ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ & ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΜΑΚΕΔΟΝΙΑΣ

ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

Διπλωματική Εργασία

Σχεδιασμός και υλοποίηση website πλατφόρμας για την διαχείριση ρομποτικών συστημάτων

Design and implementation of a web application for management of robotic systems

Καπρής Γεώργιος

Επιβλέπων καθηγητής: Δρ. Μηνάς Δασυγένης

https://arch.ece.uowm.gr/mdasyg/

Εργαστήριο Ρομποτικής, Ενσωματωμένων και Ολοκληρωμένων Συστημάτων

Κοζάνη, Φεβρουάριος 2021

Πίνακας Περιεχομένων

Λίστα εικόνων

Λίστα Πινάκων

Περίληψη

Ο όρος «ρομπότ» πρωτοεμφανίστηκε το 1921 στα πλαίσια ενός θεατρικού έργου στην Τσεχία. Από τότε ο κλάδος της τεχνολογίας έχει εξελιχθεί σε τέτοιο βαθμό που πλέον στη δευτεροβάθμια εκπαίδευση υλοποιούνται μαθήματα κατασκευής ρομποτικών συστημάτων. Ωστόσο, η πολυπλοκότητα διαχείρισης των ρομποτικών συστημάτων απαιτεί εξειδικευμένη γνώση και εξοικείωση του εκάστοτε εκπαιδευτικού με τη χρήση πολύπλοκων τεχνολογιών.

Στόχος της διπλωματικής είναι η διευκόλυνση των εκπαιδευτικών να πραγματοποιούν τα μαθήματά τους χρησιμοποιώντας ρομποτικά συστήματα, ενώ παράλληλα η χρήση αυτών να γίνεται από τους μαθητές με απλότητα και ευκολία. Σε γενικές γραμμές, ο σκοπός της έγκειται στη διευκόλυνση των εκπαιδευτικών και των μαθητών, να δίνουν εντολές εκτέλεσης σεναρίων σε ρομπότ, με μοναδική προϋπόθεση τη δυνατότητα πρόσβασης στο διαδίκτυο.

Ειδικότερα, η διπλωματική εργασία αφορά το σχεδιασμό και την υλοποίηση μιας πλατφόρμας για τη διαχείριση ρομποτικών συστημάτων, με στόχο τη δυνατότητα παροχής μιας διαδικτυακής εφαρμογής που θα εκμεταλλεύεται κάθε μέγεθος οθόνης στο μέγιστο βαθμό. Κάθε εγγεγραμμένος χρήστης θα έχει τη δυνατότητα να καταχωρεί σενάρια εκτέλεσης στην πλατφόρμα, ενώ στη συνέχεια να συνδέεται με το ρομπότ που του αντιστοιχεί και να δίνει εντολές εκτέλεσης των σεναρίων σε αυτό. Επίσης, θα παρέχονται και περαιτέρω δυνατότητες στον διαχειριστή, πάντα με γνώμονα τη διευκόλυνση του χρήστη. Πρώτον, ο διαχειριστής θα δίνει στους χρήστες σενάρια που έχει καταχωρήσει ο ίδιος στο σύστημα. Δεύτερον, αποτελεί τον υπεύθυνο για την καταχώρηση ρομπότ στο σύστημα ώστε να συνδέονται οι αντίστοιχοι χρήστες. Τρίτον, θα έχει πρόσβαση σε πίνακες επίβλεψης των ρομπότ, των σεναρίων και των χρηστών με δεδομένα κατάστασης της κάθε οντότητας.

Περιβάλλον Ανάπτυξης Διαδικτυακής εφαρμογής: Η διαδικτυακή εφαρμογή κατασκευάστηκε με τη χρήση σύγχρονων τεχνολογιών λογισμικού και καινοτόμων τεχνικών προγραμματισμού και γλωσσών προγραμματισμού ανοιχτού κώδικα. Συγκεκριμένα, πραγματοποιήθηκε η χρήση των εξής: 1) Γλώσσες προγραμματισμού: HTML5, SCSS, JavaScript(ES6, ES7, ES8, ES9, ES10, ES11), TypeScript, Angular, Angular Material, RxJs, Flex-Box, NodeJs, ExpressJS, Body Parser, Nodemon, Sequelize (ORM), MySQL 2) Τεχνικές προγραμματισμού: Reactive Programming και Object Oriented Programming. Η ανάπτυξη έγινε με τη χρήση του μεταγλωττιστή Visual Studio Code.

Abstract

The term ‘robot’ first appeared in 1921, as part of a play in the Czech Republic. Since then, technology has evolved into such an extent that robotic systems are being taught in secondary education. However, the complexity of managing robotic systems requires specialized knowledge and familiarity of the teacher with complex technologies.

This diploma thesis aims to facilitate teachers in teaching lessons to students using robotic systems, with simplicity and ease. To be more precise, the purpose is to facilitate teachers and students in giving ordered scenarios for execution to a robot, with the only prerequisite being that they have access to the internet.

The thesis concerns the design and implementation of a robotic systems management platform. The design and implementation of the information system was completed with the intention of providing an online application that will take full advantage of any screen size. In particular, each registered user will be able to upload execution scenarios on the platform and then connect to the corresponding robot to give execution commands to it. Extra features will also be provided to the administrator to make it easier for users. Firstly, they will be able to provide users with scenarios that they have uploaded into the system. Secondly, they are responsible for registering robots in the system for the users to connect. Thirdly, they will be able to access monitoring tables for the robots, scenarios and users with each entity's status data.

Web Application Development Environment: The web application was developed using state-of-the-art software technologies, innovative programming techniques and open source programming languages. Specifically, the following were used: 1) Programming languages: HTML5, SCSS, JavaScript (ES6, ES7, ES8, ES9, ES10, ES11), TypeScript, Angular, Angular Material, RxJs, Flex-Box, NodeJs, ExpressJS, Body Parser, Nodemon, Sequelize (ORM), MySQL. 2) Programming techniques: Reactive Programming and Object-Oriented Programming. The development was completed using the Visual Studio Code compiler.

Δήλωση Πνευματικών Δικαιωμάτων

Δηλώνω ρητά ότι, σύμφωνα με το άρθρο 8 του Ν. 1599/1986 και τα άρθρα 2,4,6παρ. 3 του Ν. 1256/1982, η παρούσα Διπλωματική Εργασία με τίτλο

΄Σχεδιασμός και υλοποίηση website πλατφόρμας για την διαχείριση ρομποτικών συστημάτων΄

καθώς και τα ηλεκτρονικά αρχεία και πηγαίοι κώδικες που αναπτύχθηκαν ή τροποποιήθηκαν στα πλαίσια αυτής της εργασίας και αναφέρονται ρητώς μέσα στο κείμενο που συνοδεύουν, και η οποία έχει εκπονηθεί στο Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών του Πανεπιστημίου Δυτικής Μακεδονίας, υπό την επίβλεψη του μέλους του Τμήματος κ. Μηνά Δασυγένη αποτελεί αποκλειστικά προϊόν προσωπικής εργασίας και δεν προσβάλλει κάθε μορφής πνευματικά δικαιώματα τρίτων και δεν είναι προϊόν μερικής ή ολικής αντιγραφής, οι πηγές δε που χρησιμοποιήθηκαν περιορίζονται στις βιβλιογραφικές αναφορές και μόνον. 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.

Απαγορεύεται η αντιγραφή, αποθήκευση και διανομή της παρούσας εργασίας, εξ ολοκλήρου ή τμήματος αυτής, για εμπορικό σκοπό. Επιτρέπεται η ανατύπωση, αποθήκευση και διανομή για σκοπό μη κερδοσκοπικό, εκπαιδευτικής ή ερευνητικής φύσης, υπό την προϋπόθεση να αναφέρεται η πηγή προέλευσης και να διατηρείται το παρόν μήνυμα. Ερωτήματα που αφορούν τη χρήση της εργασίας για κερδοσκοπικό σκοπό πρέπει να απευθύνονται προς τον συγγραφέα. Οι απόψεις και τα συμπεράσματα που περιέχονται σε αυτό το έγγραφο εκφράζουν τον συγγραφέα και μόνο.

Κεφάλαιο 1ο - Εισαγωγή

Η ανάπτυξη της τεχνολογίας έχει προσφέρει την ικανότητα στις ιστοσελίδες, να χρησιμοποιούνται ως κάτι περισσότερο από ένα απλό μέσο ενημέρωσης, εξελίσσοντάς τες σε ένα περιβάλλον διεργασίας. Τα τελευταία χρόνια, οι διαδικτυακές εφαρμογές έχουν κάνει την εμφάνισή τους σε κάθε τύπο επιχείρησης και οργανισμού διευκολύνοντας με αυτόν τον τρόπο, τους χρήστες τους και παράλληλα μειώνοντας το κόστος.

Διαδικτυακή εφαρμογή (*web application ή web app*) ορίζεται η εφαρμογή, η οποία είναι διαθέσιμη στους χρήστες της μέσω του Διαδικτύου (*Internet*) ή του ενδοδικτύου (*Intranet*) μίας εταιρίας ή ενός οργανισμού. Ο κάθε χρήστης για να τη χρησιμοποιήσει, χρειάζεται μόνο τον περιηγητή του. Οι εφαρμογές αυτές, ως επί το πλείστον, εκτελούνται σε ισχυρές υπολογιστικές μηχανές που έχουν τον ρόλο σταθμού εξυπηρέτησης και παρέχουν τις υπηρεσίες τους σε περισσότερους του ενός χρήστη.

Η διαφορά μεταξύ ιστοσελίδας και διαδικτυακής εφαρμογής, έγκειται στο γεγονός πως η ιστοσελίδα έχει σαν σκοπό την ενημέρωση του χρήστη προβάλλοντας, κείμενα, βίντεο και εικόνες, ενώ η διαδικτυακή εφαρμογή κατασκευάζεται για να παρέχει ένα περιβάλλον εργασίας ,στο οποίο θα μπορεί ο χρήστης να επεξεργάζεται δεδομένα και να εκτελεί διεργασίες που τον διευκολύνουν.

Η διαδικτυακή εφαρμογή έχει σαν σκοπό, να ενισχύσει τις δυνατότητες μιας ιστοσελίδας παρέχοντας περισσότερες λύσεις για εξειδικευμένες ανάγκες. Ένα τέτοιου είδους εργαλείο έχει μεγάλη ανταπόκριση, σε εταιρίες και οργανισμούς που εξελίσσονται και θέλουν να προσφέρουν περισσότερες υπηρεσίες στους πελάτες ή συνεργάτες τους. Μία από τις δυνατότητες των διαδικτυακών εφαρμογών, αποτελεί το γεγονός πως μπορούν να παρέχονται, είτε μέσω ίντερνετ στο ευρύτερο κοινό, είτε εντός κλειστού δικτύου στα πλαίσια ενδοεταιρικής εφαρμογής.

Η διπλωματική εργασία σχεδιάστηκε και υλοποιήθηκε, με σκοπό τη διευκόλυνση του διδακτικού προσωπικού στο μάθημα ρομποτικής. Η χρήση της πλατφόρμας αυτής, βοηθάει τους χρήστες να αποθηκεύουν τα σενάριά τους και να δίνουν εντολές εκτέλεσης των σεναρίων αυτών στα ρομπότ.

1.1 Περιγραφή της διδασκαλίας της ρομποτικής στη δευτεροβάθμια εκπαίδευση

Τα τελευταία χρόνια πραγματοποιούνται μαθήματα ρομποτικής, στην πρωτοβάθμια και δευτεροβάθμια εκπαίδευση. Τα μαθήματα αυτά, έχουν κεντρίσει το ενδιαφέρον της παιδικής κοινότητας και σε αρκετές περιπτώσεις υπάρχουν μαθητές που δεν μπορούν να τα παρακολουθήσουν, λόγω της πληρότητας των τμημάτων. Αυτό οφείλεται κυρίως στο γεγονός, ότι για να πραγματοποιηθεί ένα μάθημα ρομποτικής, είναι απαραίτητο ο διδάσκων να έχει ανεπτυγμένη τεχνογνωσία και να διαθέτει ικανότητες, πάνω στη διαδικασία αποσφαλμάτωσης των σεναρίων, που εκτελεί το ρομπότ.

Μέσα από τα μαθήματα ρομποτικής, το παιδί εφαρμόζει στην πράξη, ένα σύνολο επιστημών όπως την Πληροφορική και τη Μηχανολογία, αλλά και τα Μαθηματικά και τη Φυσική σε μεγάλο βαθμό, προγραμματίζοντας δικές του ρομποτικές κατασκευές, σε μια εκπαιδευτική αλλά παράλληλα ψυχαγωγική δραστηριότητα. Κύριος στόχος, ωστόσο, των μαθημάτων είναι ο μαθητής να αναπτύξει δημιουργική σκέψη, την ικανότητα επίλυσης σύνθετων προβλημάτων, καθώς και τη δεξιότητα της λεπτής κινητικότητας, μέσα από μια διαδικασία που μοιάζει περισσότερο με παιχνίδι παρά με μάθημα.

Το διδακτικό προσωπικό για να επιτύχει τους παραπάνω στόχους, χρειάζεται να είναι κοντά στον μαθητή, ώστε να εξηγεί και να επεξηγεί τους τρόπους λειτουργίας, αλλά και τις τεχνικές κατασκευής των ρομπότ. Όμως, λόγω της πολυπλοκότητας και των απαιτήσεων των συστημάτων, την περισσότερη ώρα του μαθήματος, οι εκπαιδευτικοί βρίσκονται στους δικούς τους υπολογιστές, για να κάνουν σωστή επίδειξη των λειτουργιών τους.

Τέλος, οι μαθητές που κατασκευάζουν τα ρομπότ και τα προγραμματίζουν, είτε δε διαθέτουν τη γνώση, είτε δυσκολεύονται, να εγκαταστήσουν τα λογισμικά που απαιτούνται, για την ενεργοποίηση των ρομπότ.

1.2 Ιδέα και σκοπός υλοποίησης

Η επεξήγηση που πραγματοποιήθηκε προηγουμένως, προσδιορίζει τις δυσκολίες που αντιμετωπίζει ένας διδάσκων, αλλά και οι μαθητές που παρακολουθούν το μάθημα ρομποτικής, στην εγκατάσταση και εκτέλεση των ρομποτικών σεναρίων.

Βάση αυτών, σκοπός της διπλωματικής, είναι η ανάπτυξη διαδικτυακής εφαρμογής μέσω της οποίας, οι χρήστες του συστήματος θα μπορούν να αποθηκεύουν τα ρομποτικά σενάρια που έχουν δημιουργήσει, σε έναν server και να τα διαχειρίζονται μέσω αυτής, καθώς επίσης και να δίνουν, εντολές εκτέλεσης των σεναρίων αυτών, στο ρομπότ. Ακόμη, θα παρέχονται επιπλέον δυνατότητες στο διαχειριστή του συστήματος, όπως ανάθεση σεναρίων σε χρήστες και καταχώρηση νέων ρομπότ στο σύστημα.

Οι χρήστες, θα στέλνουν εντολές εκτέλεσης και παύσης ενεργειών στα ρομποτικά συστήματα. Το προτέρημα της εφαρμογής σε σχέση με άλλα συστήματα, είναι ότι προσφέρει ευκολία χρήσης και μεγάλη προσβασιμότητα, καθώς οι μόνες προϋποθέσεις που απαιτούνται για τη χρήση της, είναι η σύνδεση στο διαδίκτυο και η εγκατάσταση περιηγητή με υποστήριξη *ECMA6 (Chrome, Mozilla, Safari, Edge).*

- Οι λειτουργίες που θα μπορούν να πραγματοποιούν οι χρήστες, είναι οι εξής:
- Εισαγωγή νέων σεναρίων
- Επεξεργασία ή διαγραφή των σεναρίων που έχουν καταχωρηθεί.
- Επεξεργασία των προσωπικών τους στοιχείων

Αντίστοιχα, οι λειτουργίες που θα παρέχονται στο διαχειριστή της εφαρμογής, θα εξασφαλίζουν περισσότερες δυνατότητες αλλά και υποχρεώσεις:

- Καταχώρηση ρομπότ στο σύστημα
- Επεξεργασία και διαγραφή ρομπότ του συστήματος
- Επεξεργασία χρηστών που έχουν εγγραφεί στο σύστημα
- Ανάθεση σεναρίων σε χρήστες

Με αυτόν τον τρόπο, η εφαρμογή, θα παρέχει ταχύτητα στη μετάδοση και εκτέλεση των σεναρίων από το ρομπότ, ενώ ταυτόχρονα θα προσφέρει ευκολία χρήσης σε ανθρώπους, που δεν είναι εξειδικευμένοι χρήστες, πάνω στον τομέα της πληροφορικής.

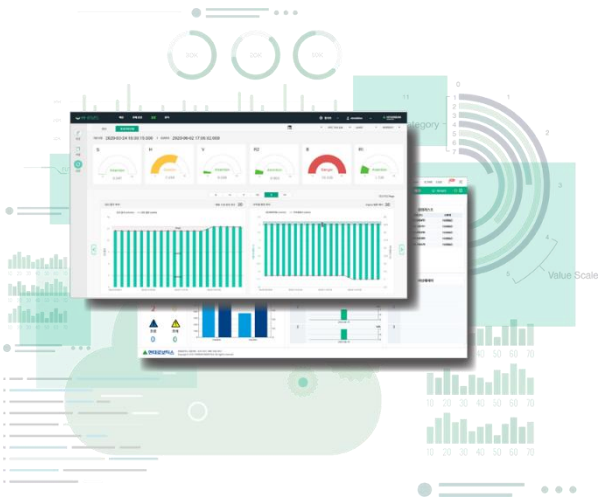
1.3 Παρόμοια πληροφοριακά συστήματα

Στη σημερινή εποχή, υπάρχουν αρκετές πλατφόρμες διαχείρισης ρομποτικών συστημάτων, οι οποίες, κυρίως αφορούν ρομποτικά συστήματα οργάνωσης αποθήκης. Στη παρούσα διαδικτυακή εφαρμογή στόχος είναι, η ευκολία και η απλότητα κατά τη χρήση του συστήματος, αφού το κοινό στο οποίο απευθύνεται, αφορά άτομα που δεν έχουν υψηλή τεχνογνωσία, όπως μαθητές και καθηγητές της δευτεροβάθμιας εκπαίδευσης.

Ένα από τα προϊόντα διαχείρισης ρομποτικών συστημάτων, είναι το *H!-RMS* από την *Hyundai Robotics [1]*. Το σύστημα διαχείρισης ρομποτικών συστημάτων τη Hyundai Robotics, αφορά τεχνολογίες λειτουργιών, *IoT* για βιομηχανίες και βασίζεται σε τεχνητή νοημοσύνη. Σκοπός του είναι, η παρακολούθηση και διάγνωση των σφαλμάτων σε ρομποτικά συστήματα, συλλέγοντας δεδομένα από τα ρομπότ, κατά την λειτουργία τους.

Κυρίως, το σύστημα αποτελείται από τρεις (3) λειτουργίες, τις λειτουργίες ελέγχου, τις λειτουργίες ανάλυσης και διάγνωσης και τις λειτουργίες συντήρησης. Οι λειτουργίες ελέγχου, παρέχουν επιλογές καταμερισμού διεργασιών στα ρομπότ, που είναι συνδεδεμένα στο σύστημα. Οι λειτουργίες ανάλυσης και διάγνωσης, περιέχουν πίνακες και διαγράμματα, που αφορούν τα σφάλματα που προκύπτουν, κατά τη διεξαγωγή διαδικασιών από τα ρομπότ. Τέλος, στις λειτουργίες συντήρησης γίνεται η διαχείριση του ιστορικού συντήρησης και προσφέρονται βασικές επιλογές αποσφαλμάτωσης, βασισμένες σε *big data* αναλύσεις.

Στην εικόνα (1) παρουσιάζονται κάποια σημεία του περιβάλλοντος χρήστη της εφαρμογής *H!-RMS* της *Hyundai*.



Εικόνα 1: Σύστημα Διαχείριση Ρομπότ H!-RMS

1.4 Σύνοψη διπλωματικής εργασίας

Η διαδικτυακή εφαρμογή ``Πλατφόρμα Διαχείρισης Ρομποτικών Συστημάτων``, αποτελείται από *Angular one-page* application και *Node Express RestAPI,* που συνδέεται σε βάση δεδομένων *MySQL*. Ο προγραμματισμός στα συστήματα, έγινε βάση τεχνικών *Object Orientation Programming* και *Reactive Programming*.

Η εφαρμογή υποστηρίζει τρία (3) είδη χρηστών. α) τον ανώνυμο χρήστη, που έχει πρόσβαση μόνο στη σελίδα εξακρίβωσης στοιχείων λογαριασμού και εγγραφής χρήστη β) τον απλό χρήστη, που έχει πρόσβαση στη σελίδα διαχείρισης σεναρίων, μέσω της οποίας, θα μπορεί να καταχωρεί σενάρια στον λογαριασμό του και να παρέχει εντολές εκτέλεσης στο ρομπότ γ) το διαχειριστή, που είναι υπεύθυνος για τη καταχώρηση των ρομπότ στο σύστημα, ώστε να αναγνωριστούν από αυτό, τη διαχείριση των χρηστών που είναι εγγεγραμμένοι στο σύστημα και να διανέμει σενάρια σε χρήστες. Ταυτόχρονα υπάρχουν και οι δυνατότητες που παρέχονται και στον απλό χρήστη.

Για τη διαχείριση, την επεξεργασία και την εκτέλεση των αναφερόμενων δυνατοτήτων, το *RestAPI* παρέχει *URL* routes, τα οποία ανάλογα με την περίπτωση χρήσης, εκτελούν τις αντίστοιχες διαδικασίες, για τη διαχείριση και μεταφορά των δεδομένων από και προς τον διακομιστή. Επιπλέον, για τη μεταφορά δεδομένων από και προς το ρομπότ έχουν δημιουργηθεί *URL* routes τα οποία χρησιμοποιούνται από το ίδιο το ρομπότ, για να λαμβάνει τα πακέτα κώδικα σεναρίου και να καταγράφεται η κατάσταση λειτουργίας του.

Η ασφάλεια που παρέχει το σύστημα, ακολουθεί *TLS Handshake Protocol*, το οποίο διασφαλίζει την πρόσβαση στο σύστημα. Με αυτόν τον τρόπο παρέχεται πρόσβαση στο σύστημα, μόνο σε clients (χρήστες/ρομπότ) δίνοντας το μοναδικό τους κλειδί μαζί με κάθε αίτημα επεξεργασίας δεδομένων (http request).

1.5 Σύνοψη 1ου κεφαλαίου

Το παραπάνω κεφάλαιο, αναφέρεται στη σχεδίαση και υλοποίηση, μίας διαδικτυακής εφαρμογής για τη διαχείριση ρομποτικών συστημάτων, στην αναγκαιότητα δημιουργίας της, καθώς και στους σκοπούς που αυτή εξυπηρετεί. Τέλος, έγινε αναφορά στη γενική δομή της διαδικτυακής εφαρμογής, καθώς επίσης στις τεχνολογίες και τεχνικές που χρησιμοποιήθηκαν κατά την εκπόνηση.

Κεφάλαιο 2ο – Θεωρητικό Υπόβαθρο

Στο κεφάλαιο αυτό, αναλύεται η διαδυκτυακή εφαρμογή που κατασκευάστηκε στο πλαίσιο της διπλωματικής εργασίας. Επιπλέον, θα αναλυθούν λεπτομερώς, όλα τα είδη τεχνολογιών που χρησιμοποιήθηκαν κατά την εκπόνηση.

2.1 Διαδίκτυο

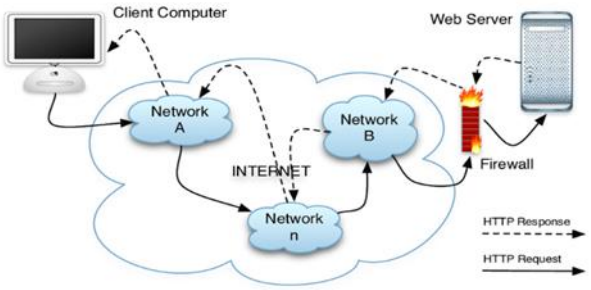
Ένα σύστημα που αποτελείται από πολυάριθμα διασυνδεδεμένα δίκτυα υπολογιστών σε παγκόσμιο επίπεδο, ονομάζεται διαδίκτυο. Η ομάδα πρωτοκόλλων, που χρησιμοποιείται, ονομάζεται "*TCP/IP*" και προσφέρει τις υπηρεσίες της σε πολυάριθμους χρήστες στον κόσμο. Το διαδίκτυο, αποτελεί ένα κοινό δίκτυο επικοινωνίας ηλεκτρονικών υπολογιστών σε όλο τον κόσμο, με το οποίο πραγματοποιείται ανταλλαγή μηνυμάτων (πακέτων) υλικού και λογισμικού επιπέδου.

Ειδικότερα, το παγκόσμιο πλέγμα διασυνδεδεμένων υπολογιστών, σε συνδυασμό με τις υπηρεσίες και τις πληροφορίες που παρέχει στους χρήστες του, περιγράφει τον όρο του Διαδικτύου.

2.1.1 Παγκόσμιος Ιστός

Μία από τις πιο διαδεδομένες και ευρέως χρησιμοποιούμενες υπηρεσίες που προσφέρει το Διαδίκτυο, είναι ο Παγκόσμιος Ιστός [2]. Το Διαδίκτυο, έχει τη δυνατότητα να αποθηκεύει την πληροφορία που δημιουργείται στον Παγκόσμιο Ιστό, μέσω των χρηστών του. Μέσω του Παγκόσμιου Ιστού διαμοιράζονται πληροφορίες και υπηρεσίες, όπως τα κοινωνικά δίκτυα και οι ηλεκτρονικές συναλλαγές.

Στην παρακάτω εικόνα (2) αποτυπώνεται η σύνδεση του Παγκόσμιου Ιστού με το Διαδίκτυο.



Εικόνα 2: Ίντερνετ και Παγκόσμιος Ιστός

Ο Παγκόσμιος Ιστός αποτελείται από πολλές ιστοσελίδες, καθεμία από τις οποίες διαθέτει την δική της μοναδική διεύθυνση, καλείται Ενιαίος Εντοπιστής Πόρων (*URL*) και έχει την εξής μορφή: *http://www.example.com*.

➢ *http*: Είναι το θεμέλιο της επικοινωνίας δεδομένων για τον Παγκόσμιο Ιστό

➢ *www*: *World Wide Web* είναι ένα πληροφοριακό σύστημα όπου τα έγγραφα και άλλοι ιστότοποι αναγνωρίζονται από το *URL*.

➢ *example*: Δηλώνει το όνομα της συγκεκριμένης ιστοσελίδας (*domain name*).

