γνώσεις που απέκτησα, πιστεύω θα άλλαζα αρκετά πράγματα. Τόσο σε κάποιες τεχνολογίες που χρησιμοποιήθηκαν αλλά και στο πως διαχειρίστηκα τον χρόνο στον οποίο αναπτύχθηκε η εφαρμογή.

Παρόλες τις δυσκολίες όμως, άξιζε τον κόπο διότι είδα πως είναι ένα μεγάλο πρακτικό κομμάτι της δουλειάς αυτής και μπορώ να πώ πως μου δημιούργησε την επιθυμία να ασχοληθώ με αυτό και μελλοντικά.

Η εφαρμογή όπως είναι φτιαγμένη είναι λειτουργική όμως αξίζει να γίνει μια αναφορά και σε κάποιες μελλοντικές επεκτάσεις που μπορούν να γίνουν. Αρχικά ένα σημαντικό κομμάτι θα ήταν η ανάπτυξη ενός καλύτερου Recommendation System τόσο για δουλειές όσο και για υποψηφίους. Θα μπορούσε για παράδειγμα να γίνει με ένα καλύτερο σύστημα τεχνητής νοημοσύνης. Ένα άλλο κομμάτι που θα μπορούσε να βελτιωθεί είναι η εμφάνιση της εφαρμογής, δηλαδή η css που χρησιμοποιεί. Θα μπορούσαν επίσης να δοθούν περισσότερες επιλογές στις φόρμες ώστε να υπάρχουν περισσότερες πληροφορίες για τους χρήστες και τις δουλειές. Έτσι θα γινόταν καλύτερη η αναζήτηση και θα υπήρχαν και περισσότερα φίλτρα που θα βοηθούσαν στην διαδικασία. Μια ακόμη βελτίωση που θα μπορούσε να γίνει είναι να προστεθούν περισσότερες δραστηριότητες μέσα στην εφαρμογή όπως πχ η δυνατότητα βιντεοκλήσης ή δημιουργίας posts ώστε να μπορούν οι χρήστες να αλληλεπιδρούν με περισσότερους τρόπους. Τέτοιες βελτιώσεις

θα ανέβαζαν συνολικά το επίπεδο της εφαρμογής καθώς είναι πολύ επεκτάσιμη και βρίσκει χρήση στην σημερινή αγορά.

Βιβλιογραφία

[1] https://about.linkedin.com - Δημοφιλής σελίδα κοινωνικής δικτύωσης στοχευμένη στον επαγγελματικό τομέα.

[2] https://gr.indeed.com/about?hl=el - Σελίδα για εύρεση εργασίας.

[3] https://www.flexjobs.com/about - Σελίδα εύρεσης εξ' αποστάσεως εργασίας.

[4] https://job.com - Σελίδα εύρεσης εργασίας με μηχανισμό NLP

[5] https://react.dev - Βιβλιοθήκη της Javascript για Front-End development.

[6] https://thenewstack.io/javascripts-history-and-how-it-led-to-reactjs/ - Η ιστορία της React.

[7] https://www.codecademy.com/article/react-virtual-dom - Το Virtual DOM της React.

[8] https://socket.io/docs/v4/ - Documentation για το Socket.io.

[9] https://jwt.io/introduction - JSON Web Token, μέθοδος authentication.

[10] https://nodejs.org/en/learn/getting-started/introduction-to-nodejs - Εισαγωγή στο Node.js.

[11] https://brainhub.eu/library/famous-apps-built-with-nodejs - Εφαρμογές που χρησιμοποιούν το Node.js

[12] https://expressjs.com - Το framework Express.js.

[13] https://expressjs.com/en/guide/routing.html - Πληροφορίες για το routing του Express.js.

[14] https://dev.mysql.com/doc/refman/8.0/en/introduction.html - Πληροφορίες για τη MySQL.

[15] https://www.mysql.com/why-mysql/ - Λόγοι επιλογής της MySQL.

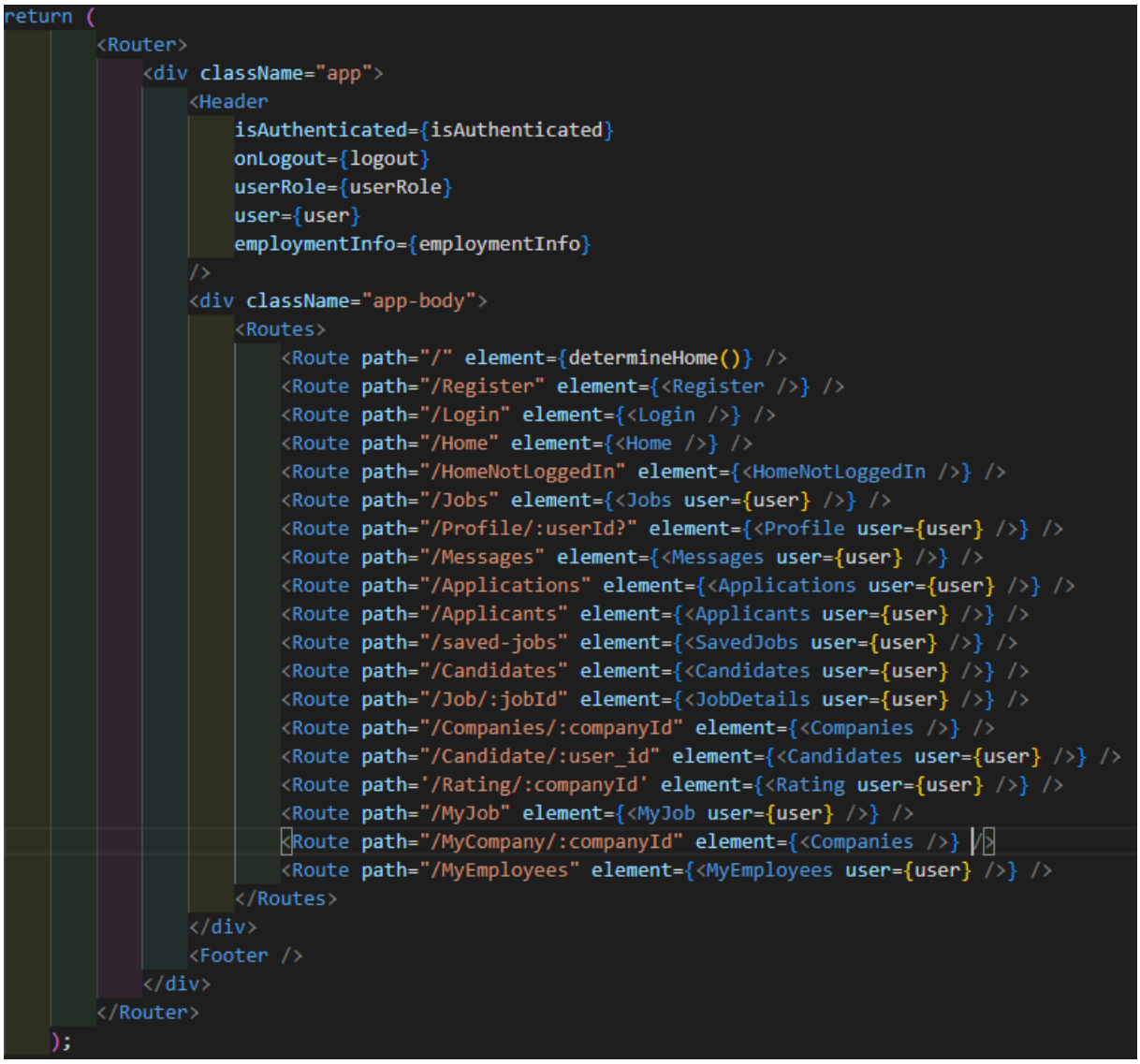
[16] https://www.oracle.com/gr/database/what-is-a-relational-database/ - Άρθρο για τις σχεσιακές βάσεις δεδομένων.

[17] https://wampserver.aviatechno.net - Ιστοσελίδα του WAMP Server.

Παράρτημα Α

Παρακάτω παραθέτω ενδεικτικά κάποια κομμάτια κώδικα από το Front End.

Το Router της εφαρμογής, που είναι το return της συνάρτησης InnerApp του App.js που είναι το κύριο component μιας React εφαρμογής. Εδώ υπάρχουν όλα τα Routes που χρησιμοποιεί η εφαρμογή για να εμφανίσει τις σωστές σελίδες.



```
return (
    <Router>
        <div className="app">
            <Header
                isAuthenticated={isAuthenticated}
                onLogout={logout}
                userRole={userRole}
                user={user}
                employmentInfo={employmentInfo}
            />
            <div className="app-body">
                <Routes>
                    <Route path="/" element={determineHome()} />
                    <Route path="/Register" element={<Register />} />
                    <Route path="/Login" element={<Login />} />
                    <Route path="/Home" element={<Home />} />
                    <Route path="/HomeNotLoggedIn" element={<HomeNotLoggedIn />} />
                    <Route path="/Jobs" element={<Jobs user={user} />} />
                    <Route path="/Profile/:userId?" element={<Profile user={user} />} />
                    <Route path="/Messages" element={<Messages user={user} />} />
                    <Route path="/Applications" element={<Applications user={user} />} />
                    <Route path="/Applicants" element={<Applicants user={user} />} />
                    <Route path="/saved-jobs" element={<SavedJobs user={user} />} />
                    <Route path="/Candidates" element={<Candidates user={user} />} />
                    <Route path="/Job/:jobId" element={<JobDetails user={user} />} />
                    <Route path="/Companies/:companyId" element={<Companies />} />
                    <Route path="/Candidate/:user_id" element={<Candidates user={user} />} />
                    <Route path='/Rating/:companyId' element={<Rating user={user} />} />
                    <Route path="/MyJob" element={<MyJob user={user} />} />
                    <Route path="/MyCompany/:companyId" element={<Companies />} />
                    <Route path="/MyEmployees" element={<MyEmployees user={user} />} />
                </Routes>
            </div>
            <Footer />
        </div>
    </Router>
);
```