➢ *com*: Δηλώνει, είτε την περιοχή στην οποία ανήκει η ιστοσελίδα (*gr*- Ελλάδα), είτε τον τύπο της ιστοσελίδας, όπως (*com*- εμπορική ή *org*-οργανισμός).

2.1.2 Διαδικτυακή εφαρμογή

Μια διαδικτυακή εφαρμογή ή αλλιώς *Web App* ονομάζεται η εφαρμογή λογισμικού που διατίθεται στους χρήστες της, μέσω του Διαδικτύου (*Internet*) και εκτελείται σε απομακρυσμένο διακομιστή. Ορισμένες διαδικτυακές εφαρμογές χρησιμοποιούνται, για παράδειγμα, σε ενδοδίκτυο εταιρειών, σχολείων, πανεπιστημίων κτλ..

2.2 Προγραμματισμός Διαδικτύου

Η ανάπτυξη αξιόπιστων και λειτουργικών διαδικτυακών εφαρμογών, που διασφαλίζουν την ικανοποίηση των βασικών ιδιοτήτων ασφαλείας, αποτελεί ζήτημα που απασχολεί τους ειδικούς του χώρου της Τεχνολογίας Λογισμικού και της Ασφάλειας Πληροφοριών. Μέσω της δημιουργίας κώδικα που γράφεται στις μέρες μας, δίνεται η δυνατότητα δικτυακής σύνδεσης στις εφαρμογές, με αποτέλεσμα, οι χρήστες να μπορούν να χρησιμοποιούν απομακρυσμένες υπηρεσίες μέσω του Διαδικτύου [3].

Για τη δημιουργία μίας ιστοσελίδας, απαιτείται ο προγραμματισμός του δημόσιου τμήματος (*front-end*) και η ανάπτυξη της διαχείρισης (*back-end*). Ένας προγραμματιστής δημοσίου τμήματος, (*front-end*) λαμβάνει τον εικαστικό σχεδιασμό μιας ιστοσελίδας και τον αναπτύσσει σε κώδικα. Αυτός ο προγραμματιστής χρησιμοποιεί *HTML* για τη δομή της ιστοσελίδας, *CSS* για την υπαγόρευση της διάταξης και του οπτικού στυλ καθώς και JavaScript. Η ανάπτυξη μίας ιστοσελίδας ή εφαρμογής (*back-end*), ασχολείται με τη διασφάλιση της βάσης δεδομένων από

ηλεκτρονικές απειλές και διαχειρίζεται τα δεδομένα που λαμβάνει ο χρήστης. Ένας προγραμματιστής διαχείρισης κώδικα (*back-end*), εστιάζει στη λειτουργικότητα της Ιστοσελίδας.

2.3 Τεχνικές Προγραμματισμού

Στη δημιουργία μιας διαδικτυακής εφαρμογής, ο προγραμματιστής κατά το αρχικό στάδιο κατασκευής, δεν μπορεί να γνωρίζει το πλήθος των αρχείων και γραμμών κώδικα που θα κληθεί να δημιουργήσει, με αποτέλεσμα στο τελικό στάδιο να καταλήγει σε μια ιστοσελίδα χωρίς δομή, δύσκολη στη συντήρηση και την επεκτασιμότητα της. Για το λόγο αυτό, έχουν οριστεί τεχνικές και πλαίσια κατασκευής, που αν τα ακολουθήσει κανείς, καταλήγει σε ένα τελικό αποτέλεσμα που μπορεί εύκολα να συντηρηθεί και να επεκταθεί, καθώς και να διαθέτει υψηλή ασφάλεια.

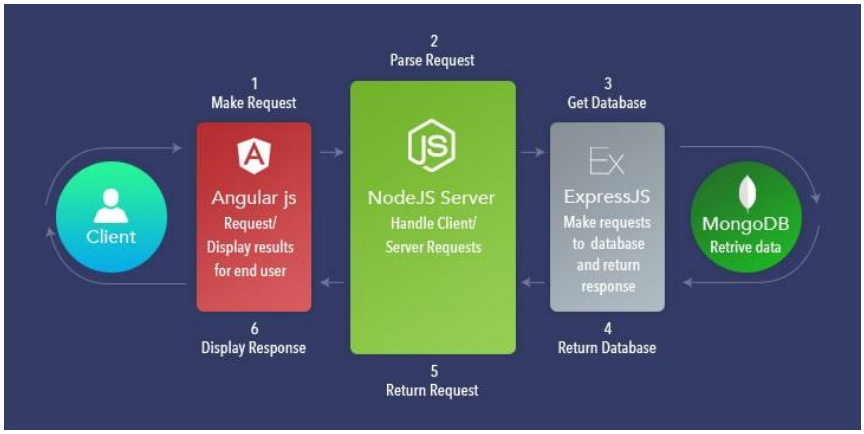
2.3.1 MEAN Αρχιτεκτονική

Για την ολοκληρωμένη κατασκευή μιας διαδικτυακής εφαρμογής, απαιτείται ανάπτυξη όλων τμημάτων που την αποτελούν. Ξεκινώντας από τη βάση δεδομένων και το διακομιστή στο ‘πίσω μέρος’ της εφαρμογής (*back-end*), έως και το περιβάλλον του χρήστη, ως προς τη διεπαφή της εφαρμογής (*front-end*), απαιτείται η ανάπτυξη υπολογιστικών συστημάτων που είναι αλληλένδετα καθώς και η παροχή σύγχρονων λύσεων.

Μία από τις πιο σύγχρονες και διαδεδομένες αρχιτεκτονικές κατασκευής διαδικτυακών εφαρμογών είναι η αρχιτεκτονική *MEAN* [4]. Η αρχιτεκτονική αυτή υποστηρίζει τις εξής τεχνολογίες:

1. *MongoDB* – βάση δεδομένων
2. *Express Framework* – *Node.Js* αρχιτεκτονική διακομιστή
3. *Angular* – Αρχιτεκτονική κατασκευής διεπαφής (*front-end framework*)
4. *Node.Js* – *API Web service*

Στο πλαίσιο υλοποίησης της διπλωματικής εργασίας, αντί για *MongoDB* χρησιμοποιήθηκε βάση δεδομένων *MySQL* με το *ORM* (*Object Relational Mapping*) της *Sequelize*.



Εικόνα 3: MEAN Αρχιτεκτονική

Στην εικόνα (3) αποτυπώνεται γραφικά, ο τρόπος με τον οποίο πραγματοποιείται η μετάδοση δεδομένων. Αναλυτικότερα, ακολουθείται η εξής διαδικασία:

Όταν ο χρήστης δημιουργεί ένα αίτημα, αυτό επεξεργάζεται από την *Angular* στο περιβάλλον του χρήστη.

Το αίτημα του χρήστη, επεξεργάζεται και δρομολογείται από τη *Node JS*, βάση των *API* routes που έχουν δημιουργηθεί.

Μέσω της *Express JS* και της βιβλιοθήκης *Sequelize* γίνεται η συλλογή των δεδομένων από τη βάση δεδομένων *MySQL*.

Η *Express JS* δέχεται τα δεδομένα από τη βάση και τα μεταδίδει στη *Node JS*

Η *Node JS* στέλνει *JSON* αρχείο στην *Angular* που περιέχει τα δεδομένα.

Ένα από τα μεγαλύτερα οφέλη της *MEAN* αρχιτεκτονικής, είναι πως ο κώδικας για το διακομιστή και για την ιστοσελίδα μπορεί να γραφεί χρησιμοποιώντας μόνο *JavaScript/Typescript*. Ακόμη, δίνεται η δυνατότητα υποστήριξης *MVC* αρχιτεκτονικής, προσφέροντας έτσι πολλαπλές επιλογές κατά την κυκλοφορία της εφαρμογής. Τέλος, καθώς οι τεχνολογίες που χρησιμοποιούνται είναι ανοιχτού κώδικα, γίνονται συχνά ενημερώσεις.

2.3.2 Node.JS

Η *Node.JS* είναι ένα περιβάλλον εκτέλεσης *Javascript* ανοιχτού κώδικα, που υποστηρίζεται από πολλαπλές πλατφόρμες και μπορεί να εκτελέσει *back-end* διεργασίες. Η *Node.JS* επιτρέπει στον προγραμματιστή, να δημιουργεί εργαλεία γραμμής, εντολών και δέσμες ενεργειών (*scripts*), στην πλευρά του διακομιστή. Με αυτόν τον τρόπο, δίνεται η δυνατότητα παραγωγής δυναμικού περιεχομένου, για κάθε ιστοσελίδα.

Ακόμη, διαθέτει αρχιτεκτονική που βασίζεται σε γεγονότα (*event-drive architecture*), προσφέροντας τη δυνατότητα ασύγχρονων διαδικασιών εισόδου και εξόδου δεδομένων. Ακόμη, επιτρέπει τη δημιουργία διακομιστών *Web* και εργαλείων δικτύωσης χρησιμοποιώντας *Javascript* και μία συλλογή από ΄ενότητες΄ (*modules*) που χειρίζονται διάφορες βασικές λειτουργίες. Τα modules της *Node.JS* χρησιμοποιούν ένα *API*, που έχει σχεδιαστεί, με σκοπό να μειώσει την πολυπλοκότητα δημιουργίας εφαρμογών διακομιστή.

Κάποιες από τις βασικές λειτουργίες που παρέχουν τα module είναι:

1. Εισαγωγή αρχείων στο σύστημα
2. Δικτύωση *(DNS, HTTP, TCP, TLS/SSL και UDP)*
3. Επεξεργασία δυαδικών δεδομένων (*buffer*)
4. Συναρτήσεις κρυπτογράφησης
5. Ροές δεδομένων

Στην εικόνα (4) παρουσιάζονται οι δυνατότητες που προσφέρει η *Node JS.*

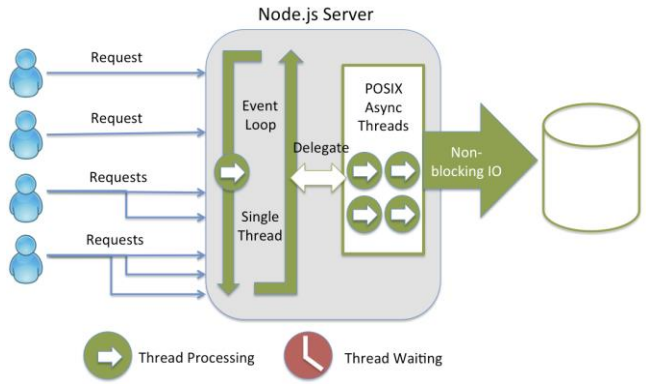
Εικόνα 4: Δυνατότητες Node.JS

Επίσης, χρησιμοποιείται κυρίως για τη δημιουργία προγραμμάτων δικτύου, όπως διακομιστές *Web*. Η μεγαλύτερη διαφορά μεταξύ *Node.JS* και *PHP*, είναι ότι οι περισσότερες διαδικασίες σταματούν, έως ότου ολοκληρωθούν κάποιες ενέργειες, σε αντίθεση με τη *Node.JS* όπου οι διαδικασίες δε διακόπτονται, αλλά εκτελούνται παράλληλα, χρησιμοποιώντας callbacks για να σηματοδοτήσουν την ολοκλήρωση ή αποτυχία μιας διαδικασίας.

Όσο αναφορά τη λειτουργία της *Node.JS* είναι *single-thread event loop*, με κλήσεις εισόδου/εξόδου χωρίς αποκλεισμούς, επιτρέποντας την υποστήριξη σε δεκάδες χιλιάδες ταυτόχρονες συνδέσεις, χωρίς να επιβαρύνεται με κόστος αλλαγής *thread*. Ο σχεδιασμός της κοινής χρήσης ενός νήματος μεταξύ όλων των αιτημάτων που χρησιμοποιούν το μοτίβο του παρατηρητή (*observer*), προορίζεται για την κατασκευή πολλών ταυτόχρονων εφαρμογών, όπου οποιαδήποτε λειτουργία εκτελεί I / O πρέπει να χρησιμοποιεί *callbacks*. Η υλοποίηση του *single-thread event loop*, πραγματοποιείται, χρησιμοποιώντας τη βιβλιοθήκη *libuv*, η οποία με τη σειρά της χρησιμοποιεί μία ομάδα νημάτων σταθερού μεγέθους, που χειρίζεται ορισμένες από τις μη αποκλεισμένες ασύγχρονες λειτουργίες εισόδου/εξόδου. Για την εκτέλεση παράλληλων διεργασιών, υπεύθυνη χρίζεται μία ομάδα νημάτων που θα τις διαχειρίζεται [5].

Το *thread* κύριας λειτουργίας, καλεί τα post request στην ουρά των κοινόχρηστων εργασιών και στη συνέχεια, καθορίζει και εκτελεί το κάθε *thread* στο *thread pool*. Ενσωματωμένες λειτουργίες συστήματος μη αποκλεισμού, όπως η δικτύωση, μεταφράζονται ως *kernel-side non-blocking sockets*, ενώ οι λειτουργίες συστήματος μπλοκαρίσματος, όπως τα αρχεία I / O, εκτελούνται με τρόπο αποκλεισμού στα δικά τους νήματα. Όταν ένα νήμα από την ομάδα νήματος ολοκληρώνει μία εργασία, ενημερώνεται το κύριο νήμα αυτού, το οποίο με τη σειρά του ενεργοποιείται και εκτελεί το καταχωρημένο *callback*.

Στην εικόνα (5) παρουσιάζεται ο τρόπος διαχείρισης των αιτημάτων μετάδοσης δεδομένων.



Εικόνα 5: Διαχείριση αιτημάτων σύνδεσης

Ένα μειονέκτημα της *sigle-threaded* προσέγγισης, είναι ότι η *Node.JS*, δεν επιτρέπει την κατακόρυφη κλιμάκωση που επιτυγχάνεται, αυξάνοντας τον αριθμό πυρήνων του επεξεργαστή του μηχανήματος που λειτουργεί. Αυτό μπορεί να επιτευχθεί χρησιμοποιώντας κάποιο επιπρόσθετο module, όπως *StrongLoop, Process Manager* ή *pm2*. Παρόλα αυτά, οι προγραμματιστές, έχουν τη δυνατότητα να αυξήσουν τον προεπιλεγμένο αριθμό νημάτων που χρησιμοποιούνται από το *thread pool* του *libuv*. Αξίζει να σημειωθεί, πως το λειτουργικό σύστημα του διακομιστή έχει τη δυνατότητα να διανείμει τα *thread* σε πολλαπλούς πυρήνες.

2.3.2 Angular Framework / Modular Αρχιτεκτονική

Το πλαίσιο ιστού της *Angular*, χρησιμοποιώντας τους decorators, καταφέρνει να συνδυάζει τα προτερήματα και από τις γλώσσες προγραμματισμού *HTML, Typescript* και *SCSS*. Μία σελίδα της *Angular* αποτελείται από, components, τα οποία συνδέονται μέσω των modules. Κάθε component αποτελείται από αρχεία *HTML*, αρχεία *Typescript* και αρχεία *SCSS*, στα οποία μπορούμε να εφαρμόζουμε διαφορετική λογική, ανάλογα με τη χρήση.

Στην εικόνα (6) παρουσιάζονται τα προτερήματα του *Angular framework*

Εικόνα 6: Χαρακτηριστικά της Angular

Κατά την εγκατάσταση του *Angular framework* γίνεται και εγκατάσταση του *Angular CLI*. Το *Angular CLI*, είναι ένα εργαλείο διεπαφής με τη κονσόλα τερματικού που χρησιμοποιείται για την αρχικοποίηση, την ανάπτυξη και τη δημιουργία σκελετού προγράμματος, καθώς και τη συντήρηση μιας *Angular* εφαρμογής, μέσω εντολών στην τερματική κονσόλα [6].

Στην *Angular*, για να μπορεί κανείς να χρησιμοποιεί κάποιο *component*, χρειάζεται να το δηλώσει στο *module*. Δίνεται η δυνατότητα χρήσης πολλαπλών *modules*, σε μια διαδικτυακή εφαρμογή με *Angular*, όμως απαιτείται η δήλωσή τους στο κεντρικό *module* της εφαρμογής.

Ένα *module*, θεωρείται ως ένα καθορισμένο κομμάτι κώδικα, που βρίσκεται στο σύστημα. Το πόσο καλά ορίζεται, εξαρτάται από το πως έχουν χωριστεί οι τομείς της εφαρμογής. Θεωρητικά, είναι εφικτό να συμπεριληφθεί όλος ο κώδικας σε ένα *module*, αλλά αυτός δε θεωρείται ως ένας αποτελεσματικός τρόπος. *Modular programming*, είναι τεχνική που σχετίζεται με το διαχωρισμό της λειτουργικότητας σε ανεξάρτητα μέρη. Με αυτόν τον τρόπο, μπορεί να κατασκευάσει κανείς διάφορα μέρη ξεχωριστά και ανάλογα το τρόπο που θα τα ενώσει, να δημιουργήσει το τελικό αποτέλεσμα [5].

Στην *Angular* υπάρχουν δυο τύποι *modules*, τα *root modules* και *feature modules*. Ωστόσο, μπορεί να αναλύσει κανείς περαιτέρω τα *feature modules*, σε πιο εξειδικευμένα μέρη, όπως *routing module*, *component* και *services*. Γενικά, όλα τα *modules* κατασκευάζονται με τον ίδιο τρόπο. Η κύρια διαφορά έγκειται στο σκοπό ύπαρξής τους, ο οποίος καθορίζει το μέγεθος και τη λειτουργία τους.

Στα πλαίσια της διπλωματικής έχουν κατασκευαστεί πέντε κύρια module:

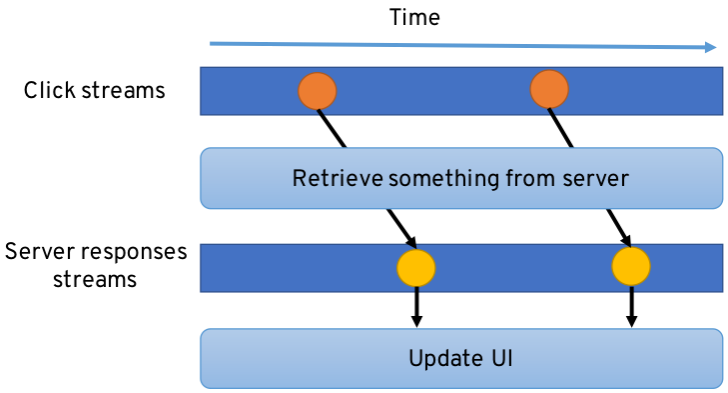
1 ***Main Module***: Είναι το κεντρικό *module*, που περιλαμβάνει όλα τα υπόλοιπα *modules* και βάσει αυτού καθορίζονται βασικές λειτουργίες, που θα λειτουργούν σε κάθε περίπτωση χρήσης της εφαρμογής.
2 ***Routing Modules***: Γίνεται καθορισμός των *url* της εφαρμογής και ορίζεται, η αντιστοίχιση των *component* με τα *url*. Για κάθε module μπορεί να οριστεί ξεχωριστό *routing module*.
3 ***Domain Module***: Αναφέρεται σε μία συγκεκριμένη λειτουργία χρήσης, που επαναλαμβάνεται σε διαφορετικές λειτουργικές ενότητες. Για παράδειγμα, μία φόρμα εγγραφής χρήστη μπορεί να χρησιμοποιηθεί και ως φόρμα επισκόπησης προφίλ. Σε αυτές τις περιπτώσεις, μπορεί κανείς να ξεχωρίσει τον κώδικα σε *component*, ώστε να το χρησιμοποιήσει σε πολλαπλές ενότητες.
4 ***Service Module***: Είναι υπεύθυνο για τη διαχείριση δεδομένων. Κυρίως, βασίζεται σε βοηθητικά προγράμματα και χρησιμοποιείται στη σύνδεση της εφαρμογής με πηγές δεδομένων. Ένα παράδειγμα είναι, πως εάν επιθυμεί κανείς να υποβάλει τα στοιχεία που εισάγει ο χρήστης μιας φόρμας σε κάποιο εξωτερικό *API*, αυτή η διαδικασία θα υλοποιηθεί από το *service module*.

Τέλος, ένα από τα μεγαλύτερα προτερήματα που προσφέρει η *Angular*, είναι τα *schematics* και η ευκολία εγκατάστασης εξωτερικών βιβλιοθηκών. Τα *Schematics*, είναι εντολές που χρησιμοποιούνται στην κονσόλα τερματικού και παράγουν αρχεία με κώδικα, που μπορούν να χρησιμοποιηθούν απευθείας στην εφαρμογή. Όσον αφορά τις εξωτερικές βιβλιοθήκες, αυτές αποτελούν πακέτα κώδικα, που παρέχουν μεθόδους και custom elements που μπορεί κανείς να

χρησιμοποιήσει, ώστε να γίνει πιο γρήγορη η κατασκευή της εφαρμογής. Ένα παράδειγμα που συνδυάζει αυτά τα δύο προτερήματα, είναι η *Angular Material*.

2.3.5 Reactive Programming / Αντιδραστικός Προγραμματισμός

Ο αντιδραστικός προγραμματισμός, είναι ο προγραμματισμός, που βασίζεται σε συμβάντα (*events*), αντί της σειράς των γραμμών στον κώδικα. Συνήθως αυτό περιλαμβάνει περισσότερα από ένα συμβάντα και αυτά γίνονται με μία σειρά με την πάροδο του χρόνου. Για την κατασκευή μιας ιστοσελίδας με αντιδραστικό προγραμματισμό, χρησιμοποιείται η βιβλιοθήκη *RxJs*. Σε αυτό το σημείο, χρειάζεται να αναφερθεί πως το *Angular framework*, έχει βασιστεί στη βιβλιοθήκη για πολλά από τα χαρακτηριστικά της.



Εικόνα 7: Τρόπος λειτουργίας Reactive Event

Η *RxJs* ,είναι μία βιβλιοθήκη που χρησιμοποιείται για τη σύνταξη ασύγχρονων προγραμμάτων βάσει *event*, χρησιμοποιώντας *observable sequences*. Παρέχει έναν τύπο πυρήνα, τα *Observables*, που αποτελούνται, από τα *satellite types* (*Observer, Schedulers, Subjects*) και τους *operators*, που είναι εμπνευσμένα από τα *Arrayextras* (*map, filter, reduce, every, etc)*, ώστε να είναι δυνατή η διαχείριση των ασύγχρονων *event*.

Οι βασικές έννοιες της *RxJs*, που χρησιμοποιούνται για τη διαχείριση των ασύγχρονων event είναι:

- ***Observable***: Αντιπροσωπεύει την επίκληση συλλογής μελλοντικών αξιών ή γεγονότων.
- ***Observer***: Είναι μία συλλογή με *callbacks*, που αναγνωρίζουν τις τιμές που παραδόθηκαν από *Observable*.
- ***Subscription***: Αντιπροσωπεύει την εκτέλεση ενός *Observable*, είναι κυρίως χρήσιμο για την ακύρωση της εκτέλεσης.

- ***Operators***: Είναι λειτουργίες που επιτρέπουν *functional* προγραμματισμό, ώστε να γίνει διαχείριση των δεδομένων. Μερικές λειτουργίες είναι *map, filter, concat, reduce*, κτλ.

- ***Subject***: Είναι ισοδύναμο με το *EventEmitter* και ο μόνος τρόπος όπου είναι εφικτή η πολλαπλή μετάδοση μιας τιμής ή συμβάντος σε πολλούς *Observer*.

Κυρίως η *RxJs*, χρησιμοποιείται για τη λήψη δεδομένων από κάποιο διακομιστή, διότι σε τέτοιου είδους περιπτώσεις, δεν είναι γνωστό ποια χρονική στιγμή θα σταλθούν τα δεδομένα στο χρήστη. Με αυτόν τον τρόπο, η σελίδα του χρήστη μπορεί να συνεχίσει να είναι λειτουργική, ώστε να παρέχει καλύτερη εμπειρία σε εκείνον.

```
addHero(hero: Hero): Observable<Hero> {
  return this.http.post<Hero>(this.heroesUrl, hero, this.httpOptions).pipe(
    tap((newHero: Hero) => this.log(`added hero w/ id=${newHero.id}`)),
    catchError(this.handleError<Hero>('addHero'))
  );
}
```

Εικόνα 8: Διαδικασία αίτησης δεδομένων από το API

```
add(name: string): void {
  name = name.trim();
  if (!name) { return; }
  this.heroService.addHero({ name } as Hero)
    .subscribe(hero => {
      this.heroes.push(hero);
    });
}
```

Εικόνα 9: Διαδικασία επεξεργασίας δεδομένων όταν ληφθούν δεδομένα από το API

2.4 Γλώσσες Προγραμματισμού Διαδικτύου

Οι τεχνολογίες που χρησιμοποιήθηκαν για την ανάπτυξη της διαδικτυακής εφαρμογής είναι, το πλαίσιο ιστού (*framework*) της *Angular 10*, για τη κατασκευή του δημοσίου τμήματος (*front-end*) και το πλαίσιο ιστού *Node Express* για την ανάπτυξη του διαχειριστικού συστήματος (*back-end*), το οποίο συνδέεται με βάση δεδομένων *MySQL*. Το πλαίσιο της *Angular 10*, συνδυάζει τις τεχνολογίες της *HTML*, *Typescript* και *SCSS* ώστε να επιταχύνει τη παραγωγή κώδικα και κάνοντας *compile* εξάγει αρχεία *HTML*, *CSS* και *Javascript*, τα οποία θα χρησιμοποιηθούν για την εμφάνιση της σελίδας στο παγκόσμιο ιστό [7]. Το πλαίσιο της *Node Express*, χρησιμοποιεί *Javascript* για τη δημιουργία και διαχείριση της βάσης δεδομένων *MySQL*.

2.4.1 HTML

Μία από τις κύριες γλώσσες προγραμματισμού είναι η *HTML* (*HyperText Markup Language*), η οποία αποτελεί γλώσσα σήμανσης, με τα βασικά δομικά στοιχεία των ιστοσελίδων (*head, body, footer*). Η *HTML* γράφεται υπό μορφή στοιχείων *HTML*, τα οποία αποτελούνται από ετικέτες (*tags*), που περικλείονται μέσα σε σύμβολα «μεγαλύτερο από» και «μικρότερο από» (για παράδειγμα <html>), μέσα στο περιεχόμενο της ιστοσελίδας. Οι ετικέτες *HTML* συνήθως λειτουργούν ανά ζεύγη (για παράδειγμα <h1> και </h1>), με την πρώτη να ονομάζεται «ετικέτα έναρξης» και τη δεύτερη «ετικέτα λήξης» (ή σε άλλες περιπτώσεις «ετικέτα ανοίγματος» και «ετικέτα κλεισίματος» αντίστοιχα). Οι σχεδιαστές των ιστοσελίδων, μπορούν να προσθέσουν κάποιο κείμενο, πίνακα ή εικόνα ανάμεσα στις ετικέτες.