42

Η συνάρτηση Home όπου είναι το Home Component της εφαρμογής.

[42] Εικόνα 42: Κώδικας του React Router

```
function Home() {
    const { user } = useContext(AuthContext);
    const navigate = useNavigate();
    return (
        <div className="home-container">
            <h1>Welcome {user ? user.first_name : 'Guest'}!</h1>
            {!user && (
                <p>Please log in to access job listings or employer tools.</p>
            )}
            {(user?.role === 'candidate' || user?.role === 'employed') && (
                <>
                    <h2>Find your next job here!</h2>
                    <button className="jobs-button" onClick={() => navigate('/Jobs')}>
                        Browse Jobs
                    </button>
                </>
            )}
            {user?.role === 'employer' && (
                <>
                    <h2>Find your next employee here!</h2>
                    <button className="jobs-button" onClick={() => navigate('/Candidates')}>
                        Browse Candidates
                    </button>
                </>
            )}
        </div>
    );
}
```

43

Η φόρμα του Login.

[43] Εικόνα 43: Συνάρτηση Home του Front-End

[44]

[44] Εικόνα 44: Φόρμα Login της εφαρμογής

Ο μηχανισμός δημιουργίας και διαγραφής του authentication token στο local storage. Αυτός ο μηχανισμός καλείται κάθε φορά που ο χρήστης κάνει Login στην εφαρμογή.

```
export const AuthProvider = ({ children }) => {
    const [user, setUser] = useState(null);
    const login = (userData) => {
        setUser(userData);
        localStorage.setItem('token', userData.token);
        localStorage.setItem('user_id', userData.user_id);
        localStorage.setItem('role', userData.role);
    };
    const logout = () => {
        localStorage.removeItem('token');
        localStorage.removeItem('user_id');
        localStorage.removeItem('role');
        setUser(null);
    };
    useEffect(() => {
        const token = localStorage.getItem('token');
        const user_id = localStorage.getItem('user_id');
        const role = localStorage.getItem('role');
        if (token && user_id && role) {
            setUser({ user_id, role, token });
        }
    }, []);
    return (
        <AuthContext.Provider value={{ user, login, logout }}>
            {children}
        </AuthContext.Provider>
    );
};
```

[45]

[45] Εικόνα 45: Συνάρτηση AuthProvider

Ο κώδικας του Interface των μηνυμάτων. Εδώ είναι το chatbox, το input container και το μήνυμα του Select a conversation.

```
<div className="chat-box">
    {selectedUser ? (
        <>
            <div className="chat-header">
                <h2>{selectedUser.first_name} {selectedUser.last_name}</h2>
            </div>
            <div className="messages">
                {messages.map((msg, index) => (
                    <div
                        key={index}
                        className={`message ${msg.sender_id == user.user_id ? "sent" : "received"}`}
                    >
                        <p>{msg.message}</p>
                    </div>
                ))}
                <div ref={messagesEndRef} />
            </div>
            <div className="input-container">
                <input
                    type="text"
                    value={message}
                    onChange={(e) => setMessage(e.target.value)}
                    placeholder="Type a message..."
                    onKeyDown={(e) => { if (e.key === "Enter") sendMessage(); }}
                />
                <button onClick={sendMessage}>Send</button>
            </div>
        </>
    ) : (
        <p className="select-chat"><strong>Select a conversation to start chatting</strong></p>
    )}
</div>
```

[46]

[46] Εικόνα 46: Κώδικας Interface Μηνυμάτων

Παράρτημα Β

Παρακάτω παραθέτω ενδεικτικά κάποια κομμάτια κώδικα από το Back End.