Ο Περιηγητής Ιστού (*web browser*), μετατρέπει το περιεχόμενο των εγγράφων της *HTML* σε σελίδες, που μπορεί κάποιος να διαβάσει ή και να ακούσει. Οι ετικέτες της *HTML* χρησιμοποιούνται για την παρουσίαση του περιεχομένου μίας σελίδας.

Με τη γλώσσα προγραμματισμού *HTML* και τα βασικά δομικά στοιχεία της, δημιουργείται ένας ιστότοπος. Στην *HTML* επιτρέπεται η ενσωμάτωση εικόνων, διάφορων αντικειμένων αλλά και η εμφάνιση διαδραστικών φορμών μέσα στη σελίδα. Μέσα από την *HTML*, υπάρχει η δυνατότητα δημιουργίας εγγράφων, με σημαντικά δομικά στοιχεία του περιεχομένου τους. Επιπλέον, εμφανίζεται ο κώδικας, ο οποίος είναι απαραίτητος για τη μορφοποίηση του περιεχομένου (ενός κειμένου), όπως παράγραφοι, λίστες, κεφαλίδες κ.ά. Τέλος, για να μετατραπεί μία σελίδα *HTML* από στατική σε δυναμική, δηλαδή να λαμβάνει και να επεξεργάζεται δεδομένα, χρειάζεται η ενσωμάτωση σεναρίων εντολών, σε γλώσσες προγραμματισμού όπως *JavaScript*, *PHP* κ.ά.

Η *Angular*, χρησιμοποιεί την *HTML* για την κατασκευή της δομής του component. Μέσω μηχανισμών όπως *data binding*, *string interpolation* και *two-way binding* δίνεται η δυνατότητα να κάνει κανείς αναφορές παραμέτρων, που χρησιμοποιούνται στα *Typescript* αρχεία του component. Με αυτόν τον τρόπο, η σελίδα παράγεται δυναμικά, αυξάνοντας χαρακτηριστικά όπως, η επεκτασιμότητα και η συντήρηση. Τέλος, μέσω των custom element που προσφέρονται από διάφορες βιβλιοθήκες της *Angular*, ένας προγραμματιστής μπορεί να δημιουργήσει εντυπωσιακό περιβάλλον για τον χρήστη, με μεγάλη ευκολία.

Στις παρακάτω εικόνες (10), (11) παρουσιάζονται οι τεχνικές αναφοράς από HTML σε Typescript παραμέτρους.

```
22   title = 'Basics with Angular';
23
24   onClick() {
25     console.log('Hello Event');
26   }
```

Εικόνα 11: Κώδικας στη Typescript

```
1  <!-- String Interpolation -->
2  <h1>
3    {{ title }}
4  </h1>
5
6  <!-- Event Binding -->
7  <button (click)="onClick()">Click me</button>
8
9  <!-- Property Binding -->
10 <h1 [innerText]="title"></h1>
11
```

Εικόνα 10: Μέθοδοι αναφοράς μεταβλητών στην HTML

2.4.2 SCSS/SASS

Η γλώσσα προγραμματισμού *Syntactically Awesome Style Sheets (SASS)*, είναι μία γλώσσα *scripting* προ-επεξεργασίας, που ερμηνεύεται ή μεταφράζεται σε *Cascading Style Sheets (CSS)*. Η σύνταξη που ακολουθεί είναι η *SCSS* (*Sassy CSS*) και χρησιμοποιεί μορφοποίηση μπλοκ, όπως και η *CSS*. Χρησιμοποιεί αγκύλες, για να δηλώσει μπλοκ κώδικα και ερωτηματικά, για να διαχωρίσει τους κανόνες μέσα σε ένα μπλοκ. Τα αρχεία σύνταξης και *SCSS* με εσοχή, έχουν παραδοσιακά τις επεκτάσεις *.scss*.

Η *CSS3*, αποτελείται από μία σειρά επιλογών και ψευδοεπιλογών, που ομαδοποιούν τους κανόνες που ισχύουν γι' αυτούς. Η *Sass* επεκτείνει τη *CSS*, παρέχοντας διάφορους μηχανισμούς, διαθέσιμους σε πιο παραδοσιακές γλώσσες προγραμματισμού, ιδιαίτερα σε αντικειμενοστραφείς γλώσσες, αλλά και μη διαθέσιμους στην ίδια τη *CSS3*. Όταν ερμηνεύεται το *SassScript*, δημιουργείται μπλοκ κανόνων *CSS* για διάφορους selector, όπως ορίζονται από το αρχείο Sass. Ο διερμηνέας *Sass*, μεταφράζει το *SassScript* σε *CSS*. Εναλλακτικά, το Sass μπορεί να παρακολουθεί το αρχείο *.scss* και να το μεταφράζει σε αρχείο εξόδου *.css*, όποτε αποθηκεύεται το αρχείο *.scss*.

Η σύνταξη με εσοχή, είναι μία μεταγλώσσα. Η *SCSS* αποτελεί μία ένθετη μεταγλώσσα, καθώς η έγκυρη *CSS* είναι η έγκυρη *SCSS*, με την ίδια έννοια. Η *SassScript* παρέχει τους εξής μηχανισμούς για τη σύνταξή της: μεταβλητές, *nesting*, *mixins*, και κληρονομικότητα *selector* [8].

Στην εικόνα (12) παρουσιάζεται παράδειγμα σύνταξης στην SCSS.

```
3   @import "~@angular/material/theming";
4
5   @include mat-core();
6
7   $primary: mat-palette($mat-blue, 700);
8   $accent: mat-palette($mat-indigo, 500);
9   $warn: mat-palette($mat-red, 600);
10
11  $theme: mat-light-theme($primary, $accent, $warn);
12  .list-item-active {
13    font-weight: bold;
14    color: mat-color($accent, darker) !important;
15  }
```

Εικόνα 12: Παράδειγμα κώδικα SCSS

2.4.3 Typescript

Η *TypeScript*, είναι μια *strongly-typed*, αντικειμενοστραφής, μεταγλωττισμένη γλώσσα, που σχεδιάστηκε από τη *Microsoft*. Η *TypeScript*, εκτός από γλώσσα, αποτελεί και ένα σύνολο εργαλείων. Γενικά, είναι ένα τυπικό υπερσύνολο της *JavaScript*, που έχει μεταγλωττιστεί σε *JavaScript*. Με άλλα λόγια, η *TypeScript* είναι *JavaScript* συν μερικές επιπλέον δυνατότητες. Η *Typescript*, θεωρείται καλύτερη σε σχέση με άλλες αντίστοιχες γλώσσες, όπως η *Coffescript* ή *Dart*, καθώς αυτή είναι επέκταση της *Javascript*. Αντίθετα, οι υπόλοιπες γλώσσες χρειάζονται εξειδικευμένο περιβάλλον για να εκτελεστούν [9].

Η *Typescript* περιέχει και άλλα προτερήματα:

1. Μεταγλώττιση/*Compilation* – Καθώς η *Javascript* είναι *interpreted* γλώσσα, τα σφάλματα του κώδικα, δεν πρόκειται να εμφανιστούν μέχρι να γίνει η εκτέλεσή του, με αποτέλεσμα να γράφεται κώδικας χωρίς να φέρει αποτελέσματα, αναγκάζοντας τον μηχανικό, να εκτελεί τον κώδικά του για κάθε αλλαγή που γίνεται. Στην *Typescript* προσφέρεται *transpiler*
2. Ισχυρά στατική σύνταξη/*Strong Static Typing* – Η *Javascript* δεν έχει *strongly type* χαρακτηριστικά. Η *Typescript* παρέχει προαιρετικό σύστημα *strongly type* και *inference type* με *TLS* (*Typescript Language Service*). Ο τύπος μιας μεταβλητής, μπορεί να οριστεί από το *TLS* μέσω της τιμής που ανατέθηκε.

3. Η *Typescript*, υποστηρίζει ορισμούς τύπων για υπάρχουσες βιβλιοθήκες *Javascript*. Το αρχείο *Typescript Definition* (με επέκταση .ts), παρέχει ορισμό για εξωτερικές βιβλιοθήκες *Javascript*. Ως εκ τούτου, ο κώδικας *Typescript* μπορεί να περιέχει αυτές τις βιβλιοθήκες.
4. Η *TypeScript* υποστηρίζει έννοιες αντικειμενοστραφής προγραμματισμού, όπως τάξεις, διεπαφές (*interfaces*), κληρονομικότητα (*Inheritance*) κ.λπ.

Στην εικόνα (13) παρουσιάζεται παράδειγμα τμήματος κώδικα γραμμένος σε *Typescript*

```
8  export class BasicsComponent {
9    @Input() name: string;
10   @Output() liked = new EventEmitter();
11
12   constructor() {}
13
14   title = 'Basics with Angular';
15
16   onClick() {
17     console.log('Hello Event');
18   }
```

Εικόνα 13: Παράδειγμα κώδικα Typescript

2.4.5 Angular Material

Η *Angular Material* είναι μια βιβλιοθήκη στοιχείων *UI*, που απευθύνεται σε προγραμματιστές της *Angular* και υποστηρίζεται από το *Angular CLI*. Τα component που παρέχονται από την *Angular Material*, βοηθούν στην κατασκευή φιλικών προς τον χρήστη, λειτουργικών ιστοσελίδων και διαδικτυακών εφαρμογών. Ακόμη, τηρούν τις σύγχρονες αρχές σχεδιασμού ιστοσελίδων, όπως η φορητότητα του προγράμματος περιήγησης, η ανεξαρτησία της συσκευής και ο ασφαλής διαχωρισμός. Με αυτόν τον τρόπο, βοηθά στη γρήγορη δημιουργία ελκυστικών και *responsive* ιστοσελίδων [10].

Κάθε στοιχείο της *Angular Material* αποτελείται από *HTML*, *CSS* και σε ορισμένες περιπτώσεις, συνοδεύει κώδικα *Typescript*. Επίσης, επεκτείνει τη λειτουργικότητα ορισμένων υφιστάμενων στοιχείων διεπαφής, όπως για παράδειγμα, μία λειτουργία αυτόματης συμπλήρωσης για πεδία εισαγωγής. Επίσης, παρέχονται επιπλέον *elements*, τα οποία διευκολύνουν τον προγραμματιστή να κατασκευάσει την ιστοσελίδα. Τα *elements*, εκμεταλλεύονται τους μηχανισμούς *two-way binding*, *interpolation* και *event binding*, δίνοντας επιπλέον επιλογές στον προγραμματιστή.

Στην εικόνα (14) παρουσιάζεται ο τρόπος σύνταξης ενός πεδίου εισόδου χρησιμοποιώντας *Material Inputs*.

```
<mat-form-field class="example-form-field">
  <mat-label>Clearable input</mat-label>
  <input matInput type="text" [(ngModel)]="value">
  <button mat-button *ngIf="value" matSuffix mat-icon-button aria-label="Clear" (click)="value=''">
    <mat-icon>close</mat-icon>
  </button>
</mat-form-field>
```

Εικόνα 14: Παράδειγμα χρήσης Material elements

2.4.6 MySQL / Sequelize

Η *MySQL*, είναι ένα σχεσιακό σύστημα διαχείρισης βάσεων δεδομένων, που βασίζεται σε *SQL - Structured Query Language* [11]. Η εφαρμογή, χρησιμοποιείται για ένα μεγάλο φάσμα σκοπών, συμπεριλαμβανομένης της αποθήκευσης δεδομένων, του ηλεκτρονικού εμπορίου και των εφαρμογών καταγραφής. Η πιο συνηθισμένη χρήση για τη *MySQL*, είναι για τους σκοπούς μιας διαδικτυακής βάσης δεδομένων. Μπορεί να χρησιμοποιηθεί, για να αποθηκεύσει ένα αρχείο πληροφοριών μέχρι και ολόκληρο κατάλογο διαθέσιμων προϊόντων, για ένα ηλεκτρονικό κατάστημα. Σε συνδυασμό με μία γλώσσα δέσμης ενεργειών, όπως η *PHP* ή η *Perl* είναι δυνατή η δημιουργία ιστοτόπων, που θα αλληλοεπιδρούν σε πραγματικό χρόνο, με μια βάση δεδομένων *MySQL* για γρήγορη εμφάνιση κατηγοριοποιημένων και αναζητήσιμων πληροφοριών, σε έναν χρήστη ιστοτόπου.

Ένα εργαλείο που διαθέτει η *Node.Js*, για την καλύτερη διαχείριση της βάσης δεδομένων, είναι η *Sequelize*. Η *Sequelize*, είναι μια βιβλιοθήκη *ORM* (*Object-relational mapping*) που έχει τη δυνατότητα, να αλληλοεπιδρά με βάσεις δεδομένων όπως *MySQL*, *Postgres*, *MariaDB* και *Microsoft SQL*.

2.4 Τεχνολογίες Ανάπτυξης λογισμικού

Σε αυτό το κεφάλαιο, αναλύονται τα λογισμικά εργαλεία, τα οποία χρησιμοποιήθηκαν για τη δημιουργία της διαδικτυακής πλατφόρμας. Τα εργαλεία αυτά, διατίθενται δωρεάν ή παρέχονται από τα προγράμματα παροχής λογισμικού, του Πανεπιστημίου Δυτικής Μακεδονίας.

2.4.1 Visual Studio Code

Το *Visual Studio Code*, είναι ένα ελαφρύ αλλά ισχυρό πρόγραμμα επεξεργασίας πηγαίου κώδικα, που λειτουργεί στην επιφάνεια εργασίας και είναι διαθέσιμο για *Windows*, *macOS* και *Linux*. Διατίθεται, με ενσωματωμένη υποστήριξη για *JavaScript*, *TypeScript* και *Node.js* και διαθέτει πλούσιο οικοσύστημα επεκτάσεων για άλλες γλώσσες, (όπως *C++*, *C#*, *Java*, *Python*, *PHP*, *Go*) και περιβάλλον εκτέλεσης (όπως *.NET* και *Unity*) [12].

Κατά τη διάρκεια υλοποίησης της διπλωματικής εργασίας, χρησιμοποιήθηκαν πολλά εργαλεία επέκτασης στην εφαρμογή. Ένα από τα πιο σημαντικά εργαλεία είναι, το *Angular Language Service*. Η επέκταση αυτή, προσέφερε πληροφορίες για πολλές από τις ενσωματωμένες μεθόδους της *Angular*, όπως ο τύπος δεδομένων που χρειάζεται να χρησιμοποιήσει κανείς, όταν καλείται μία ενσωματωμένη μέθοδος. Εξίσου σημαντικές επεκτάσεις είναι το *TsLint* και *SonarLint*, οι οποίες χρησιμοποιούνται για τον έλεγχο της εφαρμογής κατά τη συγγραφή του κώδικα, όσον αφορά λάθη στη σύνταξη του κώδικα, την ορθογραφία, σε σφάλματα, ενώ ταυτόχρονα ελέγχει αν εφαρμόζονται πολιτικές ορθής σύνταξης. Επίσης, δίνεται η δυνατότητα στον προγραμματιστή, να ορίσει εκείνος δικούς του κανόνες ανάλυσης και ελέγχου στις επεκτάσεις για επιπλέον δυνατότητες.

2.4.2 GitKraken – Git

Το *GitKraken* είναι ένα προϊόν λογισμικού, που βοηθά τους προγραμματιστές να μεγιστοποιήσουν την παραγωγικότητά τους σε διαισθητικό *GUI*. Κυρίως χρησιμοποιείται σε μεγάλες ομάδες προγραμματιστών, για καλύτερη οργάνωση κατά την διάρκεια κατασκευής. Παρέχει δυνατότητες όπως τον πίνακα εργασιών, όπου γίνεται ο σχεδιασμός αρχιτεκτονικής του συστήματος και προσφέρει πληθώρα εργαλείων που βοηθούν σε αυτό. Ακόμη μία δυνατότητά του, είναι τα χρονοδιαγράμματα *`timelines`*. Στα *`timelines,`* μπορούμε να ορίσουμε και να οργανώσουμε τις διεργασίες που απαιτούνται να γίνουν, για την κατασκευή του συστήματος. Με αυτόν τον τρόπο, μπορούμε να ορίσουμε μία σχετική ροή κατασκευής του συστήματος, με σκοπό

να αποφευχθούν οι συγκρούσεις `*conflicts*` του κώδικα, στο αποθετήριο `*repository*` του συστήματος. Κυρίως όμως το *GitKragen*, είναι ένα γραφικό περιβάλλον διαχείρισης *repository*, που προσφέρει λύσεις διεκπεραίωσης των συγκρούσεων του κώδικα και με αυτοματοποιημένες διαδικασίες οργανώνει το *online repository*.

Το *Git*, είναι ένα σύστημα ελέγχου εκδόσεων, με έμφαση στην ταχύτητα, την ακεραιότητα των δεδομένων και την υποστήριξη για κατανεμημένες μη γραμμικές ροές εργασίας. Το *Git*, σχεδιάστηκε και αναπτύχθηκε αρχικά από τον Λίνους Τόρβαλντς, για την ανάπτυξη του πυρήνα *Linux* το 2005 και από τότε έχει γίνει το πιο διαδεδομένο σύστημα ελέγχου εκδόσεων, για ανάπτυξη λογισμικού [13].

Αντίθετα με τα περισσότερα συστήματα πελάτη-διακομιστή, κάθε κατάλογος εργασίας του *Git*, αποτελεί ένα ολοκληρωμένο αποθετήριο λογισμικού, με πλήρες ιστορικό και δυνατότητες πλήρους παρακολούθησης της έκδοσης, ανεξάρτητα από την πρόσβαση δικτύου ή ενός κεντρικού διακομιστή. Όπως ο πυρήνας *Linux*, το *Git* είναι ελεύθερο λογισμικό και διανέμεται δωρεάν σε όλους τους χρήστες.

2.4.3 Oracle MySQL Workbench

Το *MySQL Workbench*, είναι ένα γραφικό εργαλείο για την εργασία με διακομιστές και βάσεις δεδομένων *MySQL*. Το *MySQL Workbench*, υποστηρίζει πλήρως τις εκδόσεις διακομιστών *MySQL* 5.6 αλλά και νεότερες. Είναι επίσης συμβατό, με παλαιότερες εκδόσεις διακομιστή *MySQL* 5.x, εκτός από ορισμένες περιπτώσεις (όπως η εμφάνιση της λίστας διεργασιών) λόγω αλλαγμένων πινάκων του συστήματος.

Η λειτουργικότητα *MySQL Workbench* καλύπτει πέντε βασικά θέματα:

1. **Ανάπτυξη *SQL***: Επιτρέπει τη δημιουργία και διαχείριση συνδέσεων, σε διακομιστές βάσης δεδομένων. Μαζί με τη δυνατότητα ρύθμισης παραμέτρων σύνδεσης, το *MySQL Workbench* παρέχει τη δυνατότητα εκτέλεσης *SQL queries*, στη βάση δεδομένων, χρησιμοποιώντας ενσωματωμένο *SQL Editor*.
2. **Μοντελοποίηση δεδομένων (Σχεδιασμός):** Δίνεται η δυνατότητα δημιουργίας μοντέλων του σχήματος της βάσης δεδομένων, χρησιμοποιώντας γραφικό περιβάλλον, κατασκευής *reverse* and *forward engineer*, μεταξύ ενός σχήματος βάσης δεδομένων και επεξεργασίας της βάση δεδομένων χρησιμοποιώντας το *Table Editor*. Το *Table Editor*, παρέχει εύχρηστες δυνατότητες

για την επεξεργασία πινάκων, στηλών, *indexes, triggers*, διαμερισμάτων, επιλογών, *inserts* και *privileges*, ρουτίνων εκτέλεσης και προβολών.

3. **Διαχείριση Διακομιστή:** Χρησιμοποιείται για τη διαχείριση οντοτήτων στον διακομιστή *SQL*, όπως διαχείριση χρηστών, δημιουργία αντιγράφων ασφαλείας, επιθεώρηση δεδομένων ελέγχου, παρακολούθηση της κατάστασης της βάσης δεδομένων και της απόδοσης του διακομιστή.
4. **Μετεγκατάσταση Δεδομένων:** Επιτρέπει τη δυνατότητα μετεγκατάστασης από *Microsoft SQL Server, Microsoft Access, Sybase ASE, SQLite, SQL Anywhere, PostreSQL* και άλλους πίνακες, αντικείμενα και δεδομένα *RDBMS* στο *MySQL* διακομιστή. Η μετεγκατάσταση, υποστηρίζει επίσης, τη μετάβαση από παλαιότερες εκδόσεις του *MySQL* στις πιο πρόσφατες.

2.5 Σύνοψη 2ου Κεφαλαίου

Στο δεύτερο κεφάλαιο, αναλύθηκαν οι τεχνικές και γλώσσες προγραμματισμού που χρησιμοποιήθηκαν, για την επεξεργασία και δημιουργία της διαδικτυακής εφαρμογής. Επιπροσθέτως, εκτός από τα προγραμματιστικά εργαλεία, περιγράφονται διεξοδικά τα λογισμικά εργαλεία, τα οποία χρησίμευσαν στην οργάνωση και σύνθεση του πληροφοριακού συστήματος, που πραγματεύεται η διπλωματική εργασία. Το επόμενο κεφάλαιο αναφέρεται στη δομή και οργάνωση του συστήματος, καθώς και τη δημιουργία της βάσης δεδομένων.

Κεφάλαιο 3ο – Δομή & Οργάνωση Συστήματος

Σε αυτό το κεφάλαιο, περιγράφονται λεπτομερώς οι λειτουργικές και μη-λειτουργικές προδιαγραφές, του συστήματος διαχείρισης ρομποτικών συστημάτων. Επίσης, αναλύονται τα δύο είδη χρηστών που έχουν πρόσβαση στο σύστημα και οι δυνατότητες που παρέχονται από το σύστημα διαχείρισης ρομποτικών συστημάτων. Ακόμα, γίνεται αναφορά στον τρόπο δημιουργίας της βάσης δεδομένων, καθώς και στην επεξήγηση αρχείων του κώδικα, ο οποίος συντέλεσε στη δημιουργία του πληροφοριακού συστήματος.

3.1 Ανάλυση Συστήματος

Για να επιτευχθεί η ορθή ανάλυση και σχεδίαση του συστήματος, είναι απαραίτητη η κατανόηση των λειτουργιών των χρηστών. Το σύστημα διαχείρισης ρομποτικών συστημάτων, χρησιμοποιείται από δύο κατηγορίες χρηστών: από τον διαχειριστή, ο οποίος είναι υπεύθυνος για τα ρομπότ και από τους απλούς χρήστες, που αποτελούνται, από τους εκπαιδευτικούς και τους μαθητές. Επίσης, υπάρχει και ο ανώνυμος χρήστης, ο οποίος έχει πρόσβαση στην αρχική σελίδα της ιστοσελίδας, που περιέχει τις λειτουργίες εγγραφής και σύνδεσης.

Ο απλός χρήστης, εισέρχεται στο σύστημα με τα στοιχεία του λογαριασμού του και μπορεί να ελέγχει το ρομπότ, που του αντιστοιχεί, μέσω της εφαρμογής. Πιο συγκεκριμένα, ο απλός χρήστης μπορεί να καταχωρεί σενάρια για εκτέλεση και να δίνει εντολές εκτέλεσης σεναρίων στο ρομπότ.

Ο διαχειριστής εισέρχεται στο σύστημα με τα στοιχεία του λογαριασμού του και μπορεί να καταχωρεί τα ρομπότ, τα οποία θα διαχειρίζεται, μέσω του συστήματος και να τα αναθέτει σε χρήστες. Επιπλέον, έχει τη δυνατότητα να εισάγει, να επεξεργάζεται και να διαγράφει από το σύστημα, τους χρήστες. Μία ακόμη ενέργεια στην οποία μπορεί να προβεί ο διαχειριστής, είναι να καταχωρεί νέα σενάρια, να αναθέτει σε χρήστες τα σενάρια που βρίσκονται στη βάση δεδομένων και να τα επεξεργάζεται ή να τα διαγράφει. Τέλος, μπορεί να διαχειριστεί την πλατφόρμα όπως και ο απλός χρήστης.

3.2 Ανάλυση Απαιτήσεων – Λειτουργικών προδιαγραφών

Η πλατφόρμα που υλοποιήθηκε, δίνει τη δυνατότητα στο εκπαιδευτικό προσωπικό να πραγματοποιεί μαθήματα με μεγαλύτερη ευκολία και απλότητα. Για να δοθεί εντολή εκτέλεσης σεναρίου, ο χρήστης εισέρχεται στο σύστημα και επιλέγει ένα σενάριο προς εκτέλεση, που είναι ήδη καταχωρημένο ή καταχωρεί ο ίδιος το σενάριο που θα είναι προς εκτέλεση.

Για να δώσει εντολή εκτέλεσης σεναρίου, ο χρήστης, μπορεί να χρησιμοποιήσει, είτε τα προ-εγκατεστημένα σενάρια που παρέχονται από τον διαχειριστή, είτε να καταχωρήσει δικά του σενάρια, συμπληρώνοντας την αντίστοιχη φόρμα και επισυνάπτοντας το αρχείο *zip* που περιέχει τον κώδικα του σεναρίου. Αφού, λοιπόν, έχει γίνει σύνδεση με το ρομπότ και υπάρχουν σενάρια καταχωρημένα στο σύστημα, ο χρήστης επιλέγοντας το εικονίδιο εκτέλεσης σεναρίου καταχωρεί το επιλεγμένο σενάριο προς εκτέλεση.

Σχετικά με τις δυνατότητες του διαχειριστή, το σύστημα προσφέρει πολλές επιλογές για έλεγχο όλων των οντοτήτων, με τις οποίες αλληλεπιδρά η πλατφόρμα. Όσον αφορά τα ρομπότ, ο διαχειριστής είναι υπεύθυνος για την καταχώρηση των ρομποτικών συστημάτων στη βάση δεδομένων και αφού παραχωρήσει το σειριακό αριθμό ενός ρομπότ στο χρήστη, τότε του δίνεται πρόσβαση σε αυτό και μόνο σε αυτό. Ακόμη, του δίνονται επιλογές επεξεργασίας των στοιχείων του ρομπότ και διαγραφή αυτών από τη βάση δεδομένων. Επιπλέον, έχει τη δυνατότητα καταχώρησης νέων χρηστών στη βάση δεδομένων, με την επιλογή να τους παρέχει δικαιώματα διαχειριστή και όπως και με τα ρομπότ, μπορεί να επεξεργάζεται και να διαγράφει, τους χρήστες, που βρίσκονται ήδη καταχωρημένοι στη βάση δεδομένων. Τέλος, ο διαχειριστής μπορεί να αναθέτει σενάρια που είναι καταχωρημένα στη βάση δεδομένων, σε χρήστες που χρησιμοποιούν την πλατφόρμα.

Στην πλατφόρμα, τα σενάρια που αντιστοιχούν στον κάθε χρήστη, εμφανίζονται σε μορφή πινάκων. Επίσης, για τον διαχειριστή παρέχονται επιπλέον πίνακες για την επίβλεψη των ρομπότ και χρηστών που είναι καταχωρημένοι στη βάση δεδομένων.

3.3 Ανάλυση Απαιτήσεων – Μη-Λειτουργικών προδιαγραφών

Η πλατφόρμα έχει κατασκευαστεί ως *one-page application*, χρησιμοποιώντας το *Angular* 9 *framework* με ενσωμάτωση της *RxJs* βιβλιοθήκης, προσφέροντας δυναμικό έλεγχο των δεδομένων και γρήγορη περιήγηση. Καθώς χρησιμοποιείται *Angular framework*, η σελίδα και κάθε *http request* που γίνεται στον *server*, διασφαλίζονται από το ένα και μοναδικό *web token* που λαμβάνει ο χρήστης, μετά από κάθε επιτυχή είσοδό του στο σύστημα και του δίνεται πρόσβαση να χρησιμοποιήσει την πλατφόρμα. Το *token*, παραμένει ενεργό στο *browser* για ένα χρονικό διάστημα 30 λεπτών, ώστε να επιτυγχάνεται η ασφάλεια του λογαριασμού. Μετά από το αναφερόμενο χρονικό περιθώριο, απαιτείται νέα είσοδος του χρήστη.

Για ευκολότερη διαχείριση της πλατφόρμας, προσφέρεται δυνατότητα επανέκδοσης κωδικού του χρήστη, με την προϋπόθεση εξακρίβωσης του ονόματος χρήστη και του *email*. Αυτό επιτυγχάνεται με αποστολή εξατομικευμένου e*mail*, που περιέχει το νέο προσωρινό κωδικό του λογαριασμού του.

Ακόμη, για να επιτευχθεί βέλτιστη εμπειρία χρήσης και ευχάριστο περιβάλλον περιήγησης, έχει γίνει εφαρμογή της *Angular Material* και *Grid-view layout design*. Μία από τις δυνατότητες της *Angular Material*, είναι το *smart-search*, που προσφέρει δυνατότητα αναζήτησης στοιχείων, ενός πίνακα ανεξαρτήτως πεδίου. Όσον αφορά το *Grid-view design*, μας δίνει τη δυνατότητα ταξινόμησης των πεδίων σε μορφή που υποστηρίζεται η λειτουργία της πλατφόρμα και από *mobile* συσκευές.

Επειδή έχει χρησιμοποιηθεί τεχνική του *Reactive Programming*, για την κατασκευή των πεδίων καταχώρησης στοιχείων, σε κάθε φόρμα της πλατφόρμας, περιέχονται έλεγχοι που εκτελούνται δυναμικά από τη στιγμή που ο χρήστης αρχίζει να πληκτρολογεί. Στην περίπτωση που δεν πληρούνται οι αντίστοιχες προδιαγραφές του πεδίου, τότε εμφανίζονται εξιδεικευμένα μηνύματα ενημέρωσης. Σε κάθε φόρμα, η επιλογή καταχώρησης παραμένει απενεργοποιημένη, μέχρι ο χρήστης να συμπληρώσει σωστά τα πεδία και να πραγματοποιηθούν με επιτυχία, οι αντίστοιχοι έλεγχοι ορθότητας της φόρμας.

3.4 Περιπτώσεις Χρήσης (*Use cases*)

- Εισαγωγή στο σύστημα

Περιγραφή: Ο τρόπος που συνδέεται ο χρήστης στην πλατφόρμα

Χειριστές: Διαχειριστής, Χρήστης

Προϋποθέσεις: Ο χρήστης να είναι καταχωρημένος στο σύστημα

Βασική ροή:

1. Ο χρήστης δεν έχει *token* στο *browser*
2. Ο χρήστης εισάγει το όνομα χρήστη και το κωδικό του λογαριασμού του
3. Επιβεβαιώνεται η επιτυχής εισαγωγή των στοιχείων και εισέρχεται στο σύστημα

Εναλλακτικό σενάριο: Όταν ο χρήστης εισάγει εσφαλμένα στοιχεία, εμφανίζεται μήνυμα σφάλματος και ζητείται από τον χρήστη να εισάγει τα σωστά στοιχεία.

- Εγγραφή χρήστη

Περιγραφή: Ο τρόπος με τον οποίο καταχωρούνται απλοί χρήστες στο σύστημα

Χειριστές: Ανώνυμος Χρήστης

Προϋποθέσεις: Ο χρήστης θα πρέπει να έχει λάβει τον σειριακό αριθμό που του αντιστοιχεί.

Βασική ροή:

1. Ο χρήστης έχει λάβει σειριακό αριθμό που του αντιστοιχεί.
2. Ο χρήστης συμπληρώνει τα στοιχεία του στην αρχική σελίδα και καταχωρεί τον σειριακό αριθμό στο ανάλογο πεδίο (όνομα χρήστη, όνομα, επίθετο, *email*, κωδικό πρόσβασης, σειριακό αριθμό ρομπότ)
3. Επιβεβαιώνεται η επιτυχής εισαγωγή στοιχείων από *client-side* και *server-side validators*. (Εξακρίβωση τύπου και πλήθους δεδομένων)
4. Να δημιουργηθεί *email* για εξακρίβωση χρήστη
5. Εμφανίζεται μήνυμα επιτυχούς καταχώρησης.
6. Γίνεται επαναφορά της φόρμας εγγραφής χρήστη.

Εναλλακτικό σενάριο: Στο βήμα 3, αν ο χρήστης εισάγει λάθος στοιχεία, θα του ζητηθεί, με ανάλογο μήνυμα, να διορθώσει την καταχώρηση του.

- Επανέκδοση κωδικού

Περιγραφή: Ο τρόπος με τον οποίο ο χρήστης θα αναθέτει νέο κωδικό στον λογαριασμό του σε περίπτωση που τον έχει ξεχάσει

Χειριστές: Χρήστης, Διαχειριστής

Προϋποθέσεις: Καμία

Βασική ροή:

1. Ο χρήστης έχει ξεχάσει τον κωδικό του
2. Ο χρήστης περιηγείται στο σημείο επαναφοράς κωδικού
3. Εισάγει τα στοιχεία που του ζητούνται (όνομα χρήστη, *email*)
4. Ελέγχει το *email* του για τον νέο κωδικό πρόσβασης.
5. Κάνει είσοδο στο σύστημα

Εναλλακτικό σενάριο: Στο βήμα 3, αν ο χρήστης δεν εισάγει σωστά στοιχεία, δεν πρόκειται να λάβει *email*.

- Υπενθύμιση Ονόματος Χρήστη

Περιγραφή: Ο τρόπος με τον οποίο ο χρήστης ζητά *email* υπενθύμισης για το όνομα χρήστη του λογαριασμού του

Χειριστές: Χρήστης, Διαχειριστής

Προϋποθέσεις: Καμία

Βασική ροή:

1. Ο χρήστης έχει ξεχάσει το όνομα χρήστη του λογαριασμού του
2. Ο χρήστης περιηγείται στο ‘Υπενθύμισή Ονόματος Χρήστη’
3. Εισάγει τα στοιχεία που του ζητούνται (*email*)
4. Ελέγχει το *email* του για το όνομα χρήστη του λογαριασμού του.
5. Κάνει είσοδο στο σύστημα

Εναλλακτικό σενάριο: Στο βήμα 3, αν ο χρήστης δεν εισάγει σωστά στοιχεία, δεν πρόκειται να λάβει *email*.

- Αποσύνδεση

Περιγραφή: Ο τρόπος με τον οποίο ο χρήστης αποσυνδέεται από το σύστημα

Χειριστές: Διαχειριστής, Χρήστης

Προϋποθέσεις: Καμία

Βασική ροή: Επιλογή της Αποσύνδεσης από το κεντρικό μενού.

- Εισαγωγή ρομπότ

Περιγραφή: Ο τρόπος με τον οποίο ο διαχειριστής καταχωρεί νέα ρομπότ για διαχείριση στο σύστημα.

Χειριστές: Διαχειριστής

Προϋποθέσεις: Σειριακός αριθμός του ρομπότ και ονομασία ρομπότ

Βασική ροή:

1. Επιλογή της κατηγορίας ρομπότ από το κεντρικό μενού.
2. Είσοδος στη φόρμα Προσθήκη Ρομπότ.
3. Συμπλήρωση των στοιχείων που ζητούνται (όνομα, σειριακός αριθμός).
4. Επιβεβαίωση από το σύστημα για το αν υπάρχει άλλο ρομπότ με τα ίδια στοιχεία.
5. Αποθήκευση της οντότητας.
6. Επιστροφή στο πίνακα επίβλεψης ρομπότ

Εναλλακτικό σενάριο: Στο βήμα 3, αν ο διαχειριστής δεν βάλει σωστά στοιχεία, το σύστημα του απαγορεύει να αποθηκεύσει την οντότητα.

- Επεξεργασία ρομπότ

Περιγραφή: Ο τρόπος με τον οποίο ο διαχειριστής επεξεργάζεται τα ρομπότ που είναι καταχωρημένα στο σύστημα.

Χειριστές: Διαχειριστής

Προϋποθέσεις: Καμία

Βασική ροή:

1. Επιλογή της κατηγορίας ρομπότ από το κεντρικό μενού.
2. Επιλογή της ιδιότητας Επεξεργασία στην οντότητα ρομπότ που θέλει να τροποποιήσει.
3. Επεξεργασία των στοιχείων που βρίσκονται στη φόρμα.
4. Επιβεβαίωση από το σύστημα
5. Αποθήκευση της οντότητας.
6. Επιστροφή στον πίνακα επίβλεψης ρομπότ

Εναλλακτικό σενάριο: Στο βήμα 3, αν ο διαχειριστής δεν βάλει σωστά στοιχεία, το σύστημα του απαγορεύει να αποθηκεύσει την οντότητα.

- Διαγραφή ρομπότ

Περιγραφή: Ο τρόπος με τον οποίο ο διαχειριστής διαγράφει τα ρομπότ που είναι καταχωρημένα στο σύστημα.

Χειριστές: Διαχειριστής

Προϋποθέσεις: Καμία

Βασική ροή:

1. Επιλογή της κατηγορίας ρομπότ από το κεντρικό μενού.
2. Επιλογή της ιδιότητας Διαγραφή στην οντότητα ρομπότ που θέλει να διαγράψει.
3. Επιβεβαίωση της επιλογής για διαγραφή οντότητας.
4. Επιστροφή στον πίνακα επίβλεψης ρομπότ

Εναλλακτικό σενάριο: Στο βήμα 3, αν ο διαχειριστής δεν επιβεβαιώσει την επιλογή, το σύστημα τον επιστρέφει στον πίνακα επίβλεψης ρομπότ.

- Προσθήκη νέου χρήστη

Περιγραφή: Ο τρόπος με τον οποίο ο διαχειριστής προσθέτει νέους χρήστες ή διαχειριστές στο σύστημα.

Χειριστές: Διαχειριστής

Προϋποθέσεις: Καμία

Βασική ροή:

1. Επιλογή της κατηγορίας χρήστες από το κεντρικό μενού.
2. Επιλογή στη Προσθήκη Νέου Χρήστη.
3. Συμπλήρωση των πεδίων που απαιτούνται από το σύστημα για τη δημιουργία νέου χρήστη ή διαχειριστή (όνομα χρήστη, όνομα, επίθετο, *email*, κωδικό πρόσβασης, ρομπότ, διαχειριστής).
4. Επιβεβαίωση οντότητας από το σύστημα και ενεργοποίηση αποθήκευσης.
5. Αποθήκευση της οντότητας.
6. Επιστροφή στον πίνακα επίβλεψης χρηστών

Εναλλακτικό σενάριο: Στο βήμα 3, αν ο διαχειριστής δεν συμπληρώσει ορθά τα στοιχεία ή υπάρχει κάποιος άλλος χρήστης με τα ίδια στοιχεία τότε δεν επιτρέπεται η αποθήκευση της οντότητας.

- Επεξεργασία χρήστη

Περιγραφή: Ο τρόπος με τον οποίο ο διαχειριστής επεξεργάζεται τους χρήστες που έχουν καταχωρηθεί στο σύστημα.

Χειριστές: Διαχειριστής

Προϋποθέσεις: Καμία

Βασική ροή:

1. Επιλογή της κατηγορίας χρήστες από το κεντρικό μενού
2. Επιλογή της ιδιότητας Επεξεργασία στην οντότητα χρήστη που θέλει να επεξεργαστεί.
3. Επεξεργασία των πεδίων που απαιτούνται από το σύστημα.
4. Επιβεβαίωση οντότητας από το σύστημα και ενεργοποίηση αποθήκευσης
5. Αποθήκευση της οντότητας.
6. Επιστροφή στον πίνακα επίβλεψης χρηστών

Εναλλακτικό σενάριο: Στο βήμα 3, αν ο διαχειριστής δε συμπληρώσει ορθά τα στοιχεία ή υπάρχει κάποιος άλλος χρήστης με τα ίδια στοιχεία τότε δεν επιτρέπεται η αποθήκευση της οντότητας.

- Διαγραφή χρήστη

Περιγραφή: Ο τρόπος με τον οποίο ο διαχειριστής διαγράφει τους χρήστες που είναι καταχωρημένοι στο σύστημα.

Χειριστές: Διαχειριστής

Προϋποθέσεις: Καμία

Βασική ροή:

1. Επιλογή της κατηγορίας χρήστες από το κεντρικό μενού.
2. Επιλογή της ιδιότητας Διαγραφή στην οντότητα χρήστη που θέλει να διαγράψει.
3. Επιβεβαίωση της επιλογής για διαγραφή οντότητας.
4. Επιστροφή στο πίνακα επίβλεψης χρηστών

Εναλλακτικό σενάριο: Στο βήμα 3, αν ο διαχειριστής δεν επιβεβαιώσει την επιλογή, το σύστημα τον επιστρέφει στο πίνακα επίβλεψης χρηστών.

- Προσθήκη νέου σεναρίου

Περιγραφή: Ο τρόπος με τον οποίο ο χρήστης προσθέτει νέο σενάριο.

Χειριστές: Διαχειριστής, Χρήστης

Προϋποθέσεις: Καμία

Βασική ροή:

1. Επιλογή της κατηγορίας σενάρια από το κεντρικό μενού.
2. Επιλογή στη Προσθήκη Νέου Σεναρίου.
3. Συμπλήρωση των πεδίων που απαιτούνται από το σύστημα για τη δημιουργία νέου σεναρίου (όνομα, σύναψη *.zip* αρχείου).
4. Επιβεβαίωση οντότητας από το σύστημα και ενεργοποίηση αποθήκευσης.
5. Αποθήκευση της οντότητας.
6. Επιστροφή στον πίνακα επίβλεψης σεναρίων

Εναλλακτικό σενάριο: Στο βήμα 3, αν ο χρήστης δε συμπληρώσει ορθά τα στοιχεία ή υπάρχει κάποιο άλλο σενάριο με τα ίδια στοιχεία τότε δεν επιτρέπεται η αποθήκευση της οντότητας.

- Επεξεργασία σεναρίου

Περιγραφή: Ο τρόπος με τον οποίο ο χρήστης επεξεργάζεται καταχωρημένο σενάριο.

Χειριστές: Διαχειριστής, Χρήστης

Προϋποθέσεις: Καμία

Βασική ροή:

1. Επιλογή της κατηγορίας σενάρια από το κεντρικό μενού.
2. Επιλογή 'Επεξεργασία' στο σενάριο προς επεξεργασία.
3. Επεξεργασία των πεδίων του σεναρίου (όνομα, νέα σύναψη *.zip* αρχείου)
4. Επιβεβαίωση οντότητας από το σύστημα και ενεργοποίηση αποθήκευσης.
5. Αποθήκευση της οντότητας.
6. Επιστροφή στον πίνακα επίβλεψης σεναρίων

Εναλλακτικό σενάριο: Στο βήμα 3, αν ο χρήστης δε συμπληρώσει ορθά τα στοιχεία ή υπάρχει κάποιο άλλο σενάριο με τα ίδια στοιχεία τότε δεν επιτρέπεται η αποθήκευση της οντότητας.

- Διαγραφή σεναρίου

Περιγραφή: Ο τρόπος με τον οποίο ο χρήστης διαγράφει τα σενάρια που έχουν καταχωρηθεί στο σύστημα.

Χειριστές: Διαχειριστής, Χρήστης

Προϋποθέσεις: Καμία

Βασική ροή:

1. Επιλογή της κατηγορίας σενάρια από το κεντρικό μενού.
2. Επιλογή της ιδιότητας Διαγραφή στην οντότητα σεναρίου που θέλει να διαγράψει.
3. Επιβεβαίωση της επιλογής για διαγραφή οντότητας.
4. Επιστροφή στον πίνακα επίβλεψης σεναρίων

Εναλλακτικό σενάριο: Στο βήμα 3, αν ο διαχειριστής δε συμπληρώσει ορθά τα στοιχεία ή υπάρχει κάποιος άλλος χρήστης με τα ίδια στοιχεία τότε δεν επιτρέπεται η αποθήκευση της οντότητας.

- Εκτέλεση σεναρίου

Περιγραφή: Ο τρόπος με τον οποίο ο χρήστης δίνετε εντολή εκτέλεσης σεναρίου

Χειριστές: Διαχειριστής, Χρήστης

Προϋποθέσεις: Καμία

Βασική ροή:

1. Επιλογή της κατηγορίας σενάρια από το κεντρικό μενού
2. Επιλογή του εικονίδιού Σύνδεση
    a. Στη περίπτωση του διαχειριστή που έχει στη διάθεση του όλα τα ρομπότ θα ζητείται να επιλέξει σε πιο ρομπότ θέλει να συνδεθεί.
3. Επιβεβαίωση σύνδεση με το ρομπότ (αλλαγή χρώματος του εικονιδίου και ενεργοποίηση εικονιδίου εκτέλεσης)
4. Επιλογή του εικονιδίου Εκτέλεση ▸.
5. Για να σταματήσει την εκτέλεση του σεναρίου επιλέγεται το εικονίδιο Παύσης ■

Εναλλακτικό σενάριο: Στο βήμα 3, αν ο χρήστης δεν έχει συνδεθεί με το ρομπότ δεν θα αλλάξει το χρώμα της Σύνδεσης και το εικονίδιο εκτέλεσης θα παραμείνει ανενεργό

- Ανάθεση Σεναρίου

Περιγραφή: Ο τρόπος με τον οποίο ο διαχειριστής αναθέτει σενάριο σε χρήστες

Χειριστές: Διαχειριστής

Προϋποθέσεις: Καμία

Βασική ροή:

1. Επιλογή της κατηγορίας ανάθεση από το κεντρικό μενού.
2. Επιλογή του κουμπιού Ανάθεση
3. Συμπλήρωση της φόρμας ανάθεσης σεναρίων.
4. Αποθήκευση επιλογών.

5. Αυτόματη επιστροφή στο πίνακα επίβλεψης αναθέσεων

Εναλλακτικό σενάριο: Στην περίπτωση που ο χρήστης αλλάξει σελίδα θα ακυρωθεί η φόρμα.

- Διαγραφή Ανάθεσης

Περιγραφή: Ο τρόπος με τον οποίο ο διαχειριστής αφαιρεί πρόσβαση ενός σεναρίου από συγκεκριμένο χρήστη

Χειριστές: Διαχειριστής

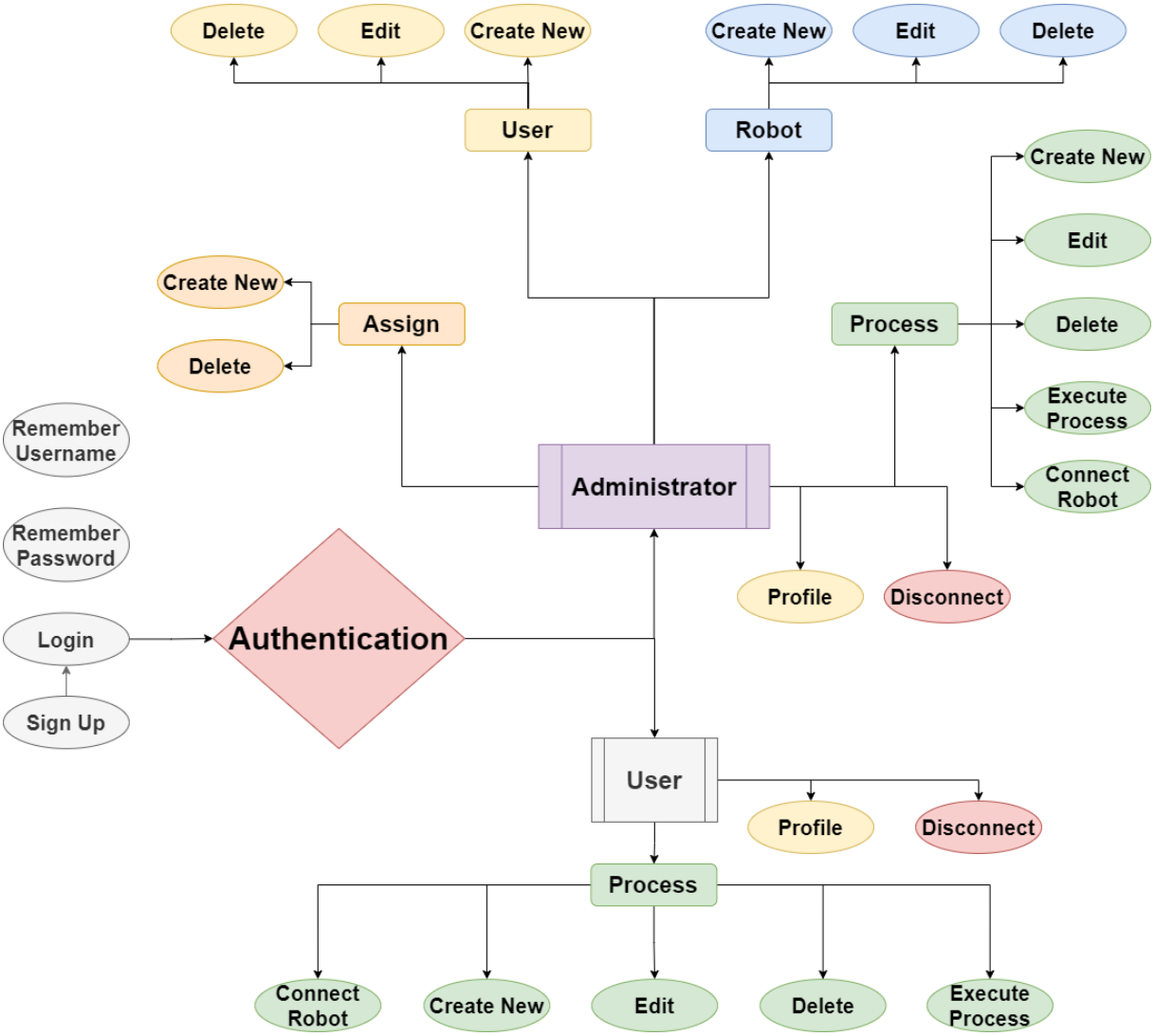
Προϋποθέσεις: Καμία

Βασική ροή:

1. Επιλογή της κατηγορίας ανάθεση από το κεντρικό μενού.
2. Επιλογή του κουμπιού Διαγραφή από την επιλεγμένη αντιστοιχία

Εναλλακτικό σενάριο: Κανένα

Στην εικόνα (15) παρουσιάζεται το διάγραμμα ER που γίνεται εικονική περιγραφή των λειτουργιών που έχει πρόσβαση κάθε χρήστης.

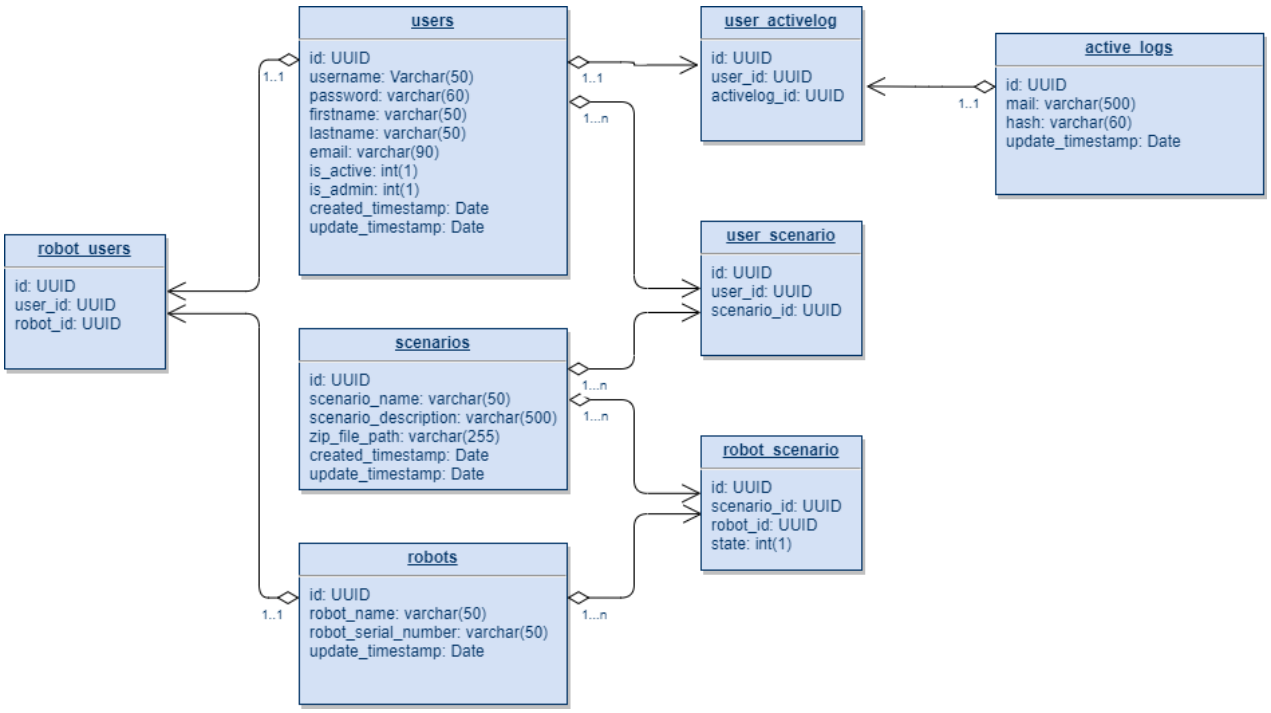


Εικόνα 15: Διάγραμμα ER

3.5 Σχεσιακό διάγραμμα βάσης δεδομένων

Για τη δόμηση του σχεσιακού μοντέλου, έχουν οριστεί οι σχέσεις (*relations*) και οι πίνακες (*tables*) της βάσης δεδομένων. Το σύνολο των πληροφοριών που αποθηκεύονται στη βάση δεδομένων αποτελείται από μία συλλογή πινάκων. Οι γραμμές των πινάκων καλούνται πλειάδες (*tuples*) ή εγγραφές (*records*) των πινάκων (*σχέσεις*) και αντιστοιχούν σε οντότητες. Η εικόνα αναπαριστά τη δομή της βάσης δεδομένων με τους πίνακες, τα πεδία, τους τύπους δεδομένων και τις συσχετίσεις που υπάρχουν μεταξύ τους.