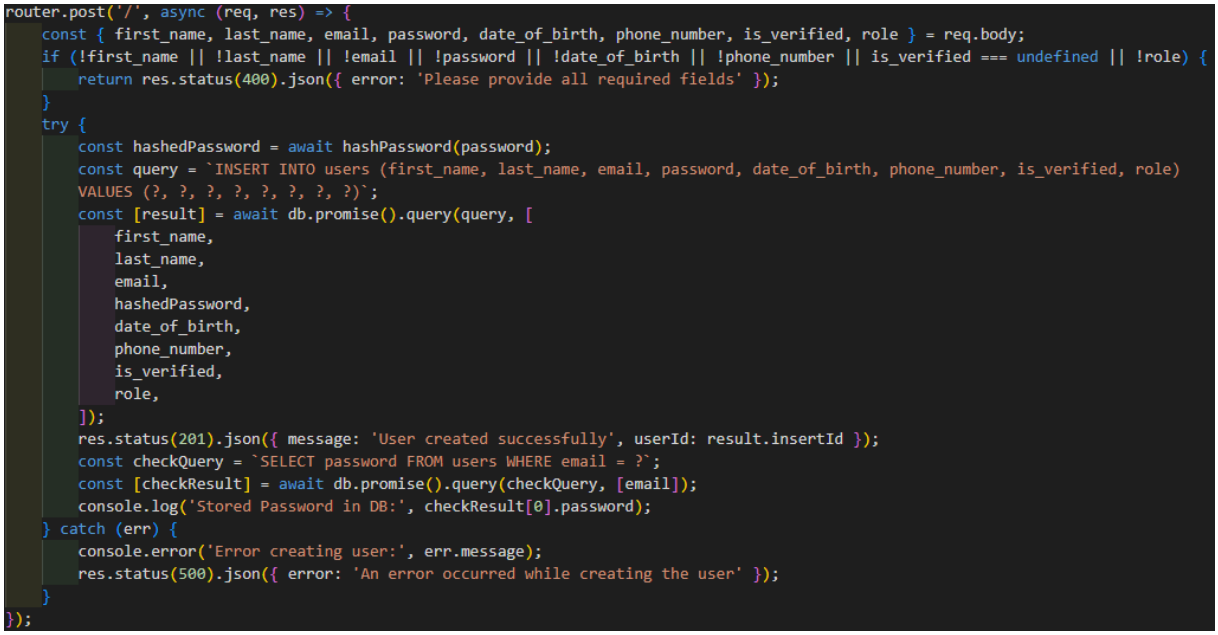
Οι συναρτήσεις που χρησιμοποιούνται για το Hashing & Salting του Password πριν μπεί στη βάση δεδομένων.

```
const bcrypt = require('bcrypt');

const saltRounds = 10;
const hashPassword = async (password) => {
    const salt = await bcrypt.genSalt(saltRounds);
    return bcrypt.hash(password, salt);
};
const comparePassword = async (password, hashedPassword) => {
    return bcrypt.compare(password, hashedPassword);
};
module.exports = { hashPassword, comparePassword };
```

[47]

Το POST request του Register. Όταν χρησιμοποιηθεί δημιουργείται ένας νέος user στη βάση δεδομένων.

[47] Εικόνα 46: Συνάρτηση κρυπτογράφησης και επαλήθευσης του κωδικού πρόσβασης



```
router.post('/', async (req, res) => {
    const { first_name, last_name, email, password, date_of_birth, phone_number, is_verified, role } = req.body;
    if (!first_name || !last_name || !email || !password || !date_of_birth || !phone_number || is_verified === undefined || !role) {
        return res.status(400).json({ error: 'Please provide all required fields' });
    }
    try {
        const hashedPassword = await hashPassword(password);
        const query = `INSERT INTO users (first_name, last_name, email, password, date_of_birth, phone_number, is_verified, role)
        VALUES (?, ?, ?, ?, ?, ?, ?, ?)`;
        const [result] = await db.promise().query(query, [
            first_name,
            last_name,
            email,
            hashedPassword,
            date_of_birth,
            phone_number,
            is_verified,
            role,
        ]);
        res.status(201).json({ message: 'User created successfully', userId: result.insertId });
        const checkQuery = `SELECT password FROM users WHERE email = ?`;
        const [checkResult] = await db.promise().query(checkQuery, [email]);
        console.log('Stored Password in DB:', checkResult[0].password);
    } catch (err) {
        console.error('Error creating user:', err.message);
        res.status(500).json({ error: 'An error occurred while creating the user' });
    }
});
```

48

Το POST request του Login.

[48] Εικόνα 47: Κώδικας του Back-End για Register στην εφαρμογή

```
router.post('/login', async (req, res) => {
    const { email, password } = req.body;
    if (!email || !password) {
        return res.status(400).json({ error: 'Email and password are required' });
    }
    try {
        const query = `SELECT * FROM users WHERE email = ?`;
        const [rows] = await db.promise().query(query, [email]);
        if (rows.length === 0) {
            return res.status(401).json({ error: 'Invalid email or password' });
        }
        const user = rows[0];
        let isMatch = false;
        isMatch = await comparePassword(password, user.password);
        if (!isMatch) {
            return res.status(401).json({ error: 'Invalid email or password' });
        }
        const token = jwt.sign(
            { user_id: user.user_id, role: user.role },
            JWT_SECRET,
            { expiresIn: '24h' }
        );
        res.status(200).json({
            message: 'Login successful',
            token,
            user: {
                user_id: user.user_id,
                first_name: user.first_name,
                last_name: user.last_name,
                role: user.role
            }
        });
    } catch (err) {
        console.error('Error during login:', err.message);
        res.status(500).json({ error: 'An error occurred while processing your request' });
    }
});
```

49

[49] Εικόνα 48: Κώδικας του Back-End για Login στην εφαρμογή

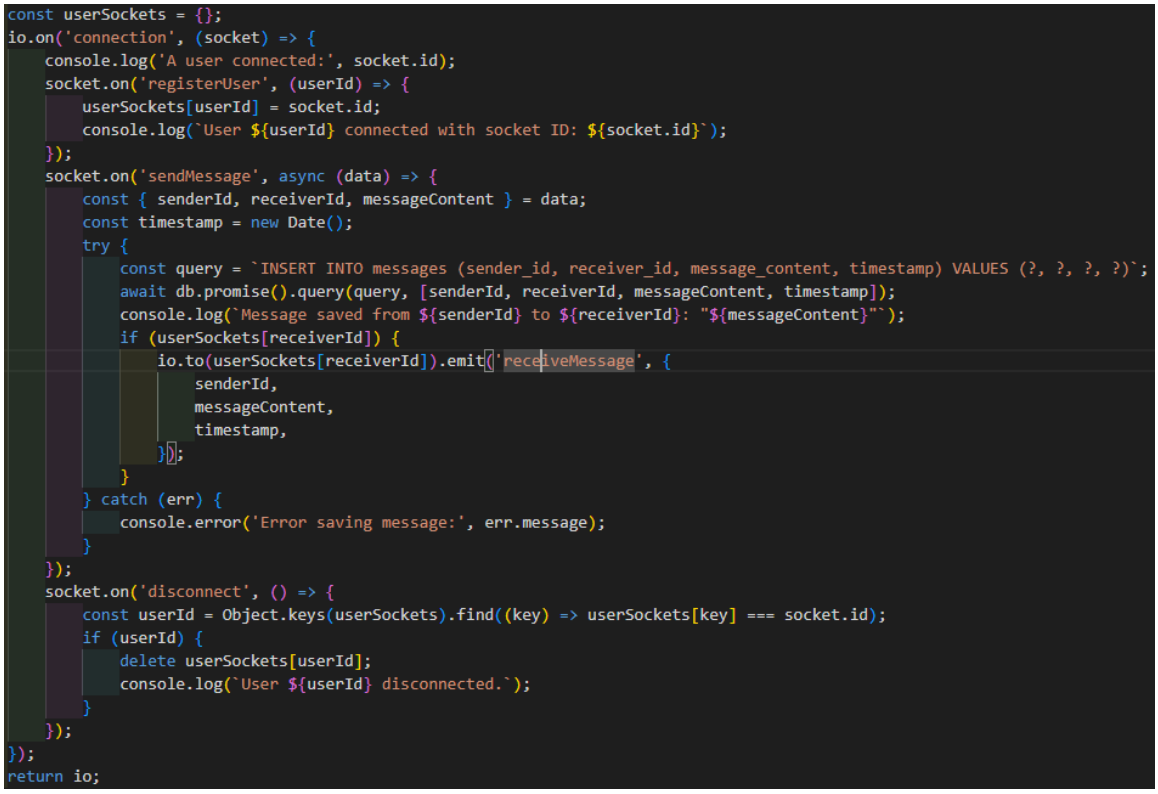
Η συνάρτηση Consume Similarity Batch που υπολογίζει ποσοστά ομοιότητας μεταξύ μιας δουλειάς και ενός υποψηφίου.

```
async function computeSimilarityBatch(jobInputs, resumeInputs) {
    await initModel();

    const jobTexts = jobInputs.map(composeText);
    const resumeTexts = resumeInputs.map(composeText);
    const allTexts = [...jobTexts, ...resumeTexts];

    const embeddings = await model.embed(allTexts);

    const jobEmbeddings = embeddings.slice([0, 0], [jobInputs.length, -1]);
    const resumeEmbeddings = embeddings.slice([jobInputs.length, 0], [resumeInputs.length, -1]);

    const normJobs = l2Normalize(jobEmbeddings);
    const normResumes = l2Normalize(resumeEmbeddings);

    const similarityMatrix = tf.matMul(normJobs, normResumes, false, true);
    const rawScores = await similarityMatrix.array();

    const normalizedScores = rawScores.map(row => row.map(score => normalizeScore(score)));
    return normalizedScores;
}
```

[50]

Δημιουργία του Socket IO server, διαχείριση σύνδεσης και αποσύνδεσης των χρηστών και αποστολής μηνυμάτων μεταξύ τους.