Στην εικόνα (16) παρουσιάζεται το σχεσιακό διάγραμμα της βάσης δεδομένων.



Εικόνα 16: Σχεσιακό διάγραμμα βάσης δεδομένων

3.6 Σχεδιασμός και δημιουργία βάσης δεδομένων

Για την αποδοτική λειτουργία της πλατφόρμας που υλοποιήθηκε, σχεδιάστηκε η βάση δεδομένων που πρόκειται να υποστηρίζει τις λειτουργικές προδιαγραφές των απαιτήσεων του συστήματος. Η βάση δεδομένων έχει σχεδιαστεί με κανονικοποίηση *4NF*, με στόχο να επιτευχθεί ευκολία ως προς τα ερωτήματα διαχείρισης. Μέσα από τη χρήση αυτής της μεθόδου τα ανεξάρτητα πεδία διακρίνονται σε πολλαπλούς πίνακες, ενώ από τη χρήση των σχεσιακών πινάκων μπορεί να ορίσει κανείς τις σχέσεις μεταξύ τους.

Η βάση δεδομένων αποτελείται από εννέα (9) πίνακες, στους οποίους αποθηκεύονται οι χρήστες του συστήματος, τα ρομπότ και τα σενάρια προς εκτέλεση, καθώς, επίσης, και πίνακες συσχέτισης που ορίζουν τις σχέσεις των πινάκων μεταξύ τους. Στις παρακάτω ενότητες περιγράφεται ειδικότερα το περιεχόμενο των πινάκων καθώς και οι σχέσεις μεταξύ τους. Στη συνέχεια, αναλύεται κάθε πίνακας ξεχωριστά, περιγράφοντας τις ιδιότητες και τα πεδία από τα οποία αποτελείται.

Χρησιμοποιώντας το *ORM Sequelize* για την κατασκευή των πινάκων, για την καλύτερη κατανόηση του κώδικα επιλέχθηκαν οι εξής ονομασίες των πινάκων:

1. *users*
2. *scenarios*
3. *robots*
4. *active_logs*
5. *user_activelog*
6. *user_scenario*
7. *robot_scenario*
8. *robot_user*

3.6.1 Πίνακας *users*

Ο πίνακας *users*, περιέχει τα στοιχεία των χρηστών που χρησιμοποιούν την πλατφόρμα. Οι χρήστες αποτελούνται από τις κατηγορίες των απλών χρηστών και των διαχειριστών. Τα δεδομένα που αποθηκεύονται στον πίνακα αποτελούν τα συνθηματικά που χρησιμοποιεί κανείς για να εισέλθει στο σύστημα και τα προσωπικά τους δεδομένα.

Στην εικόνα (17) παρουσιάζεται η δομή του πίνακα *users*.

Column	Type	Default Value	Nullable
username	varchar(50)		NO
update_timestamp	datetime		NO
password	varchar(60)		NO
lastName	varchar(50)		NO
is_admin	tinyint(1)	0	NO
is_active	tinyint(1)	0	NO
id	char(36)		NO
firstName	varchar(50)		NO
email	varchar(90)		NO
created_timestamp	datetime		NO

Εικόνα 17: Oracle Workbench πίνακας users

Ανάλυση πεδίων

- **id**: Είναι το *UUID*, που χαρακτηρίζει ξεχωριστά κάθε οντότητα μέσα στον πίνακα. Χρησιμοποιήθηκε *UUID* για λόγους ασφάλειας των request. Στο *RestAPI*, έχει δημιουργηθεί μέθοδος που ορίζει ένα unique αλφαριθμητικό με εύρος χαρακτήρων τριάντα έξη (36).
- **username**: Το όνομα του χρήστη, που χρησιμοποιείται κατά την είσοδό του στο σύστημα. *Unique* αλφαριθμητικό με εύρος χαρακτήρων πενήντα (50).
- **password**: Κρυπτογραφημένο μήνυμα κωδικού πρόσβασης (*hash password*) το οποίο εισάγει ο χρήστης κατά την είσοδό του στο σύστημα. Αλφαριθμητικό με εύρος χαρακτήρων εξήντα (60).
- **firstName**: Το πραγματικό όνομα του χρήστη. Αλφαριθμητικό με εύρος χαρακτήρων πενήντα (50).
- **lastName**: Το πραγματικό επίθετο του χρήστη. Αλφαριθμητικό με εύρος χαρακτήρων πενήντα (50).
- **email**: Το email του χρήστη. Αλφαριθμητικό με εύρος χαρακτήρων ενενήντα (90).

- **is_admin**: Καθορίζει την ιδιότητα του χρήστη ως διαχειριστή. Το πεδίο είναι τύπου tinyint και του καταχωρούνται οι τιμές 0 ή 1, με προεπιλεγμένη την τιμή 0, που καταχωρείται από το *ORM*. Το εύρος τιμών είναι ένα (1).
- **is_active**: Παράμετρος που χρησιμοποιείται για τον έλεγχο ενεργοποίησης του λογαριασμού του χρήστη. Σε περίπτωση που είναι μηδέν (0), τότε ο λογαριασμός του δεν έχει ενεργοποιηθεί και δεν έχει πρόσβαση στο σύστημα. Σε περίπτωση που είναι ένα (1), τότε μπορεί να χρησιμοποιήσει την εφαρμογή.
- **updated_timestamp**: Είναι η στιγμή (*datetime*) κατά την οποία έγινε η τελευταία τροποποίηση της οντότητας. Τύπος δεδομένων *DateTime*

3.6.2 Πίνακας *robots*

Ο πίνακας *robots*, περιέχει τα στοιχεία του κάθε ρομπότ που έχει καταχωρηθεί στη πλατφόρμα. Τα δεδομένα της κάθε οντότητας του πίνακα είναι *unique*, ώστε να είναι ξεκάθαρα ως προς τον διαχειριστή, οι ιδιότητες κάθε ρομπότ. Για να δοθεί πρόσβαση στο ρομπότ να συνδεθεί με το σύστημα, είναι απαραίτητο να έχει γίνει καταχώρηση των στοιχείων του στον συγκεκριμένο πίνακα.

Στην εικόνα (18) παρουσιάζεται η δομή του πίνακα *robots*

Column	Type	Nullable
id	char(36)	NO
robot_name	varchar(50)	NO
robotSerialNumber	varchar(50)	NO
update_timestamp	datetime	NO

Εικόνα 18: Oracle Workbench πίνακας robots

Ανάλυση πεδίων

- **id**: Είναι το *UUID* που χαρακτηρίζει ξεχωριστά κάθε οντότητα μέσα στον πίνακα. Χρησιμοποιήθηκε *UUID* για λόγους ασφάλειας των request. Στο *RestAPI* έχει δημιουργηθεί μέθοδος που ορίζει ένα unique αλφαριθμητικό με εύρος χαρακτήρων τριάντα έξη (36).
- **robot_name**: Είναι το χαρακτηριστικό όνομα που θα δώσει ο διαχειριστής σε κάθε ρομπότ. *Unique* αλφαριθμητικό με εύρος χαρακτήρων πενήντα (50).

- **robot_serial_number**: Είναι ο σειριακός αριθμός που χαρακτηρίζει το κάθε ρομπότ. Unique αλφαριθμητικό με εύρος χαρακτήρων πενήντα (20).
- **update_timestamp**: Είναι η στιγμή (*datetime*) κατά την οποία έγινε η τελευταία τροποποίηση της οντότητας. Τύπος δεδομένων *DateTime*

3.6.3 Πίνακας *scenarios*

Ο πίνακας *scenarios*, περιέχει στοιχεία για κάθε σενάριο που καταχωρείται στην πλατφόρμα. Τα δεδομένα της κάθε οντότητας στον πίνακα είναι *unique*, ώστε να είναι ξεκάθαρες, ως προς τον διαχειριστή, οι ιδιότητες κάθε σεναρίου.

Στην εικόνα (19) παρουσιάζεται η δομή του πίνακα *scenarios*

Column	Type	Nullable
created_timestamp	datetime	NO
id	char(36)	NO
scenario_description	varchar(500)	YES
scenario_name	varchar(50)	NO
update_timestamp	datetime	NO
zip_file_path	varchar(255)	NO

Εικόνα 19: Oracle Workbench πίνακας scenarios

Ανάλυση πεδίων

- **id**: Είναι το *UUID* που χαρακτηρίζει ξεχωριστά κάθε οντότητα μέσα στον πίνακα. Χρησιμοποιήθηκε *UUID* για λόγους ασφάλειας των request. Στο *RestAPI* έχει δημιουργηθεί μέθοδος που ορίζει ένα unique αλφαριθμητικό με εύρος χαρακτήρων τριάντα έξη (36).
- **scenario_name**: Είναι το χαρακτηριστικό όνομα που θα δώσει ο χρήστης σε κάθε σενάριο. *Unique* αλφαριθμητικό με εύρος χαρακτήρων πενήντα (50).
- **scenario_description**: Είναι κείμενο 250 χαρακτήρων που θα περιγράφει το σενάριο. Αλφαριθμητικό με εύρος χαρακτήρων διακόσια πενήντα πέντε (255).
- **zip_file_path**: Είναι το σημείο στον δίσκο που έχει αποθηκευτεί το αρχείο *zip*. Αλφαριθμητικό με εύρος χαρακτήρων διακόσια πενήντα πέντε (255).
- **created_timestamp**: Είναι η στιγμή (*datetime*) κατά την οποία έγινε η καταχώρηση της οντότητας στον πίνακα. Τύπος δεδομένων *DateTime*.
- **updated_timestamp**: Είναι η στιγμή (*datetime*) κατά την οποία έγινε η τελευταία τροποποίηση της οντότητας. Τύπος δεδομένων *DateTime*.

3.6.4 Πίνακας *active_logs*

Ο πίνακας *active_logs*, περιέχει δεδομένα που χρησιμοποιούνται για την εξακρίβωση *mail* ενός νέου χρήστη. Αφού καταχωρήσει τα στοιχεία του ο χρήστης στη φόρμα εγγραφής νέου χρήστη και γίνει επιτυχής υποβολή των στοιχείων του, το σύστημα στέλνει *mail* επιβεβαίωσης στο *mail* του χρήστη που καταχώρησε κατά την εγγραφή του. Το *mail*, που στέλνεται, περιέχει μια αλφαριθμητική τιμή (*hash*) που δημιουργείται από το σύστημα και αποθηκεύεται στον πίνακα.

Στην εικόνα (20) παρουσιάζεται η δομή του πίνακα *active_log*:

Column	Type	Nullable
hash	varchar(60)	NO
id	char(36)	NO
mail	varchar(500)	NO
update_timestamp	datetime	NO

Εικόνα 20: Oracle Workbench πίνακας active_logs

Ανάλυση πεδίων:

- **id:** Είναι το *UUID* που χαρακτηρίζει ξεχωριστά κάθε οντότητα μέσα στον πίνακα. Χρησιμοποιήθηκε *UUID* για λόγους ασφάλειας των request. Στο *RestAPI* έχει δημιουργηθεί μέθοδος που ορίζει ένα unique αλφαριθμητικό με εύρος χαρακτήρων τριάντα έξη (36).
- **hash**: Είναι αλφαριθμητική τιμή που δημιουργείται στο *RestAPI* κατά την εγγραφή του χρήστη στο σύστημα. Η τιμή είναι ξεχωριστή για κάθε χρήστη και χρησιμοποιείται για την εξακρίβωση του *mail* του. Έχει εύρος χαρακτήρων εξήντα (60).
- **mail**: Σε αυτό περιέχεται το κείμενο του mail που στέλνεται σε κάθε χρήστη που κάνει εγγραφή στο σύστημα. Κυρίως χρησιμοποιείται για λόγους παρακολούθησης ορθής διεξαγωγής της διαδικασίας. Έχει εύρος χαρακτήρων πεντακόσια (500).
- **update_timestamp**: Είναι η στιγμή (*datetime*) κατά την οποία έγινε η τελευταία τροποποίηση της οντότητας. Τύπος δεδομένων *DateTime*.

3.6.5 Πίνακας *robot_users*

Ο πίνακας *robot_users*, περιέχει τις σχέσεις μεταξύ των οντοτήτων του χρήστη και του ρομπότ. Βασισμένο σε αυτό το πίνακα το *RestAPI* καθορίζει σε πιο ρομπότ έχει πρόσβαση ο χρήστης. Λόγο ότι κάθε απλός χρήστης έχει πρόσβαση μόνο με ένα ρομπότ, η σχέση είναι πολλά-προς-ένα. Οι διαχειριστές λαμβάνουν πρόσβαση σε όλα τα ρομπότ που έχουν καταχωρηθεί στο σύστημα. Επίσης ένα ρομπότ μπορεί να διαχειρίζεται από πολλούς χρήστες.

Στην εικόνα (21) παρουσιάζεται η δομή του πίνακα *robot_users*.

Column	Type	Nullable
id	char(36)	NO
robotId	char(36)	NO
userId	char(36)	YES

Εικόνα 21: Oracle Workbench πίνακας robot_users

Ανάλυση πεδίων

- **id**: Είναι το *UUID* που χαρακτηρίζει ξεχωριστά κάθε οντότητα μέσα στον πίνακα. Χρησιμοποιήθηκε *UUID* για λόγους ασφάλειας των request. Στο *RestAPI* έχει δημιουργηθεί μέθοδος που ορίζει ένα unique αλφαριθμητικό με εύρος χαρακτήρων τριάντα έξη (36).
- **user_id**: Είναι *foreign key* που αντιστοιχεί στο *id* του χρήστη, στον οποίο ανήκει το ρομπότ. Αλφαριθμητικό με εύρος χαρακτήρων τριάντα έξη (36).
- **robot_id**: Είναι το *foreign key* που αντιστοιχεί στο *id* του ρομπότ, το οποίο ανήκει στον χρήστη. Αλφαριθμητικό με εύρος χαρακτήρων τριάντα έξη (36).

3.6.6 Πίνακας *user_scenario*

Ο πίνακας *user_scenario,* περιέχει τις σχέσεις μεταξύ των οντοτήτων του χρήστη και των σεναρίων. Ο πίνακας αυτός χρησιμοποιείται για να γίνεται εξατομίκευση των σεναρίων, στα οποία έχει πρόσβαση ο κάθε χρήστης, είτε εκείνα που έχει καταχωρήσει ο ίδιος στην εφαρμογή, είτε εκείνα τους παρέχονται από κάποιον διαχειριστή.

Στην εικόνα (22) παρουσιάζεται η δομή του πίνακα *user_scenario_tbl*.

Column	Type	Nullable
id	char(36)	NO
scenarioId	char(36)	NO
userId	char(36)	YES

Εικόνα 22: Oracle Workbench πίνακας user_scenario

Ανάλυση πεδίων

- **id**: Είναι το *UUID* που χαρακτηρίζει ξεχωριστά κάθε οντότητα μέσα στον πίνακα. Χρησιμοποιήθηκε *UUID* για λόγους ασφάλειας των request. Στο *RestAPI* έχει δημιουργηθεί μέθοδος που ορίζει ένα unique αλφαριθμητικό με εύρος χαρακτήρων τριάντα έξη (36).
- **user_id**: Είναι *foreign key* που αντιστοιχεί στο *id* του χρήστη, στον οποίο ανήκει το ρομπότ. Αλφαριθμητικό με εύρος χαρακτήρων τριάντα έξη (36).
- **scenario_id**: Είναι το *foreign key* που αντιστοιχίζεται με το *id* του σεναρίου, που ανήκει στον χρήστη. Αλφαριθμητικό με εύρος χαρακτήρων τριάντα έξη (36).

3.6.7 Πίνακας *robot_scenario*

Ο πίνακας *robot_scenario*, περιέχει τις σχέσεις μεταξύ των οντοτήτων του ρομπότ και των σεναρίων. Με αυτόν τον τρόπο, καταγράφεται και γίνεται η διαχείριση της εκτέλεσης σεναρίων από το ρομπότ. Κατά την εκκίνηση ενός σεναρίου γίνεται νέα καταγραφή στον πίνακα και η εκτέλεση ξεκινά από κατάσταση (0). Όταν το ρομπότ ξεκινήσει να εκτελεί το σενάριο, τότε η κατάσταση μετατρέπεται σε (1). Στην περίπτωση που ο χρήστης επιθυμεί να σταματήσει την εκτέλεση, μεταβαίνει σε κατάσταση (2). Τέλος, κατά τη λήξη της επιτυχούς εκτέλεσης σεναρίου, μεταβαίνει σε κατάσταση (3) και σε περίπτωση σφάλματος, σε κατάσταση (4).

Στην εικόνα (23) παρουσιάζεται η δομή του πίνακα *robot_scenario*:

Column	Type	Nullable
id	char(36)	NO
robotId	char(36)	YES
scenarioId	char(36)	YES
state	int	NO

Εικόνα 23: Oracle Workbench πίνακας robot_scenario

Ανάλυση πεδίων

- **id**: Είναι το *UUID* που χαρακτηρίζει ξεχωριστά κάθε οντότητα μέσα στον πίνακα. Χρησιμοποιήθηκε *UUID* για λόγους ασφάλειας των request. Στο *RestAPI* έχει δημιουργηθεί μέθοδος που ορίζει ένα unique αλφαριθμητικό με εύρος χαρακτήρων τριάντα έξη (36).
- **scenario_id**: Είναι το *foreign key* που αντιστοιχεί στο *id* του σεναρίου που πρόκειται να εκτελέσει το ρομπότ. Αλφαριθμητικό με εύρος χαρακτήρων τριάντα έξη (36).
- **robot_id**: Είναι το *foreign key* που αντιστοιχεί στο *id* του ρομπότ που θα εκτελέσει το σενάριο. Αλφαριθμητικό με εύρος χαρακτήρων τριάντα έξη (36).
- **state**: Καταγράφεται η κατάσταση εκτέλεσης του σεναρίου: Μηδέν (0), σε κατάσταση αναμονής εκτέλεσης, ένα (1), όταν το ρομπότ έχει λάβει το σενάριο και έχει ξεκινήσει την εκτέλεση, δύο (2), όταν ο χρήστης δίνει εντολή το σενάριο να σταματήσει, τρία (3), όταν το ρομπότ έχει ολοκληρώσει την εκτέλεση του σεναρίου και τέσσερα (4), όταν έχει δημιουργηθεί κάποιο σφάλμα κατά την εκτέλεση του σεναρίου. Είναι οντότητα αριθμού με εύρος ένα (1).

3.6.8 Πίνακας *user_active_logs*

Ο πίνακας *user_active_logs*, περιέχει τις σχέσεις μεταξύ των οντοτήτων από τους πίνακες *users* και *active_logs*. Κατά την εγγραφή του, ο χρήστης δημιουργεί μία οντότητα στον πίνακα *users* με τη μεταβλητή *is_active* σε κατάσταση *false*. Όταν ολοκληρωθεί η εξακρίβωση του *mail* , του μέσω της μοναδικής διεύθυνσης *URL* που δημιουργείται χρησιμοποιώντας το *hash* του αντίστοιχου *active_log*, τότε η μεταβλητή *is_active* μετατρέπεται σε *true* και με αυτόν τον τρόπο δίνεται πρόσβαση στο σύστημα για τον συγκεκριμένο χρήστη.

Στην εικόνα (24) παρουσιάζεται η δομή του πίνακα *user_active_logs*

Column	Type	N...
id	char(36)	NO
activeLogId	char(36)	NO
userId	char(36)	YES

Εικόνα 24: Oracle Workbench πίνακας user_active_log

Ανάλυση πεδίων

- **id**: Είναι το *UUID* που χαρακτηρίζει ξεχωριστά κάθε οντότητα μέσα στον πίνακα. Χρησιμοποιήθηκε *UUID* για λόγους ασφάλειας των request. Στο *RestAPI* έχει δημιουργηθεί μέθοδος που ορίζει ένα unique αλφαριθμητικό με εύρος χαρακτήρων τριάντα έξη (36)
- **activeLogId**: Είναι το *foreign key* που αντιστοιχεί σε οντότητα του πίνακα *active_log*.
- **userId**: Είναι το *foreign key* που αντιστοιχεί σε οντότητα του πίνακα *users*.

3.7 Επεξήγηση Αρχείων

Στην ενότητα αυτή, αναλύονται τα σημαντικότερα αρχεία που δημιουργήθηκαν με σκοπό την ορθή λειτουργία της παρούσας διαδικτυακής πλατφόρμας. Τα αρχεία, στα οποία γίνεται αναφορά παρακάτω, είναι γραμμένα στις γλώσσες που έχουν αναφερθεί στις προηγούμενες ενότητες.

Η αρχική διάκριση των αρχείων έγινε βάσει της λειτουργικής κατηγορίας που ανήκει σε *front-end* και *back-end*. Τα αρχεία που ανήκουν στην κατηγορία *front-end* περιέχουν κυρίως, κώδικα σε *Typescript, HTML, SCSS* και *JSON*. Στην κατηγορία του *back-end* περιέχονται αρχεία σε *JavaScript*. Επιπλέον, έγινε διαχωρισμός κάθε κατηγορίας σε υποκατηγορίες βάση των διεργασιών που επιτελούν.

Οι κατηγορίες του *back-end* είναι οι εξής:

1. **Models**: Σε αυτήν την κατηγορία περιέχονται τα αρχεία που περιγράφουν τη δομή των πινάκων-μοντέλων της βάσης δεδομένων και τις συσχετίσεις που έχουν μεταξύ τους.
2. **Controllers**: Βρίσκονται τα αρχεία στα οποία υλοποιούνται τα αιτήματα του χρήστη. Αναλυτικότερα, γίνεται επεξεργασία των μοντέλων του συστήματος με βάση το αίτημα που έκανε ο χρήστης.
3. **Routes**: Περιέχονται τα αρχεία που είναι υπεύθυνα για την αντιστοίχιση των αιτημάτων με τις μεθόδους υλοποίησης του αιτήματος. Επίσης, σε αυτό το σημείο, γίνεται έλεγχος για το αν ο χρήστης έχει εξουσιοδότηση να λάβει δεδομένα, βάσει ελέγχου με το κλειδάριθμο που έχει λάβει κατά την είσοδό του.
4. **Util**: Σε αυτήν την κατηγορία αρχείων είναι αποθηκευμένες στατικές παράμετροι, που χρειάζονται για τη σύνδεση του back-end με τη βάση δεδομένων και για την επικοινωνία με την υπηρεσία *SMTP*. Ακόμη, περιέχονται αρχεία που ευθύνονται για την αποθήκευση των αρχείων *.zip* στο διακομιστή σε καθορισμένη μορφή.
5. **ZipFiles**: Είναι ο φάκελος στον οποίο αποθηκεύονται τα αρχεία *.zip*, τα οποία με τη σειρά τους περιέχουν τον κώδικα των σεναρίων όταν δημιουργείται ένα νέο σενάριο στο σύστημα.

Στην κατηγορία του *front-end*, έγινε διαχωρισμός των αρχείων στις υποκατηγορίες *shared* και *app*. Αναλυτικότερα, τα αρχεία που ανήκουν στην κατηγορία *shared* περιέχουν κώδικα και εικόνες που χρησιμοποιούνται επανειλημμένα από την κατηγορία *app*. Στην κατηγορία *app* περιέχονται τα αρχεία δομής της διαδικτυακής εφαρμογής που χωρίζονται βάσει των *modules* στα οποία ανήκουν.

1. **Shared**: Περιέχονται *component*, τα οποία χρησιμοποιούνται επανειλημμένα μέσα στον κώδικα. Τα *component* αυτά είναι τα εξής:
    a. **Alert**: Εμφανίζει το παράθυρο ειδοποίησης σε περίπτωση που υπάρξει κάποια λάθος συναλλαγή δεδομένων με το *back-end*
    b. **Language Select**: Περιέχει τις μεθόδους διαχείρισης της γλώσσας στην οποία θα εμφανίζονται τα στοιχεία της εφαρμογής.
2. **App**: Σε αυτό βρίσκονται οι φάκελοι που περιέχουν τα *components* από τα οποία αποτελείται η διαδικτυακή εφαρμογή:
    a. **Auth**: Περιέχει τα *components* με τα οποία δημιουργείται η σελίδα εισόδου και εγγραφής του χρήστη. Επίσης, περιέχει αρχεία τα οποία είναι κρίσιμα για τον έλεγχο πρόσβασης του χρήστη σε σημεία της σελίδας.
    b. **User**: Περιέχει τα *component* τα οποία αναφέρονται στις σελίδες παρακολούθησης όλων των χρηστών, τη σελίδα ελέγχου λογαριασμού και τη σελίδα εγγραφής νέου χρήστη από τον διαχειριστή.
    c. **Robot**: Περιέχει τα *component* τα οποία αναφέρονται στις σελίδες παρακολούθησης των ρομπότ που έχουν καταχωρηθεί στο σύστημα, καθώς και τη σελίδα καταχώρησης νέου ρομπότ στο σύστημα.
    d. **Scenario**: Περιέχει τα *component* τα οποία αναφέρονται στις σελίδες παρακολούθησης των σεναρίων που έχουν καταχωρηθεί στο σύστημα. Επιπλέον, περιέχονται οι μέθοδοι με τις οποίες δίνεται εντολή εκτέλεσης σεναρίου στο ρομπότ. Τέλος, σε αυτό βρίσκεται και η φόρμα με την οποία πραγματοποιείται νέα καταχώρηση σεναρίου στο σύστημα.
    e. **Assign**: Σε αυτό το σημείο, βρίσκονται τα αρχεία που περιέχουν τον κώδικα για τον πίνακα στον οποίο περιγράφονται οι αναθέσεις των σεναρίων στους χρήστες του συστήματος. Επίσης, περιέχει και λειτουργία δημιουργίας αντιστοιχίσεων χρήστη με σενάριο.

f. **MainNav**: Σε αυτόν τον φάκελο περιέχονται τα αρχεία με τα οποία δομείται το μενού της σελίδας. Ακόμα, περιέχει μεθόδους που ορίζουν, ποια σημεία της σελίδας είναι ορατά στον κάθε χρήστη βάσει των δικαιωμάτων τους.