[50] Εικόνα 49: Συνάρτηση Consume Similarity Batch

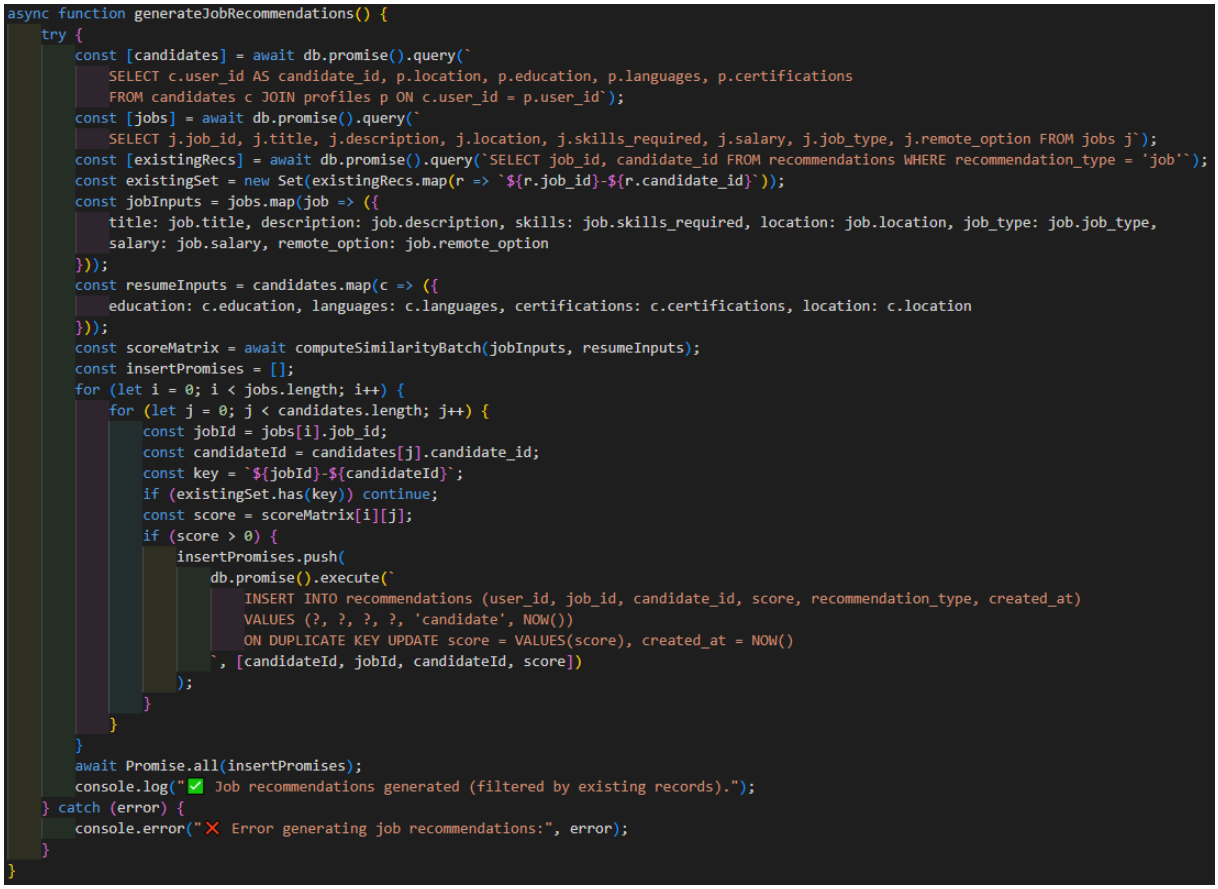


```
const userSockets = {};
io.on('connection', (socket) => {
    console.log('A user connected:', socket.id);
    socket.on('registerUser', (userId) => {
        userSockets[userId] = socket.id;
        console.log(`User ${userId} connected with socket ID: ${socket.id}`);
    });
    socket.on('sendMessage', async (data) => {
        const { senderId, receiverId, messageContent } = data;
        const timestamp = new Date();
        try {
            const query = `INSERT INTO messages (sender_id, receiver_id, message_content, timestamp) VALUES (?, ?, ?, ?)`;
            await db.promise().query(query, [senderId, receiverId, messageContent, timestamp]);
            console.log(`Message saved from ${senderId} to ${receiverId}: "${messageContent}"`);
            if (userSockets[receiverId]) {
                io.to(userSockets[receiverId]).emit('receiveMessage', {
                    senderId,
                    messageContent,
                    timestamp,
                });
            }
        } catch (err) {
            console.error('Error saving message:', err.message);
        }
    });
    socket.on('disconnect', () => {
        const userId = Object.keys(userSockets).find((key) => userSockets[key] === socket.id);
        if (userId) {
            delete userSockets[userId];
            console.log(`User ${userId} disconnected.`);
        }
    });
});
return io;
```

51

Η συνάρτηση Generate Job Recommendations όπου δημιουργεί τις προτάσεις θέσεων εργασίας στους υποψήφιους.