Κατά την κατασκευή της διαδικτυακής εφαρμογής για τη διαχείριση ρομποτικών συστημάτων δημιουργήθηκαν συνολικά 158 αρχεία, ώστε να επιτευχθεί πλήρης και ασφαλής λειτουργία της εφαρμογής. Παρακάτω πραγματοποιείται ανάλυση των αρχείων που περιέχουν τις πιο κρίσιμες λειτουργίες της εφαρμογής.

3.7.1 Back-End

- **server.js**

Είναι το αρχείο, που καλείται για να ενεργοποιηθεί το *web service* του *API*. Αρχικά, έχουν δημιουργηθεί μέθοδοι που ευθύνονται για την ομαλή λειτουργία του *API*. Σε περίπτωση που προκύψει κάποιο σφάλμα στο σύστημα κατά τη διάρκεια επεξεργασίας δεδομένων, οι μέθοδοι που περιέχονται στο αρχείο θα εκτελέσουν μία σειρά από ενέργειες, ώστε να μεταδοθούν πληροφορίες που θα ωφελούν τον χρήστη για την καλύτερη πληροφόρησή του. Ακόμη, παρέχονται πιστοποιητικό και κλειδί κρυπτογράφησης, ώστε να δημιουργηθούν ασφαλείς δίαυλοι επικοινωνίας https, μεταξύ του *front-end* και του *back-end*.

Στην εικόνα (25) παρουσιάζεται μέρος του κώδικα από το αρχείο *server.js*.

server.js > ...

```
44  const onListening = () => {
45    const addr = server.address();
46    const bind = typeof addr === "string" ? "pipe " + addr : "port " + port;
47    debug("Listening on " + bind);
48  };
49
50  var privateKey = fs.readFileSync(__dirname + "/ssl/server.key", "utf8");
51  var certificate = fs.readFileSync(__dirname + "/ssl/server.crt", "utf8");
52  var credentials = { key: privateKey, cert: certificate };
53
54  var httpServer = http.createServer(app);
55  var httpsServer = https.createServer(credentials, app);
56
57  httpServer.listen(8087);
58  httpsServer.listen(8443);
```

Εικόνα 25: Κώδικας αρχείου server.js

- **app.js**

Στο αρχείο αυτό, ορίζονται όλες οι παράμετροι και τα πακέτα που χρησιμοποιούνται από το *API*, ώστε να γίνει σωστή δρομολόγηση των *request* που κάνει ο χρήστης στους *controller*, με σκοπό να τα ικανοποιήσουν. Επιπλέον, ορίζονται οι παράμετροι *Cross-Origin* ώστε ο διακομιστής να παρέχει πρόσβαση στα αιτήματα *HTTP* που γίνονται από τον χρήστη. Ακόμη, σε αυτό το σημείο ορίζεται ο τρόπος με τον οποίο στέλνονται τα αρχεία της ιστοσελίδας καθώς, επίσης, και η διαχείριση των αρχείων *.zip* που στέλνει ο χρήστης για αποθήκευση στο σύστημα.

Στην εικόνα (26) παρουσιάζεται μέρος του κώδικα από το αρχείο *app.js*.

app.js > app.use() callback

```
31  // Manage files of service (angular/zip)
32  app.use(bodyParser.json());
33  app.use(bodyParser.urlencoded({ extended: false }));
34  app.use('/zipFiles', express.static(path.join(__dirname, 'zipFiles')));
35  app.use('/', express.static(path.join(__dirname, 'angular')));
36
37  // CORS authorization
38  app.use((req, res, next) => {
39    res.setHeader('Access-Control-Allow-Origin', '*');
40    res.setHeader('Access-Control-Allow-Methods', 'GET, POST, PUT, PATCH, DELETE');
41    res.setHeader('Access-Control-Allow-Headers', 'Origin, X-Requested-With, Content-Type, Accept, Authorization');
42    next();          You, 8 days ago • Code Review
43  });
44
45  // Use routes for the requests
46  app.use('/robot', robotRoutes);
47  app.use('/user', userRoutes);
48  app.use('/scenario', scenarioRoutes);
49  app.use('/activation', activeLogRoutes);
50  app.use('/user_scenario', userScenarioRoutes);
51  app.use('/robot_scenario', robotScenarioRoutes);
52  app.use((req, res, next) => {
53    res.sendFile(path.join('angular', 'index.html'));
54  });
```

Εικόνα 26: Κώδικας αρχείου app.js

- **check-auth.js**

Μέσα σε αυτό το αρχείο, έχουν οριστεί δύο (2) μέθοδοι που εκτελούνται για κάθε *request* που κάνει ο χρήστης ή το ρομπότ. Κατά τη διάρκεια ικανοποίησης των *request*, οι μέθοδοι ελέγχουν αν μέσα στο *request* περιέχεται το *header*, στο οποίο είναι αποθηκευμένο το κλειδί (*token*) που λαμβάνει ο χρήστης και το ρομπότ κατά την είσοδό τους στο σύστημα. Τα *route* που σχετίζονται με την είσοδο και εγγραφή του χρήστη, την επανέκδοση κωδικού για τον χρήστη, την υπενθύμιση ονόματος χρήστη και την είσοδο του ρομπότ, δεν ελέγχεται ο *header* του κλειδιού, διότι αυτά θεωρούνται *request* του ανώνυμου χρήστη.

Στην εικόνα (27) παρουσιάζεται μέρος του κώδικα από το αρχείο check-auth.*js*.

Εικόνα 27: Κώδικας αρχείου check-auth.js

3.7.2 Front-End

- **app.module.ts**

Σε αυτό το αρχείο, γίνεται η δήλωση όλων των πακέτων και component που χρησιμοποιούνται από τη διαδικτυακή εφαρμογή. Ακόμη, έχει υλοποιηθεί η μέθοδος που χειρίζεται τη γλώσσα στην οποία εμφανίζονται τα στοιχεία της εφαρμογής. Το αρχείο αυτό θεωρείται ο πυρήνας της διαδικτυακής εφαρμογής.

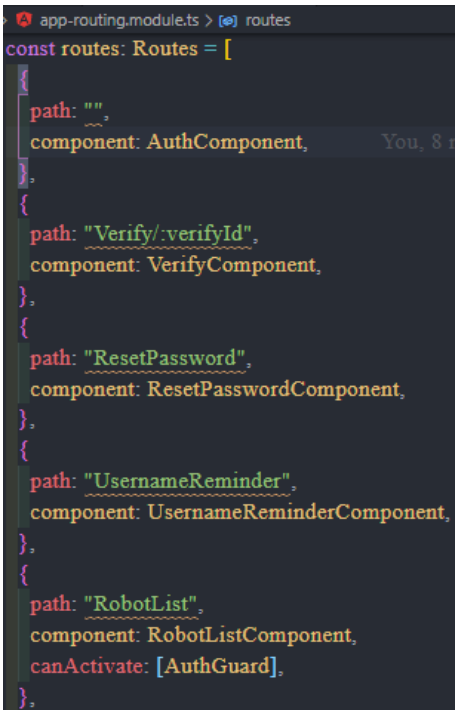
Στην εικόνα (28) παρουσιάζεται μέρος του κώδικα από το αρχείο *app.module.ts*.

Εικόνα 28: Κώδικας αρχείου app.module.ts

- **app-routing.module.ts**

Για την ομαλή περιήγηση του χρήστη εντός της εφαρμογής έχουν οριστεί *URL*, τα οποία χρησιμοποιούνται για τη μετάβαση του χρήστη από τη μία σελίδα στην άλλη. Τα *URL* αυτά, δημιουργούνται πραγματοποιώντας αντιστοίχιση ορισμένων λεκτικών παραμέτρων με τα *component* που θα αντιπροσωπεύουν. Επιπλέον, χρησιμοποιώντας τα *Route Guard*, διασφαλίζεται η ασφάλεια των δεδομένων από ανεπιθύμητους χρήστες

Στην εικόνα (29) παρουσιάζεται μέρος του κώδικα από το αρχείο *app-routing.module.ts*.

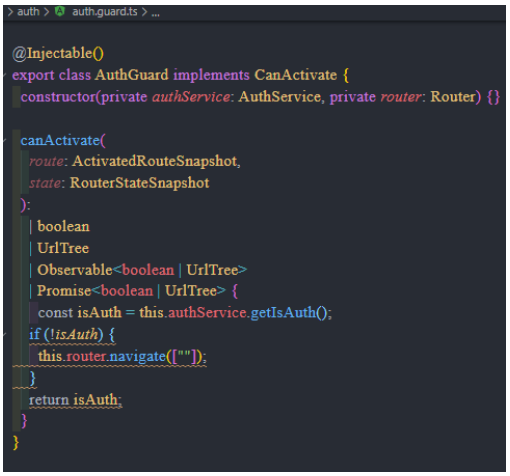


Εικόνα 29: Κώδικας αρχείου app-routing.module.ts

- **auth.guard.ts**

Το συγκεκριμένο αρχείο κατηγοριοποιείται ως route guard. Τα *route guard* της *Angular* είναι διασυνδέσεις που ενημερώνουν, τον router της εφαρμογής κατά τη διάρκεια της πλοήγησης στη σελίδα. Αναλυτικότερα, μέσω των route guard ορίζεται κατά πόσο, ο χρήστης μπορεί να προηγηθεί στη ζητούμενη σελίδα ελέγχοντας, αν έχει καταχωρηθεί κλειδί (*token*) για τον χρήστη και αν το κλειδί αυτό είναι έγκυρο.

Στην εικόνα (30) παρουσιάζεται μέρος του κώδικα από το αρχείο *auth.guard.ts*.

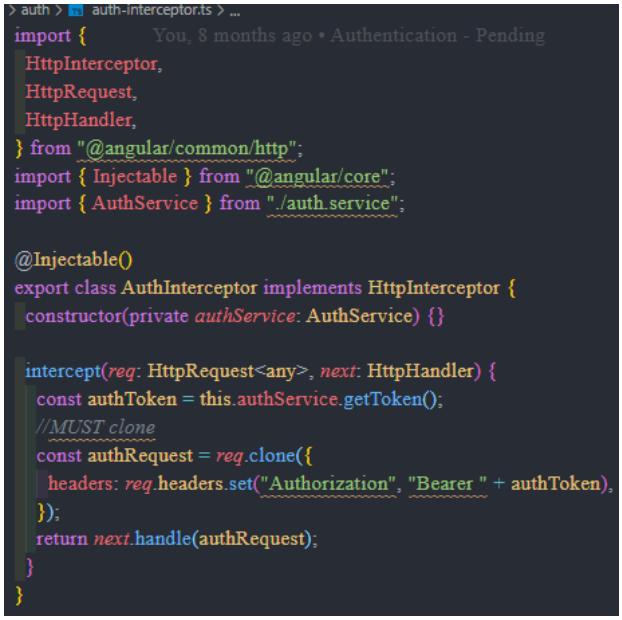


Εικόνα 30: Κώδικας αρχείου auth.guard.ts

- **auth-interceptor.ts**

Οι *Angular interceptor* είναι μέθοδοι που ενεργοποιούνται όταν γίνεται κάποιο *HTTP request* από τον χρήστη στο *back end* και αντίστροφα. Στη συγκεκριμένη περίπτωση, ο ρόλος του *interceptor* είναι να εισάγει το κλειδί (*token*) που έχει λάβει ο χρήστης, κατά την είσοδό του στην εφαρμογή, στους *header* του *request*. Με αυτόν τον τρόπο εξασφαλίζεται η ομαλή και ασφαλής διάδοση δεδομένων με το *back end*.

Στην εικόνα (31) παρουσιάζεται μέρος του κώδικα από το αρχείο *auth-interceptor.ts*.

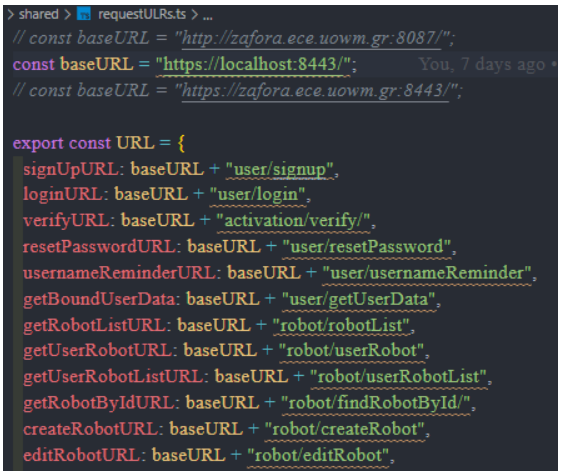


Εικόνα 31: Κώδικας αρχείου auth-interceptor.ts

- **requestURL.ts**

Για να γίνει οποιοδήποτε *HTTP request* στο *back end* της εφαρμογής απαιτείται να εισάγουμε το *URL*, στο οποίο και θα πραγματοποιηθεί το *request* αυτό. Για την καλύτερη υποστήριξη και επεκτασιμότητα της εφαρμογής ορίστηκε το αρχείο *request URL* που περιέχει όλα τα *URL* που είναι απαραίτητα για την επικοινωνία του *front end* με το *back end*

Στην εικόνα (32) παρουσιάζεται μέρος του κώδικα από το αρχείο *requestURL.ts*.

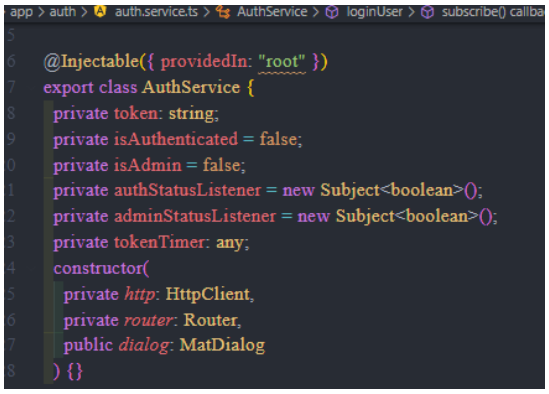


Εικόνα 32: Κώδικας αρχείου requestURL.ts

- **auth.service.ts**

Είναι το αρχείο που διαχειρίζεται την επικοινωνία με το *back end* κατά τη διάρκεια εισόδου ή εγγραφής του χρήστη. Επιπλέον, αποθηκεύονται τιμές μεταβλητών που χρησιμοποιούνται για την εξακρίβωση των δικαιωμάτων του χρήστη στις σελίδες της εφαρμογής.

Στην εικόνα (33) παρουσιάζεται μέρος του κώδικα από το αρχείο *auth.service.ts*.



Εικόνα 33: Κώδικας του αρχείου auth.service.ts

- **scenarioList.component.ts**

Το *scenarioList component*, περιέχει την κύρια λειτουργία της διαδικτυακής εφαρμογής. Η λειτουργικότητα της σελίδας ορίζεται από το αρχείο *scenarioList.component.ts*. Σε αυτό το αρχείο έχει κατασκευαστεί ο κώδικας που είναι υπεύθυνος για τα εξής:

1. Δημιουργία του πίνακα παρακολούθησης των σεναρίων
2. Μετάδοση εντολής εκτέλεσης του σεναρίου
3. Αναζήτηση σεναρίου του πίνακα
4. Μετάβαση στη σελίδα καταχώρησης νέου σεναρίου
5. Σύνδεση του χρήστη με το ρομπότ που του αντιστοιχεί
6. Λήψη του αρχείου *.zip* για κάθε σενάριο του πίνακα
7. Διαγραφή σεναρίου από το σύστημα.

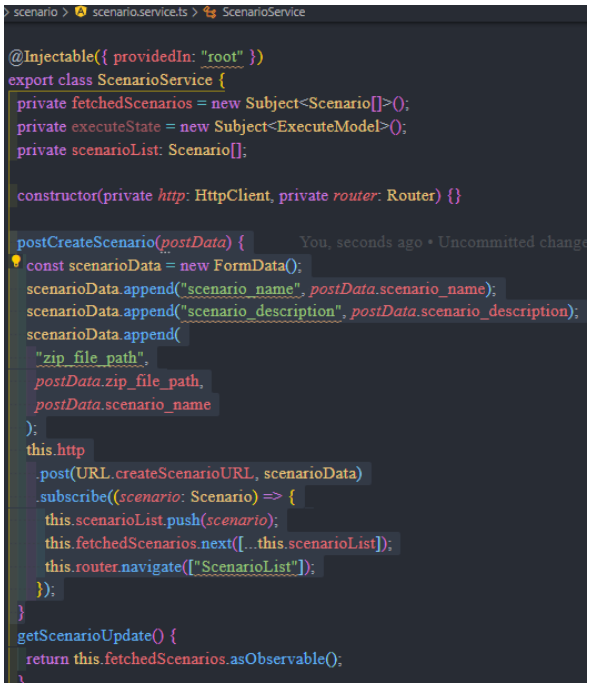
Στην εικόνα (34) παρουσιάζεται μέρος του κώδικα από το αρχείο *scenarioList.component.ts*.

Εικόνα 34: Κώδικας του αρχείου scenarioList.component.ts

- **scenario.service.ts**

Σε αυτό το αρχείο, δημιουργούνται τα *request* προς το *back-end* ώστε να εκπληρωθούν οι λειτουργίες που αναφέρθηκαν παραπάνω, ενώ έχουν υλοποιηθεί όλες οι τεχνικές για *HTTP request* που χρησιμοποιούνται και στα υπόλοιπα *component* της εφαρμογής. Ακόμα, για την καταχώρηση νέων σεναρίων έχει χρησιμοποιηθεί ξεχωριστή τεχνική αποστολής των δεδομένων, καθώς είναι το μοναδικό σημείο στο οποίο γίνεται τέτοιου είδους διαδικασία.

Στην εικόνα (35) παρουσιάζεται μέρος του κώδικα από το αρχείο *scenario.service.ts*.

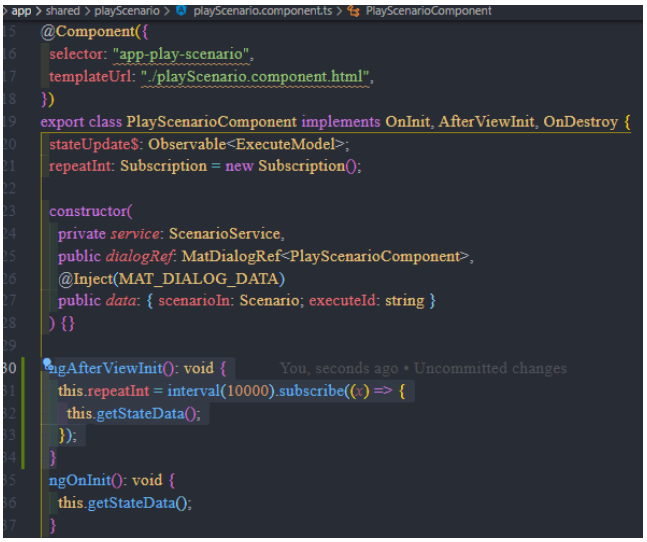


Εικόνα 35: Κώδικας του αρχείου scenario.service.ts

- **playScenario.component.ts**

Το συγκεκριμένο *component*, ενεργοποιείται όταν ο χρήστης δώσει εντολή εκτέλεσης σεναρίου. Κατά τη διάρκεια εκτέλεσής του έχει υλοποιηθεί τεχνική *polling request*. Χρησιμοποιώντας αυτήν την τεχνική γίνονται συνεχόμενα *request* σε καθορισμένο URL, έως ότου βρεθεί αλλαγή στην τιμή που έχει καθοριστεί.

Στην εικόνα (36) παρουσιάζεται μέρος του κώδικα από το αρχείο *playScenario.component.ts*.



Εικόνα 36: Κώδικας του αρχείου playScenario.component.ts

3.8 Ασφάλεια Συστήματος

Η λειτουργικότητα ενός πληροφοριακού συστήματος θεωρείται βασική προδιαγραφή για την ποιότητά του. Εξίσου ύψιστης σημασίας θεωρείται, όμως, και η ασφάλεια ενός συστήματος, ώστε να προλαμβάνονται τυχόν απειλές προς αυτό.

Η δημοφιλέστερη τεχνική ασφαλείας που εφαρμόζεται σε πληθώρα διαδικτυακών εφαρμογών είναι το πρωτόκολλο *HTTPS (HyperText Transfer Protocol Secure)*, μέσω του οποίου κρυπτογραφούνται τα δεδομένα που μεταδίδονται μεταξύ χρήστη και διακομιστή μέσω κρυπτογράφησης *SSL* ή *TLS*. Στα πλαίσια αυτής της διπλωματικής χρησιμοποιήθηκε κρυπτογράφηση *TLS*.

Το πρωτόκολλο *TLS*, στοχεύει κυρίως στην παροχή απορρήτου και ακεραιότητας δεδομένων κατά την επικοινωνία μεταξύ δύο ή περισσότερων υπολογιστών. Για την ικανοποίηση των

απαιτήσεων του πρωτοκόλλου *TLS*, πρέπει να ικανοποιείται μία ή περισσότερες από τις ακόλουθες ιδιότητες:

1. Ιδιωτική σύνδεση, που βασίζεται σε αλγόριθμο συμμετρικού κλειδιού, ο οποίος χρησιμοποιείται για την κρυπτογράφηση των μεταδιδόμενων δεδομένων. Τα κλειδιά είναι μοναδικά για κάθε σύνδεση που γίνεται με τον διακομιστή.
2. Η ταυτότητα των επικοινωνιακών μερών μπορεί να πιστοποιηθεί χρησιμοποιώντας κρυπτογράφηση δημόσιου κλειδιού.
3. Η σύνδεση είναι αξιόπιστη, καθώς κάθε μήνυμα που μεταδίδεται περιλαμβάνει έλεγχο ακεραιότητας μηνύματος χρησιμοποιώντας έναν κωδικό ελέγχου ταυτότητας αυτού για να αποφευχθεί η μη εντοπισμένη απώλεια ή αλλοίωση των δεδομένων κατά τη μετάδοση.

3.8.1 Τεχνικές Ασφάλειας του Συστήματος

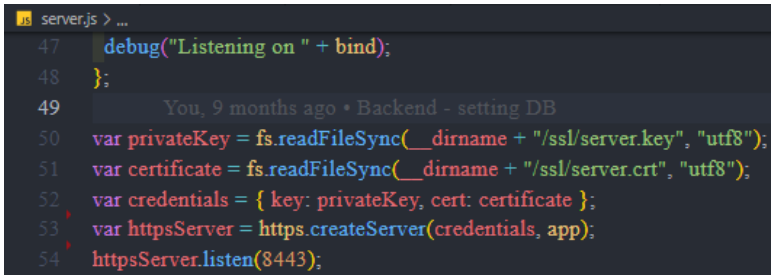
Για την εξασφάλιση της ασφάλειας του συστήματος, χρησιμοποιήθηκαν τεχνικές κρυπτογράφησης και επικύρωσης των δεδομένων στο μέρος του *back-end*, όπου ο χρήστης δεν έχει πρόσβαση.

- **Κρυπτογράφηση**

Για να επιτευχθεί ιδιωτική και αξιόπιστη σύνδεση μεταξύ των υπολογιστών του χρήστη και του διακομιστή, χρησιμοποιήθηκε πιστοποιητικό με κλειδί πιστοποίησης. Κατά την έναρξη μετάδοσης δεδομένων ο διακομιστής ελέγχει αν το πιστοποιητικό που λαμβάνει από τον χρήστη είναι έγκυρο, ώστε να δεχτεί τα αιτήματα που πρόκειται να γίνουν. Στη συνέχεια, για κάθε αίτημα που κάνει ο χρήστης γίνεται κρυπτογράφηση των δεδομένων βάσει ιδιωτικού κλειδιού (*private key*).

Στο *back-end* χρησιμοποιείται το πακέτο *https* της *node* ώστε να υπάρχει πρόσβαση στις μεθόδους που ορίζουν το πιστοποιητικό και το κλειδί κρυπτογράφησης, το οποίο με τη σειρά του θα χρησιμοποιήσει ο διακομιστής κατά την επικοινωνία του με τον χρήστη.

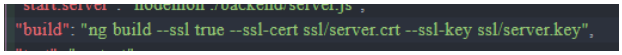
Στην εικόνα (37) παρουσιάζεται το σημείο που ορίζεται η κρυπτογράφηση της σελίδας.



Εικόνα 37: Σημείο κρυπτογράφησης στο κώδικα αρχείου server.js

Στο *front-end* για να οριστεί το πιστοποιητικό και το ιδιωτικό κλειδί για την κρυπτογράφηση των δεδομένων χρησιμοποιώντας τις εντολές που παρέχει το *Angular Framework*, η διαδικασία είναι πολύ απλή. Κατά τη μεταγλώττιση της σελίδας από *Typescript* σε *JavaScript*, ορίζονται οι παράμετροι για το κλειδί και την πιστοποίηση που πρόκειται να χρησιμοποιηθούν μέσα στον φάκελο *package.json*.

Στην εικόνα (37) παρουσιάζεται το σημείο που ορίζεται η εντολή compile του κώδικα της Typescript.



Εικόνα 38: Κώδικας αρχείου package.json

- **Αυθεντικοποίηση και Αξιοπιστία**

Για να επιτευχθεί η ικανοποίηση των ιδιοτήτων του *TLS* πρωτοκόλλου χρειάζεται το πληροφοριακό σύστημα να παρέχει αξιοπιστία της ακεραιότητας των δεδομένων που μεταδίδονται και αυθεντικοποίηση των χρηστών που χρησιμοποιούν τις λειτουργίες. Αυτό επιτυγχάνεται με χρήση μίας αλφαριθμητικής παραμέτρου. Η αλφαριθμητική παράμετρος ορίζεται κατά την είσοδο του χρήστη, στο περιβάλλον που βρίσκονται οι λειτουργίες του συστήματος. Η παράμετρος δημιουργείται βασισμένη στο ιδιωτικό κλειδί του συστήματος και στο μοναδικό *id* του χρήστη που εισάγεται στο σύστημα. Με αυτόν τον τρόπο, η πιθανότητα κλοπής του κλειδιού από μη εξουσιοδοτημένους χρήστες μειώνεται στο ελάχιστο.

Μία ακόμη δυνατότητα που παρέχει η αλφαριθμητική παράμετρος στο σύστημα αποτελεί η εξακρίβωση των στοιχειών του χρήστη για κάθε αίτημα που κάνει στο *back-end*. Μέσω του μηχανισμού *check-auth* η παράμετρος αποκρυπτογραφείται χρησιμοποιώντας το ιδιωτικό κλειδί.

Σαν αποτέλεσμα της αποκρυπτογράφησης, αποθηκεύεται η παράμετρος *id* του χρήστη που έκανε το αίτημα και χρησιμοποιείται για εξακρίβωση δικαιωμάτων στα δεδομένα της βάσης δεδομένων.