[51] Εικόνα

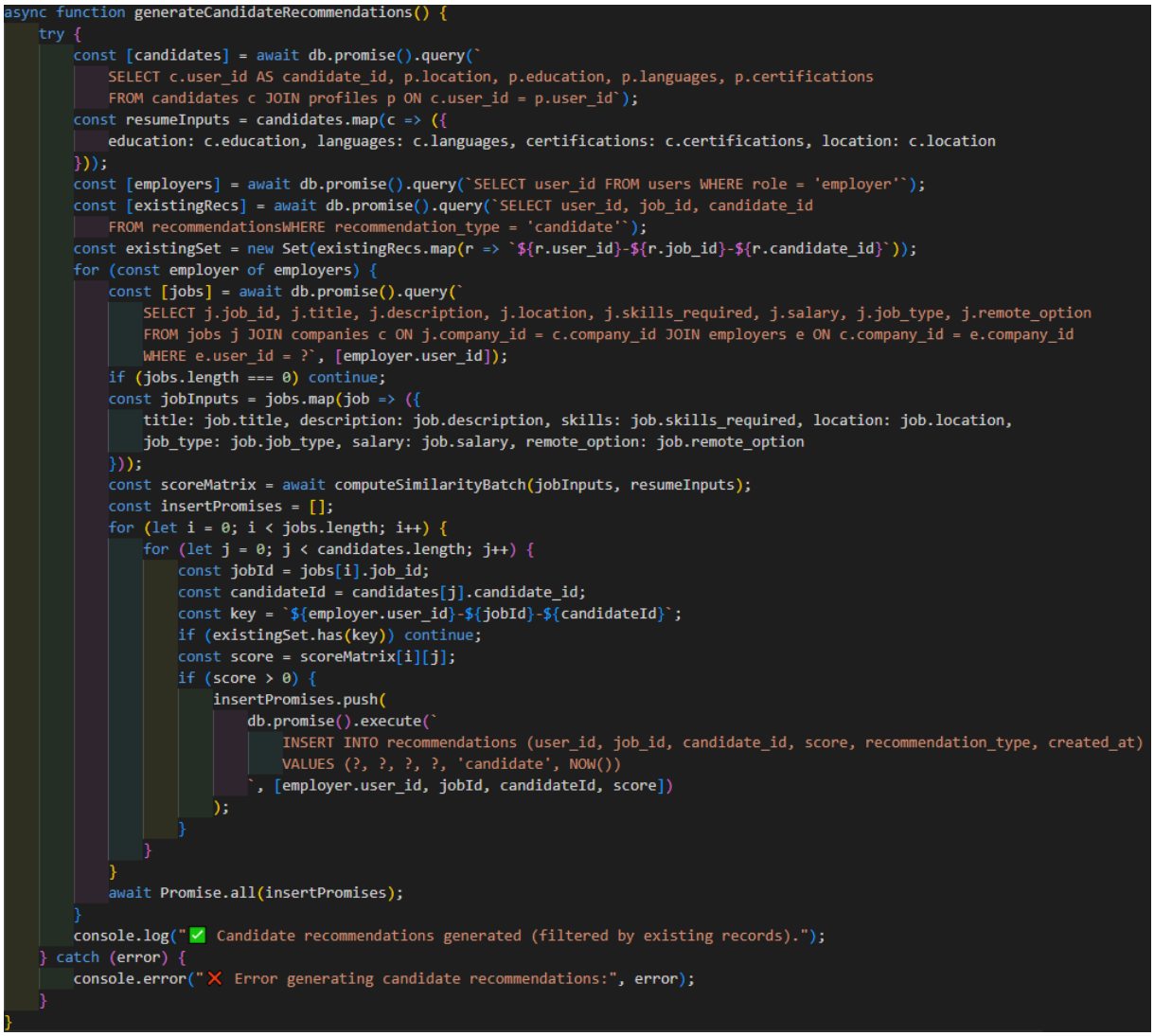


```
async function generateJobRecommendations() {
    try {
        const [candidates] = await db.promise().query(`
            SELECT c.user_id AS candidate_id, p.location, p.education, p.languages, p.certifications
            FROM candidates c JOIN profiles p ON c.user_id = p.user_id`);
        const [jobs] = await db.promise().query(`
            SELECT j.job_id, j.title, j.description, j.location, j.skills_required, j.salary, j.job_type, j.remote_option FROM jobs j`);
        const [existingRecs] = await db.promise().query(`SELECT job_id, candidate_id FROM recommendations WHERE recommendation_type = 'job'`);
        const existingSet = new Set(existingRecs.map(r => `${r.job_id}-${r.candidate_id}`));
        const jobInputs = jobs.map(job => ({
            title: job.title, description: job.description, skills: job.skills_required, location: job.location, job_type: job.job_type,
            salary: job.salary, remote_option: job.remote_option
        }));
        const resumeInputs = candidates.map(c => ({
            education: c.education, languages: c.languages, certifications: c.certifications, location: c.location
        }));
        const scoreMatrix = await computeSimilarityBatch(jobInputs, resumeInputs);
        const insertPromises = [];
        for (let i = 0; i < jobs.length; i++) {
            for (let j = 0; j < candidates.length; j++) {
                const jobId = jobs[i].job_id;
                const candidateId = candidates[j].candidate_id;
                const key = `${jobId}-${candidateId}`;
                if (existingSet.has(key)) continue;
                const score = scoreMatrix[i][j];
                if (score > 0) {
                    insertPromises.push(
                        db.promise().execute(`
                            INSERT INTO recommendations (user_id, job_id, candidate_id, score, recommendation_type, created_at)
                            VALUES (?, ?, ?, ?, 'candidate', NOW())
                            ON DUPLICATE KEY UPDATE score = VALUES(score), created_at = NOW()
                        `, [candidateId, jobId, candidateId, score])
                    );
                }
            }
        }
        await Promise.all(insertPromises);
        console.log("✅ Job recommendations generated (filtered by existing records).");
    } catch (error) {
        console.error("❌ Error generating job recommendations:", error);
    }
}
```

52

[52] Εικόνα 52: Κώδικας Συνάρτησης Generate Job Recommendations

Η συνάρτηση Generate Candidate Recommendations όπου δημιουργεί τις προτάσεις υποψηφίων στους εργοδότες.



```
async function generateCandidateRecommendations() {
    try {
        const [candidates] = await db.promise().query(`
            SELECT c.user_id AS candidate_id, p.location, p.education, p.languages, p.certifications
            FROM candidates c JOIN profiles p ON c.user_id = p.user_id`);
        const resumeInputs = candidates.map(c => ({
            education: c.education, languages: c.languages, certifications: c.certifications, location: c.location
        }));
        const [employers] = await db.promise().query(`SELECT user_id FROM users WHERE role = 'employer'`);
        const [existingRecs] = await db.promise().query(`SELECT user_id, job_id, candidate_id
            FROM recommendationsWHERE recommendation_type = 'candidate'`);
        const existingSet = new Set(existingRecs.map(r => `${r.user_id}-${r.job_id}-${r.candidate_id}`));
        for (const employer of employers) {
            const [jobs] = await db.promise().query(`
                SELECT j.job_id, j.title, j.description, j.location, j.skills_required, j.salary, j.job_type, j.remote_option
                FROM jobs j JOIN companies c ON j.company_id = c.company_id JOIN employers e ON c.company_id = e.company_id
                WHERE e.user_id = ?`, [employer.user_id]);
            if (jobs.length === 0) continue;
            const jobInputs = jobs.map(job => ({
                title: job.title, description: job.description, skills: job.skills_required, location: job.location,
                job_type: job.job_type, salary: job.salary, remote_option: job.remote_option
            }));
            const scoreMatrix = await computeSimilarityBatch(jobInputs, resumeInputs);
            const insertPromises = [];
            for (let i = 0; i < jobs.length; i++) {
                for (let j = 0; j < candidates.length; j++) {
                    const jobId = jobs[i].job_id;
                    const candidateId = candidates[j].candidate_id;
                    const key = `${employer.user_id}-${jobId}-${candidateId}`;
                    if (existingSet.has(key)) continue;
                    const score = scoreMatrix[i][j];
                    if (score > 0) {
                        insertPromises.push(
                            db.promise().execute(`
                                INSERT INTO recommendations (user_id, job_id, candidate_id, score, recommendation_type, created_at)
                                VALUES (?, ?, ?, ?, 'candidate', NOW())
                            `, [employer.user_id, jobId, candidateId, score])
                        );
                    }
                }
            }
            await Promise.all(insertPromises);
        }
        console.log("✅ Candidate recommendations generated (filtered by existing records).");
    } catch (error) {
        console.error("❌ Error generating candidate recommendations:", error);
    }
}
```

53

[53] Εικόνα 53: Κώδικας Συνάρτησης Generate Candidate Recommendations