Στην εικόνα (39) παρουσιάζεται ο κώδικας του αρχείου *check-auth.js* όπου γίνεται έλεγχος αυθεντικότητας του κλειδιού που παρέχει ο χρήστης στους *headers*.

```
const jwt = require("jsonwebtoken");

module.exports = (req, res, next) => {
  try {
    const token = req.headers.authorization.split(" ")[1];
    const decodedToken = jwt.verify(
      token,
      this.env.server.key
    );
    req.userData = {
      username: decodedToken.username,
      userId: decodedToken.userId,
    };
    next();
  } catch (error) {
    res.status(401).json({ message: "Auth failed" });
  }
};
```

Εικόνα 39: Κώδικας αρχείου check-auth.js

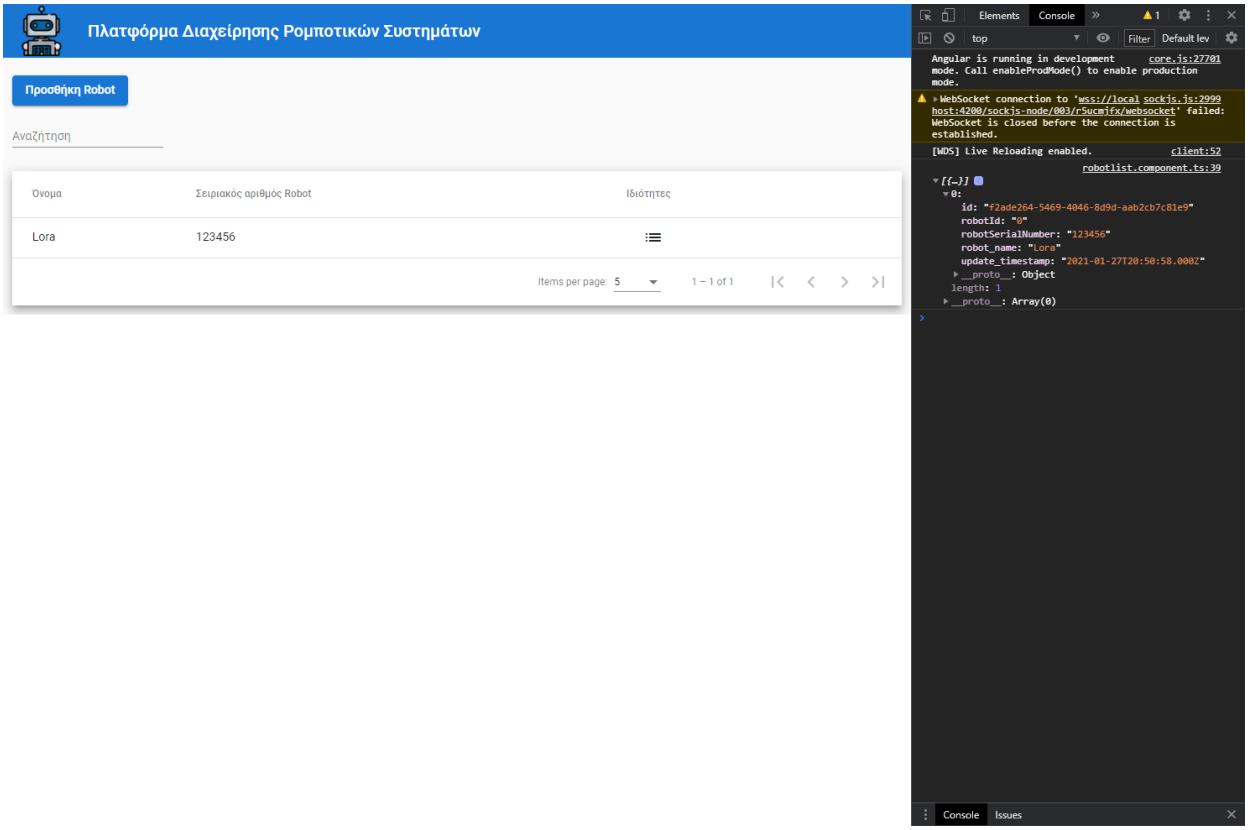
- **Έλεγχος και καταγραφή συμβάντων**

Για την κατασκευή του πληροφοριακού συστήματος χρησιμοποιήθηκε το πρόγραμμα *Visual Studio Code*. Μία από τις επεκτάσεις που παρέχεται από την ηλεκτρονική κοινότητα της εφαρμογής είναι το *Chrome Debugger*.

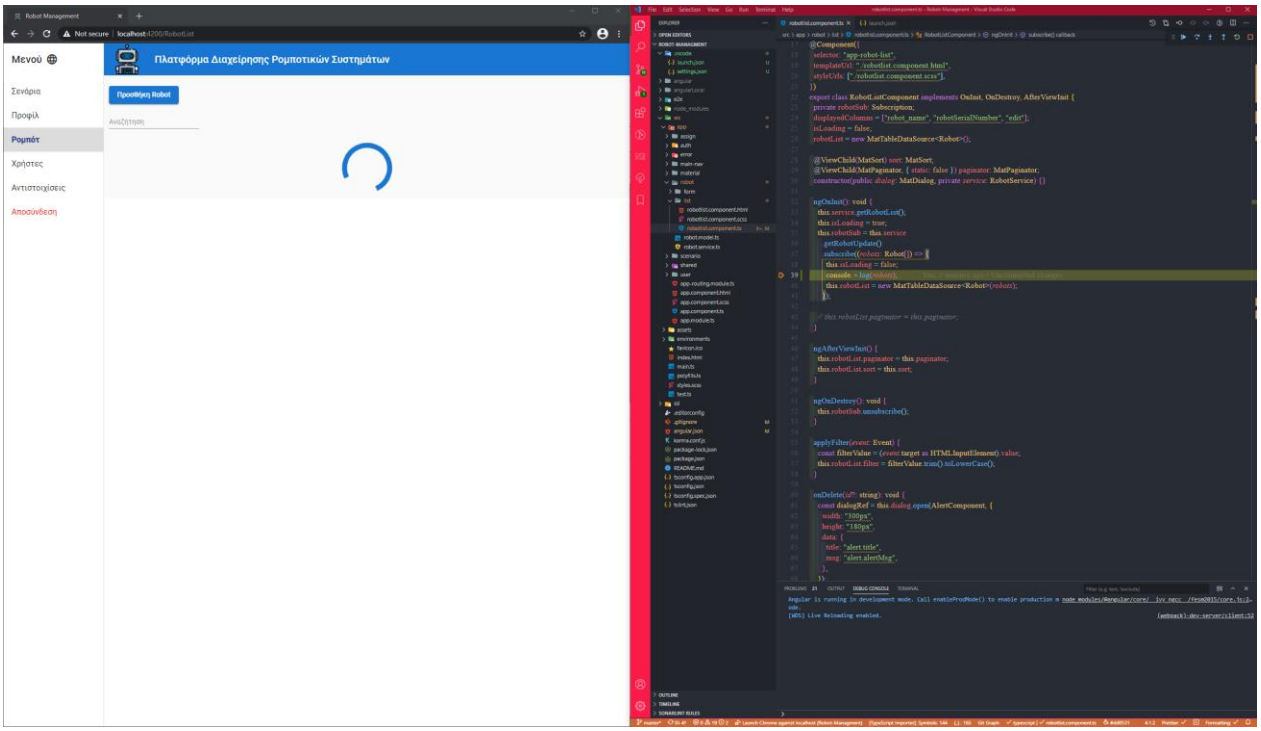
Η επέκταση *Chrome Debugger* είναι, όπως και το όνομα χαρακτηρίζει, ένας μηχανισμός αποσφαλμάτωσης. Ορίζοντας σημεία ελέγχου μέσω της εφαρμογής *Visual Studio Code* επιτυγχάνεται η επίβλεψη εκτέλεσης κώδικα *Typescript*. Με τον τρόπο αυτόν, η αποσφαλμάτωση του κώδικα, η επικύρωση ορθής λειτουργίας της εφαρμογής και η παρακολούθηση της ροής δεδομένων υλοποιείται με μεγάλη ευκολία.

Επιπλέον, χρησιμοποιήθηκε και η κονσόλα του *Chrome* σε συνδυασμό με εντολές *console.log()* παρέχοντας δεδομένα κατάστασης σε πραγματικό χρόνο εκτέλεσης της εφαρμογής.

Στην εικόνα (40) παρουσιάζεται η σελίδα `Ρομπότ` με ανοιχτή τη κονσόλα του *Chrome*. Στην εικόνα (41) παρουσιάζεται η ίδια σελίδα με ανοιχτό το *Visual Studio Code* χρησιμοποιώντας το *Chrome Debugger extension.*



Εικόνα 40: Οθόνη πίνακα ρομπότ με κονσόλα καταγραφής του Chrome



Εικόνα 41: Οθόνη πίνακα ρομπότ με Visual Code Chrome Debugger

3.9 Σύνοψη 3ου κεφαλαίου

Τα ζητήματα που αναλύθηκαν στο 3ο κεφάλαιο αφορούν τη δομή και οργάνωση του πληροφοριακού συστήματος που υλοποιήθηκε. Αρχικά, υπήρξε περιγραφή των στόχων, των λειτουργικών προδιαγραφών καθώς και των περιπτώσεων χρήσης (*Use Cases*).

Ακόμη, έγινε αναλυτική περιγραφή του σχεδιασμού της βάσης δεδομένων, παρουσιάστηκε το σχεσιακό διάγραμμα και αναλύθηκαν τα μοντέλα των πινάκων που αποτελούν τη βάση δεδομένων.

Στη συνέχεια, έγινε εκτενής περιγραφή της δομής του πληροφοριακού συστήματος, τόσο για το *front-end*, όσο και για το *back-end*. Αναλύθηκαν λεπτομερώς τα αρχεία κώδικα που υλοποιούν τις κυριότερες λειτουργίες τους συστήματος. Τέλος, έγινε περιγραφή των πρωτοκόλλων ασφαλείας και οι τεχνικές που υλοποιήθηκαν, ώστε να παρέχεται ασφάλεια κατά τη χρήση της διαδικτυακής εφαρμογής.

Κεφάλαιο 4ο – Διεπαφή Χρήστη (Website User Interface)

Σε αυτό το κεφάλαιο, αναλύονται οι λειτουργίες που παρέχει η διαδικτυακή εφαρμογή στους χρήστες της μέσω *User Interface* (Διεπαφή Χρήστη). Ανάλογα με το είδος του χρήστη που έχει εισέλθει στη σελίδα, παρέχονται και οι αντίστοιχες λειτουργίες. Η εμφάνιση της εφαρμογής ακολουθεί όμοιο μοτίβο για κάθε είδος χρήστη.

Το *User Interface* θεωρείται από τα πιο σημαντικά μέρη της διαδικτυακής εφαρμογής, καθώς αυτό προσφέρει ευκολία χρήσης των λειτουργιών του συστήματος σε χρήστες που δεν είναι εξειδικευμένοι στον τομέα της πληροφορικής. Κατά την κατασκευή του *User Interface*, χρησιμοποιήθηκε *Reactive Programming*, προσφέροντας, με αυτόν τον τρόπο, ευχάριστη εμπειρία χρήσης της εφαρμογής, και *Angular Material theming*, το οποίο με τη σειρά του προσφέρει μινιμαλιστική εμφάνιση στην εφαρμογή.

4.1 Εγγραφή και σύνδεση στο σύστημα (Sign up & Login)

Στην αρχή της περιήγησης ενός ανώνυμου χρήστη στην εφαρμογή συναντάται η σελίδα εισόδου ή εγγραφής του χρήστη. Αξίζει να σημειωθεί, πως όταν ένας χρήστης έχει εισέλθει στη σελίδα τα τελευταία τριάντα (30) λεπτά, τότε προωθείται αυτόματα στη σελίδα ελέγχου σεναρίων και δε χρειάζεται να κάνει είσοδο στην εφαρμογή.

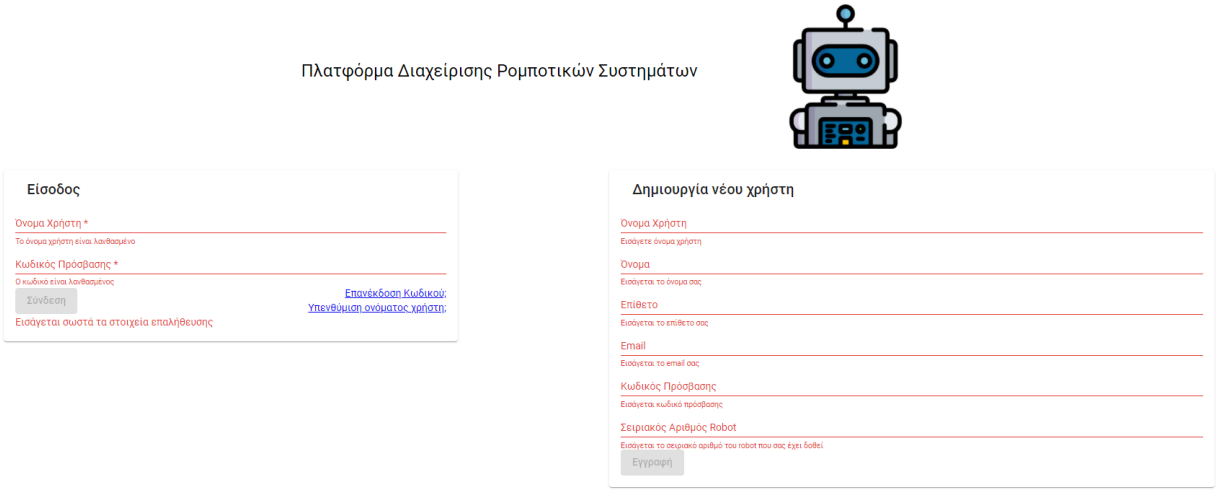
Για να εισέλθει κάποιος χρήστης στις κύριες λειτουργίες του συστήματος απαιτείται η εγγραφή του στο σύστημα και η ταυτοποίησή της μέσω του ηλεκτρονικού ταχυδρομείου που καταχωρεί κατά την εγγραφή του. Στη συνέχεια, εισάγει το όνομα χρήστη και τον κωδικό πρόσβασης που έχει ορίσει. Επιλέγοντας ΄Σύνδεση΄, υποβάλλεται αίτημα ταυτοποίησης στο *back-end* του συστήματος, όπου, ανάλογα με την απόκριση, παρέχεται πρόσβαση στις κύριες λειτουργίες.

Η φόρμα ‘Δημιουργία νέου χρήστη’, παρέχει τη δυνατότητα καταχώρησης νέου χρήστη στο σύστημα. Κατά τη συμπλήρωση στοιχείων, ο χρήστης απαιτείται να ακολουθήσει ορισμένους κανόνες. Αρχικά, είναι αναγκαίο να μην υπάρχουν χρήστες με όμοιο ‘Όνομα Χρήστη’. Σε περίπτωση που γίνει εντοπισμός όμοιας καταχώρησης, εμφανίζεται μήνυμα που προτρέπει τον χρήστη να εισάγει διαφορετικό όνομα. Επιπλέον, κατά την καταχώρηση κωδικού πρόσβασης ο χρήστης χρειάζεται να εισάγει τουλάχιστον έξι (6) χαρακτήρες στο πεδίο και μέχρι να συμπληρωθεί το επιθυμητό πλήθος χαρακτήρων εμφανίζεται μήνυμα ενημέρωσης. Με παρόμοιο τρόπο, όταν

γίνεται η εισαγωγή της ηλεκτρονικής διεύθυνσης, πραγματοποιείται έλεγχος μοτίβου και στην περίπτωση λανθασμένης εισαγωγής πεδίου, εμφανίζεται μήνυμα ενημέρωσης. Τέλος, Κατά την καταχώρηση του σειριακού αριθμού για το ρομπότ, το σύστημα κάνει ταυτοποίηση του αριθμού με τους σειριακούς αριθμούς των ρομπότ που έχουν εισαχθεί στο σύστημα.

Αξιοσημείωτο είναι, πως καθώς έχει οριστεί μία συνθήκη κατά την είσοδο ή εγγραφή ενός χρήστη, σε περίπτωση που το σύστημα αναγνωρίσει πως ο χρήστης δεν έχει εισάγει στοιχεία που ακολουθούν τους κανόνες, οι επιλογές 'Σύνδεση' ή 'Εγγραφή' παραμένουν απενεργοποιημένες.

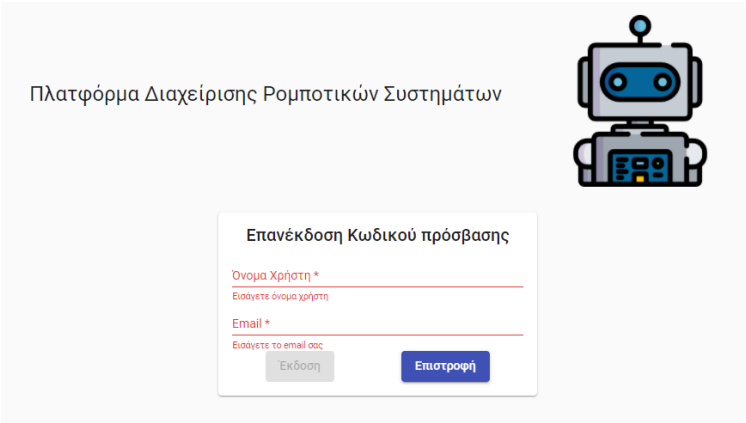
Στην εικόνα (42) παρουσιάζεται η οθόνη εισόδου και εγγραφής χρήστη.



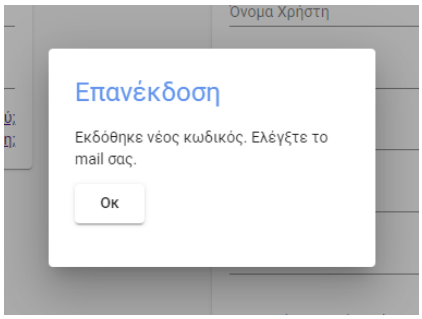
Εικόνα 42: Σελίδα Εισόδου και Εγγραφής

4.2 Επανέκδοση Κωδικού

Ένα πολύ σύνηθες φαινόμενο κατά την περιήγηση των χρηστών στο διαδίκτυο, είναι να ξεχάσουν τον κωδικό πρόσβασης, με τον οποίο έχουν συνδέσει έναν λογαριασμό. Για αυτόν τον λόγο, έχει κατασκευαστεί λειτουργία, που επιτρέπει στον χρήστη να εκδώσει νέο κωδικό, που θα σταλεί στον λογαριασμό του. Για την έκδοση νέου κωδικού, χρειάζεται ο χρήστης να εισάγει το όνομα χρήστη και το ηλεκτρονικό του ταχυδρομείο, εικόνα (43). Η φόρμα συμπλήρωσης στοιχείων περιλαμβάνει μεθόδους που ελέγχουν το μοτίβο εισαγωγής email. Αφού ο χρήστης επιλέξει 'Έκδοση', μεταβαίνει στη σελίδα 'Εισόδου' και τότε εμφανίζεται μήνυμα προτροπής ελέγχου των εισερχομένων μηνυμάτων στο email του, εικόνα (44).



Εικόνα 43: Σελίδα επανέκδοσης κωδικού πρόσβασης

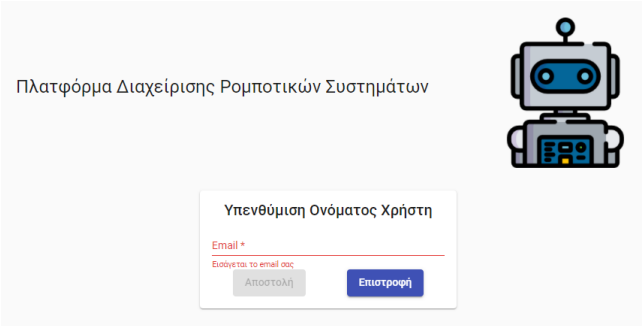


Εικόνα 44: Μήνυμα ολοκλήρωσης διαδικασίας

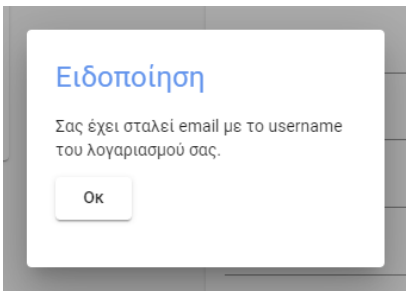
4.3 Υπενθύμιση Ονόματος Χρήστη

Στην περίπτωση που ο χρήστης ξεχάσει το όνομα χρήστη που έχει καταχωρήσει στον λογαριασμό του, του δίνεται η δυνατότητα στη σελίδα εισόδου να του σταλεί email υπενθύμισης. Για να ενεργοποιηθεί αυτή η διαδικασία, ζητείται στον χρήστη να συμπληρώσει τη φόρμα εξακρίβωσης του *email*, εικόνα (45) και αυτόματα γίνεται η εξακρίβωση. Στη συνέχεια, συντάσσεται τυποποιημένο email που περιέχει το όνομα χρήστη στο οποίο αντιστοιχεί η διεύθυνση *email*.

Όταν η διαδικασία ολοκληρωθεί με επιτυχία, τότε εμφανίζεται ειδοποίηση στον χρήστη να ελέγξει τα email του για νέο μήνυμα, εικόνα (46).



Εικόνα 45: Σελίδα υπενθύμισης ονόματος χρήστη



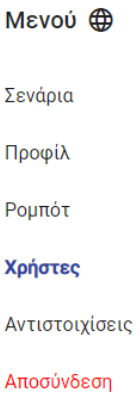
Εικόνα 46: Μήνυμα ολοκλήρωσης διαδικασίας

4.4 Μενού πλοήγησης

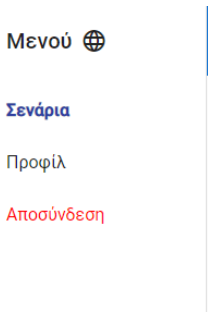
Για τη δυνατότητα πλοήγησης στη σελίδα, έχει δημιουργηθεί μενού πλοήγησης στο οποίο εμφανίζονται οι κατηγορίες λειτουργιών. Κάθε κατηγορία, περιλαμβάνει ένα σύνολο λειτουργιών που ικανοποιεί τις απαιτήσεις του συστήματος που αναφέρθηκαν στα προηγούμενα κεφάλαια.

Οι χρήστες με δικαιώματα διαχειριστή έχουν σαφώς περισσότερες επιλογές σε σχέση με τους απλούς χρήστες, εικόνα (47) και εικόνα (48). Για να επιτευχθεί αυτή η διαφοροποίηση, κατασκευάστηκε ένα μενού πλοήγησης, ενώ χρησιμοποιώντας μεθόδους ελέγχου δικαιωμάτων ενεργοποιούνται οι αντίστοιχες επιλογές για κάθε χρήστη.

Επιπλέον, στο πάνω μέρος του μενού πλοήγησης, δίνεται δυνατότητα αλλαγής γλώσσας χωρίς να γίνει επαναφόρτιση της διαδικτυακής εφαρμογής.



Εικόνα 47: Μενού πλοήγησης του διαχειριστή

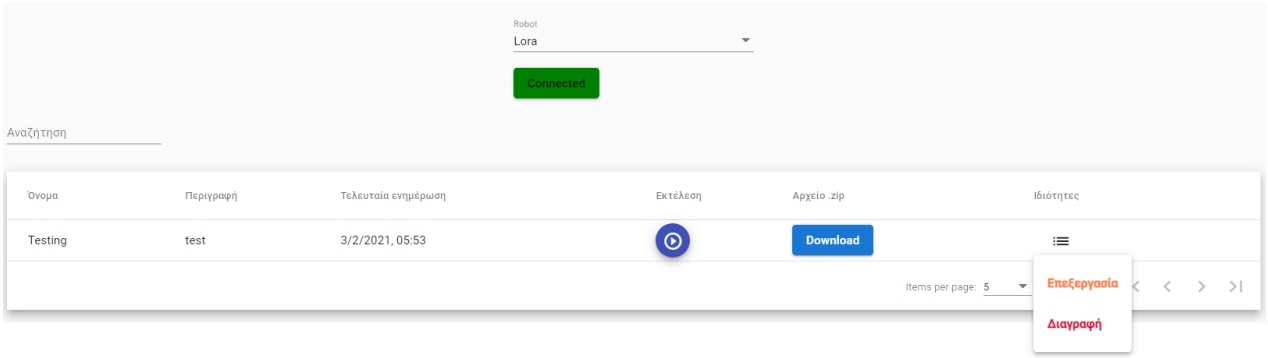


Εικόνα 48: Μενού πλοήγησης του χρήστη

4.5 Διαχείριση Σεναρίων

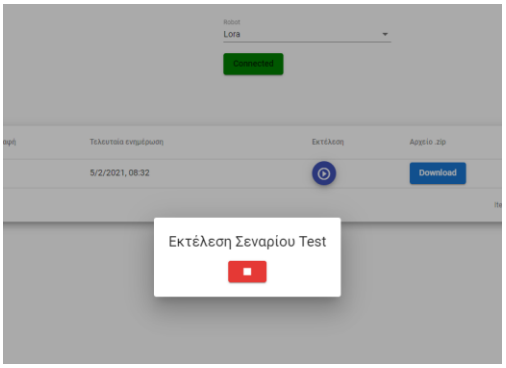
Η σελίδα 'Σενάρια', εμφανίζεται στις επιλογές των χρηστών που έχουν δικαιώματα 'απλού χρήστη' και σε εκείνους που έχουν δικαιώματα 'διαχειριστή'. Σε αυτήν τη σελίδα περιλαμβάνεται ο πίνακας που περιέχει τα καταχωρημένα σενάρια από τον συγκεκριμένο χρήστη. Στην περίπτωση που ο χρήστης έχει δικαιώματα 'διαχειριστή', τότε στον πίνακα εμφανίζονται τα σενάρια που έχουν καταχωρηθεί από όλους τους χρήστες.

Στον πίνακα εμφανίζονται οι πληροφορίες του κάθε σεναρίου και προσφέρονται λειτουργίες λήψης του αρχείου, που περιέχονται στο κάθε σενάριο, εικόνα (49). Ακόμη, δίνονται επιλογές τροποποίησης ή διαγραφής για κάθε οντότητα του πίνακα.



Εικόνα 49: Σελίδα διαχείρισης σεναρίων

Η κύρια λειτουργία της σελίδας είναι η σύνδεση με το ρομπότ και η εντολή εκτέλεσης σεναρίου σε αυτό. Για να επιτευχθεί η εντολή εκτέλεσης σεναρίου, αρχικά ο χρήστης χρειάζεται να επιλέξει το ρομπότ στο οποίο έχει δικαιώματα χρήσης και να συνδεθεί με αυτό. Στη συνέχεια, ενεργοποιείται η επιλογή 'Εκτέλεση' και έτσι δίνεται η δυνατότητα να οριστεί εντολή εκτέλεσης επιλεγμένου σεναρίου. Αφού πραγματοποιηθεί η εκκίνηση της διαδικασίας, οι ευρύτερες λειτουργίες της σελίδας απενεργοποιούνται και εμφανίζεται ένδειξη ενημέρωσης κατάστασης. Σε αυτό το σημείο, ο χρήστης ενημερώνεται για την κατάσταση εκτέλεσης του σεναρίου που έχει επιλέξει. Επίσης, ο χρήστης είναι σε θέση να επιλέξει τερματισμό εκτέλεσης διαδικασίας από το ρομπότ και να επιστρέψει στη σελίδα διαχείρισης σεναρίων, εικόνα (50).



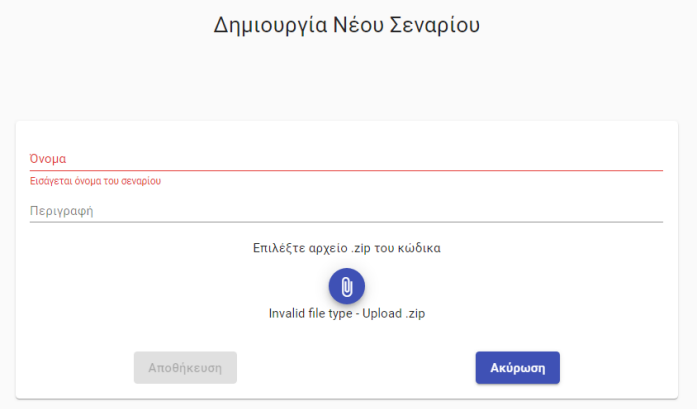
Εικόνα 50: Ένδειξη Εκτέλεσης Σεναρίου

4.6 Καταχώρηση Σεναρίου

Στην προηγούμενη ενότητα 'Σενάρια' περιέχεται η επιλογή 'Προσθήκη νέου Σεναρίου', με αυτόν τον τρόπο ο χρήστης μεταβαίνει σε σελίδα που περιέχει φόρμα καταχώρησης νέου σεναρίου στο σύστημα, εικόνα (51).

Όπως και στις φόρμες που έχουν προαναφερθεί, η επιλογή καταχώρησης είναι απενεργοποιημένη, έως ότου καλυφθούν οι προδιαγραφές καταχώρησης νέου σεναρίου. Για να γίνει αυτό, ζητείται από τον χρήστη να καταχωρήσει κάποιο όνομα που χαρακτηρίζει το σενάριο και να ενσωματώσει το αρχείο *.zip* που περιέχει τον κώδικα που πρόκειται να εκτελέσει το ρομπότ. Σε περίπτωση, που δεν εισάγει κάποιο από τα πεδία σωστά του εμφανίζεται μήνυμα ενημέρωσης λάθους.

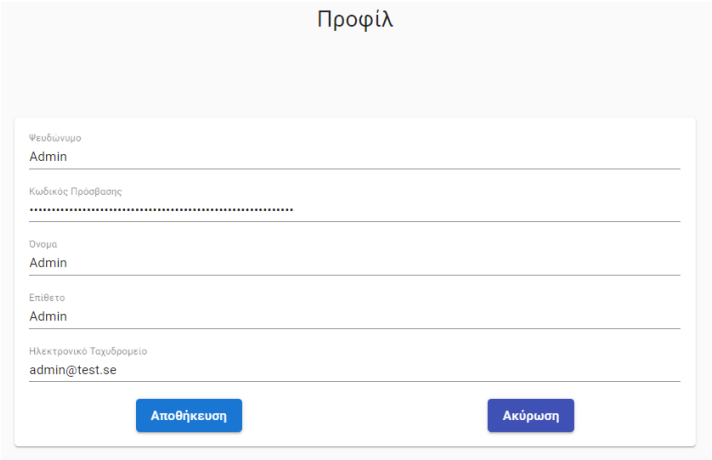
Αφού συμπληρωθεί σωστά η φόρμα, η επιλογή καταχώρησης ενεργοποιείται. Όταν την επιλέξει ο χρήστης, τότε επιστρέφει στη σελίδα διαχείρισης σεναρίων όπου έχει προστεθεί και το νέο σενάριο.



Εικόνα 51: Σελίδα Δημιουργίας Νέου Σεναρίου

4.7 Προφίλ

Στη σελίδα αυτή, ο κάθε χρήστης μπορεί να παρακολουθεί και να επεξεργάζεται τις πληροφορίες που είναι καταχωρημένες για τον λογαριασμό του, εικόνα (52). Και σε αυτήν την περίπτωση, έχουν προστεθεί επιλογές ελέγχου ορθότητας της φόρμας, στην οποία εμφανίζονται μηνύματα ενημέρωσης σφάλματος, όταν ο χρήστης δεν έχει συμπληρώσει πλήρως τα πεδία της φόρμας. Όταν η φόρμα είναι ελλιπώς συμπληρωμένη, τότε απενεργοποιείται η επιλογή αποθήκευσης των αλλαγών.

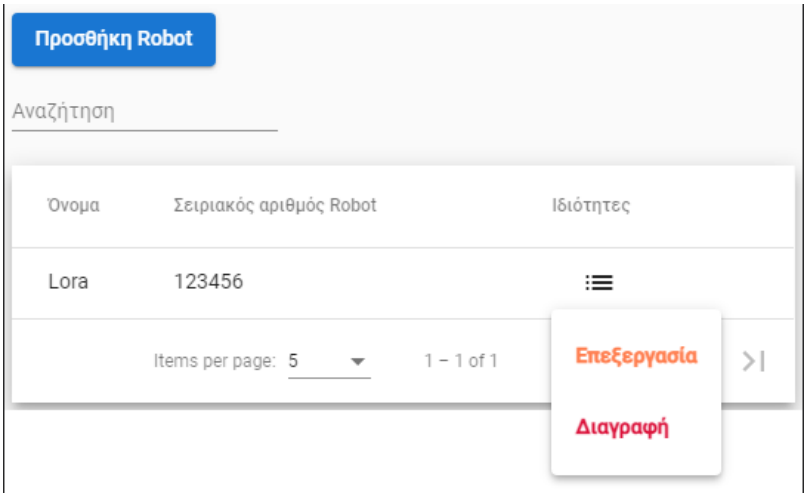


Εικόνα 52: Σελίδα Προφίλ

4.8 Ρομπότ

Η σελίδα 'Ρομπότ', περιέχει τον πίνακα στον οποίο εμφανίζονται όλα τα καταχωρημένα ρομπότ στο σύστημα, εικόνα (53). Στη σελίδα έχουν πρόσβαση χρήστες με δικαιώματα διαχειριστή, στους οποίους παρέχονται λειτουργίες επεξεργασίας ή διαγραφής των οντοτήτων.

Μία ακόμη λειτουργία της σελίδας είναι η επιλογή 'Προσθήκη Ρομπότ', με την οποία ο χρήστης μεταβαίνει στη φόρμα καταχώρησης νέου ρομπότ στο σύστημα.



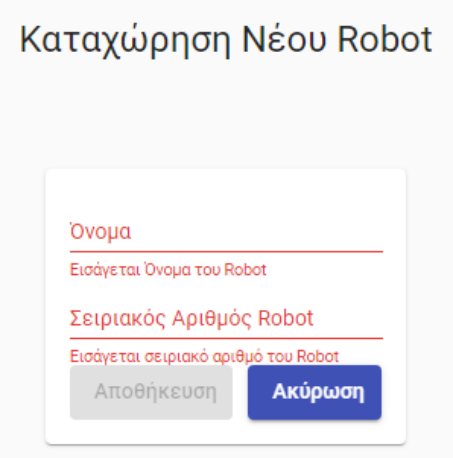
Εικόνα 53: Σελίδα Ρομποτ

4.9 Καταχώρηση Ρομπότ

Στη σελίδα 'Καταχώρηση Ρομπότ', περιέχεται η φόρμα καταχώρησης νέου ρομπότ στο σύστημα. Για την καταχώρηση νέου ρομπότ, ο χρήστης καλείται να συμπληρώσει τα πεδία περιγραφής του ρομπότ. Το πεδίο 'Σειριακός Αριθμός Ρομπότ', χρειάζεται να συμπληρωθεί με την τιμή του αλφαριθμητικού που χρησιμοποιεί το ρομπότ ως κλειδί ταυτοποίησης με το σύστημα, στοχεύοντας στην επιτυχή επικοινωνία του ρομπότ με αυτό όταν ορίζεται εντολή εκτέλεσης σεναρίου.

Όπως και στις υπόλοιπες φόρμες του συστήματος, έχει οριστεί λειτουργία που απενεργοποιεί τη δυνατότητα αποθήκευσης της φόρμας στο σύστημα, στην περίπτωση που δεν έχουν συμπληρωθεί ορθά τα πεδία της φόρμας, εικόνα (54).

Μετά την καταχώρηση της οντότητας ο χρήστης μεταβαίνει στη σελίδα 'Ρομπότ'.



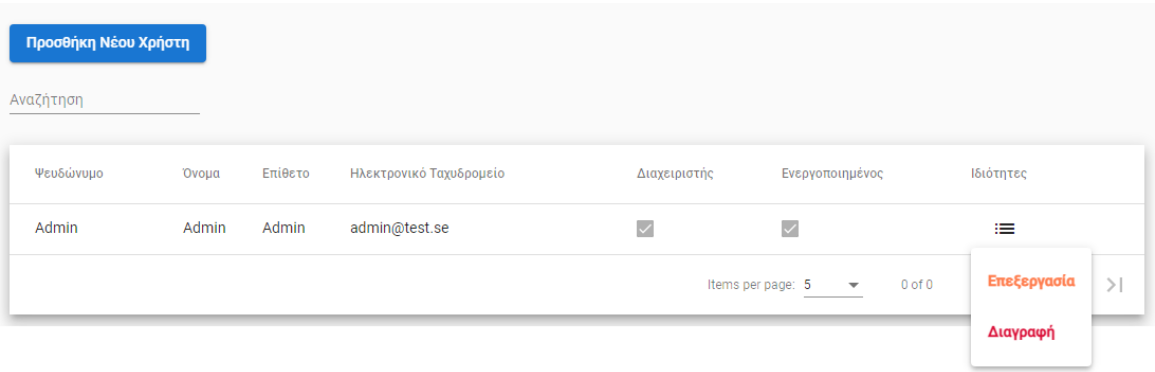
Εικόνα 54: Σελίδα Καταχώρησης Νέου Ρομπότ

4.10 Χρήστες

Στη σελίδα 'Χρήστες' έχουν πρόσβαση μόνο οι χρήστες με δικαιώματα διαχειριστή. Σε αυτό το σημείο της εφαρμογής υπάρχει πίνακας στον οποίο εμφανίζονται όλες οι οντότητες χρηστών που έχουν καταχωρηθεί στο σύστημα καθώς και ορισμένες λεπτομέρειες για κάθε είδος χρήστη. Ακόμη, ο διαχειριστής έχει τη δυνατότητα να διαγράψει τον χρήστη από το σύστημα, εικόνα (55).

Μία επιπλέον δυνατότητα είναι η επεξεργασία των στοιχείων του χρήστη. Με αυτόν τον τρόπο, οι χρήστες με δικαιώματα διαχείρισης έχουν τη δυνατότητα να παραχωρήσουν δικαιώματα διαχείρισης του συστήματος και σε άλλους χρήστες. Επιπλέον, μέσω της επεξεργασίας χρήστη, ο διαχειριστής μπορεί να αναθέσει στο χρήστη διαφορετικό ρομπότ για χρήση.

Τέλος, παρέχεται η επιλογή 'Προσθήκη Νέου Χρήστη', στην περίπτωση που ο διαχειριστής επιθυμεί να καταχωρήσει νέο χρήστη στο σύστημα.

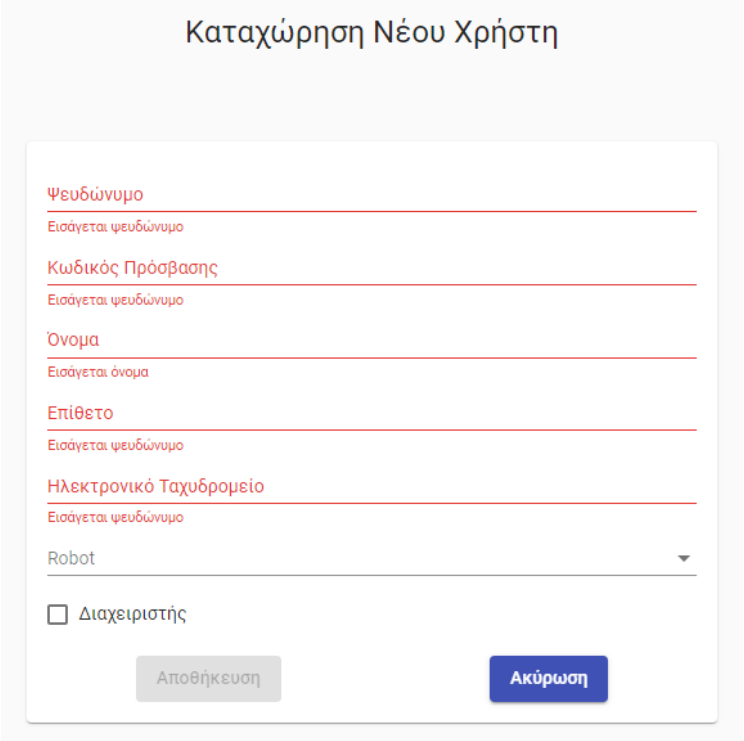


Εικόνα 55: Σελίδα Διαχείρισης Χρηστών

4.11 Καταχώρηση Νέου Χρήστη

Στη σελίδα 'Καταχώρηση Νέου Χρήστη', έχουν πρόσβαση οι χρήστες που έχουν λάβει δικαιώματα διαχειριστή. Η διαφορά σε σχέση με την εγγραφή νέου χρήστη έγκειται στο γεγονός, ότι σε αυτήν την περίπτωση δεν είναι υποχρεωτική η εξακρίβωση του ηλεκτρονικού ταχυδρομείου από τον χρήστη. Όπως και στη φόρμα εγγραφής, έτσι και εδώ απαιτείται από τον χρήστη να συμπληρωθούν ορθά όλα τα πεδία της φόρμας ώστε να ενεργοποιηθεί η επιλογή 'Αποθήκευση', εικόνα (56).

Μία επιπλέον δυνατότητα που προσφέρεται σε αυτήν τη σελίδα είναι η παραχώρηση δικαιωμάτων διαχείρισης στον χρήστη που επρόκειτο να γίνει η καταχώρηση. Σε αυτήν την περίπτωση, η καταχώρηση ρομπότ δεν είναι υποχρεωτική, καθώς οι διαχειριστές έχουν πρόσβαση σε όλα τα ρομπότ που είναι καταχωρημένα στο σύστημα.

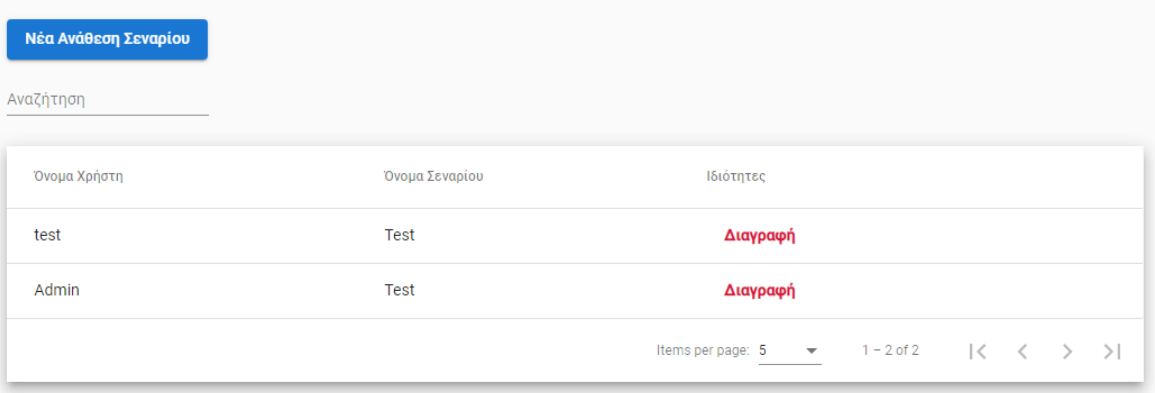


Εικόνα 56: Σελίδα Καταχώρησης Νέου Χρήστη

4.12 Αναθέσεις Σεναρίων

Η σελίδα 'Αναθέσεις Σεναρίων' περιέχει πίνακα που περιλαμβάνει όλες τις συσχετίσεις των οντοτήτων σεναρίων με τις οντότητες των χρηστών. Στη σελίδα αυτή, έχουν πρόσβαση οι χρήστες με δικαιώματα διαχειριστή. Με αυτόν τον τρόπο, ο διαχειριστής μπορεί να παρακολουθεί τα σενάρια που έχει καταχωρήσει κάθε χρήστης. Επιπλέον, δίνεται η δυνατότητα κατάργησης ανάθεσης, ώστε να καταργηθεί η πρόσβαση του επιλεγμένου χρήστη στο αντίστοιχο σενάριο, διαγράφοντας τη καταχώρηση, εικόνα (57).

Τέλος, περιέχεται η επιλογή 'Νέα Ανάθεση Σεναρίου' που αναλύεται παρακάτω.

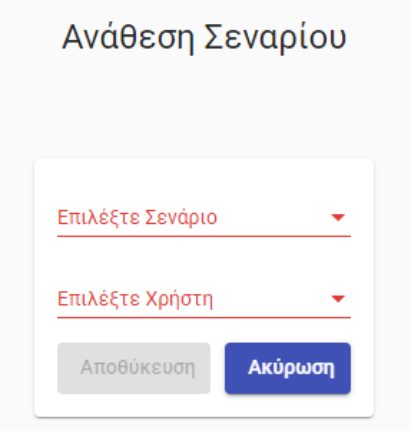


Εικόνα 57: Σελίδα Αναθέσεις Σεναρίων

4.13 Ανάθεση Σεναρίων σε Χρήστη

Στη φόρμα 'Ανάθεση Σεναρίου', έχουν πρόσβαση οι χρήστες με δικαιώματα διαχειριστή. Η λειτουργία αυτή στοχεύει στην επαναχρησιμοποίηση των ήδη καταχωρημένων σεναρίων από νέους χρήστες του συστήματος. Έτσι, ελαττώνεται η ανάγκη μεγάλου αποθηκευτικού χώρου και επιτυγχάνεται διαμοίραση των σεναρίων μέσω του συστήματος.

Όπως και στις υπόλοιπες φόρμες του συστήματος, η επιλογή αποθήκευσης της νέας αντιστοίχισης απενεργοποιείται μέχρι να οριστούν ορθά τα πεδία της φόρμας, εικόνα (58).



Εικόνα 58: Ανάθεση Σεναρίου σε Χρήστη

4.14 Σύνοψη 4ου Κεφαλαίου

Στο κεφάλαιο αυτό, αναλύθηκαν όλες οι λειτουργίες της διαδικτυακής εφαρμογής, καθώς και όλες οι ενέργειες που καλείται να κάνει ένας νέος χρήστης για να εκμεταλλευτεί τις λειτουργίες αυτές. Επιπλέον, για κάθε λειτουργία παρουσιάστηκε οπτικό υλικό στο οποίο απεικονίζεται το περιβάλλον της διαδικτυακής εφαρμογής κατά την περιήγηση του χρήστη. Στο επόμενο κεφάλαιο ακολουθεί συνοπτική αναφορά της διπλωματικής εργασίας, συμπεράσματα, πλεονεκτήματα και προτάσεις για μελλοντικές αναβαθμίσεις.

Κεφάλαιο 5ο – Επίλογος

Στο τελευταίο κεφάλαιο, πραγματοποιείται συνοπτική παρουσίαση των θεμάτων που αναλύθηκαν έως τώρα. Στη συνέχεια παρουσιάζονται τα συμπεράσματα που προέκυψαν κατά τη διαδικασία σχεδίασης και κατασκευής της διαδικτυακής εφαρμογής.

Επιπλέον, γίνεται λεπτομερής ανάλυση των *SWOT* (*Strengths*, *Weaknesses*, *Opportunities*, *Threats*), καθώς και των προτάσεων για μελλοντικές μεθόδους βελτιστοποίησης και επέκτασης του συστήματος.

5.1 Ανακεφαλαίωση Διπλωματικής Εργασίας

Η διαδικτυακή εφαρμογή που σχεδιάστηκε και υλοποιήθηκε στα πλαίσια της διπλωματικής εργασίας, αφορά την απομακρυσμένη διαχείριση ρομποτικών συστημάτων που ενισχύουν τη διδασκαλία στη δευτεροβάθμια εκπαίδευση.

Με τη διαδικτυακή πλατφόρμα που κατασκευάστηκε, παρέχεται η δυνατότητα σε καθηγητές να διεξάγουν μαθήματα σε ρομποτικά συστήματα, χωρίς να είναι υποχρεωτική η τεχνογνωσία που απαιτείται έως τώρα. Μέχρι στιγμής, οι καθηγητές χρειαζόταν να γνωρίζουν τους τρόπους μεταφοράς αρχείων σε ρομποτικά συστήματα, την επικοινωνία ρομποτικών συστημάτων με ηλεκτρονικούς υπολογιστές μέσω δικτύου και τη δημιουργία-εκτέλεση κώδικα σε *Python* για ρομποτικά συστήματα. Η διαδικτυακή εφαρμογή στοχεύει στην αντιμετώπιση αυτών των δυσκολιών, δίνοντας τη δυνατότητα στους καθηγητές να επικεντρώσουν την προσοχή τους στην καλύτερη εκπαίδευση των μαθητών.

Στις λειτουργίες της διαδικτυακής πλατφόρμας έχουν πρόσβαση δύο είδη χρηστών, οι διαχειριστές και οι απλοί χρήστες. Το γραφικό περιβάλλον τροποποιείται δυναμικά για κάθε χρήστη που εισέρχεται στο σύστημα. Οι απλοί χρήστες έχουν πρόσβαση στις σελίδες διαχείρισης σεναρίων και στη σελίδα επεξεργασίας των προφίλ τους. Για τον διαχειριστή προσφέρονται επιπλέον επιλογές όσον αφορά τη διαχείριση των ρομποτικών συστημάτων που έχουν πρόσβαση στο *API* της εφαρμογής. Μία ακόμη λειτουργία στην οποία έχει πρόσβαση ο διαχειριστής αφορά την αντιστοίχιση των σεναρίων σε νέους χρήστες. Τέλος, ο διαχειριστής έχει τη δυνατότητα να επιβλέπει τους λογαριασμούς των χρηστών που έχουν εγγραφεί στο σύστημα.

Κυρίως, για τους χρήστες προσφέρεται η δυνατότητα να δοθεί εντολή εκτέλεσης στο ρομποτικό σύστημα με απλές και εύκολα κατανοητές επιλογές. Επιπλέον η πλατφόρμα, μπορεί να παρέχεται ως ένα μέσο αποθήκευσης των σεναρίων που πρόκειται να εκτελεστούν από το ρομπότ, με σκοπό να μπορεί να δοθεί εντολή εκτέλεσης από οπουδήποτε.

Για τον διαχειριστή της σελίδας, μία από τις επιπλέον επιλογές που του παρέχονται είναι η διαχείριση των χρηστών που έχουν πρόσβαση στο σύστημα και στα δικαιώματα που κατέχουν, καθώς δίνεται η επιλογή παροχής δικαιωμάτων διαχειριστή σε απλούς χρήστες. Επίσης, έχουν τη δυνατότητα να αντιστοιχούν σενάρια που έχουν αποθηκευτεί στο σύστημα και σε άλλους χρήστες. Τέλος, ο διαχειριστής έχει την ευθύνη καταχώρησης και διαχείρισης των ρομπότ που επικοινωνούν με το σύστημα.

Για τη διασφάλιση πρόσβασης των χρηστών στη διαδικτυακή πλατφόρμα, παρέχονται επιλογές υπενθύμισης του ονόματος χρήστη και επανέκδοσης του κωδικού πρόσβασης των λογαριασμών. Επιπλέον, η διαδικτυακή εφαρμογή κατασκευάστηκε ακολουθώντας πρωτόκολλο ασφαλείας *TLS*, ικανοποιώντας τις ιδιότητες κρυπτογραφημένης επικοινωνίας με το *API* βάσει πιστοποιητικού και κλειδιού ασφαλείας. Για τη διασφάλισή κάθε επικοινωνίας με το *API* χρησιμοποιείται ένα μοναδικό κλειδί που εκδίδεται με κάθε σύνδεση του χρήστη στο σύστημα.

Για την κατασκευή της διαδικτυακής εφαρμογής χρησιμοποιήθηκαν οι εξής γλώσσες προγραμματισμού: *HTML 5, SCSS, Typescript* και *JavaScript*. Για την καλύτερη οργάνωση και τις επιπλέον δυνατότητες κατασκευής λειτουργιών χρησιμοποιήθηκαν τα *framework Angular 11* για το *front-end* και *Node Express framework* για το *back-end*. Ακόμη, για τη βελτιστοποίηση της αισθητικής του περιβάλλοντος του χρήστη, έγινε χρήση της βιβλιοθήκης *Angular Material*. Επιπλέον, έγινε έκδοση της εφαρμογής σε διακομιστή *Node JS* που επικοινωνεί με βάση δεδομένων *MySQL*. Τέλος, αναφέρονται οι μετρικές του συστήματος, οι οποίες εκτυπώθηκαν αυτόματα μέσω του προγράμματος *Visual Studio Code*, στον παρακάτω πίνακα (Πίνακας 1).

Γλώσσα	Αρχεία	Γραμμές Κώδικα	Σχόλια	Κενά	Σύνολο
JSON	2	259	6	7	272
TypeScript	54	2,729	140	335	3,204
HTML	23	1,364	80	90	1,534
SCSS	7	104	6	27	137
JavaScript	28	1,546	180	162	1,888

Πίνακας 1: Πίνακας Μετρικών συστήματος

5.2 Μοντέλο Ανάλυσης S.W.O.T.

Το μοντέλο ανάλυσης *S.W.O.T.* είναι εργαλείο στρατηγικού σχεδιασμού, στο οποίο αναλύονται τα Ισχυρά (*Strengths*) και Αδύναμα σημεία (*Weaknesses*) ενός συστήματος, καθώς επίσης οι Ευκαιρίες (*Opportunities*) και οι Απειλές (*Threats*) του. Στην ενότητα αυτή, αναλύεται η συγκεκριμένη διαδικτυακή εφαρμογή βάσει του εργαλείου *S.W.O.T.*

5.2.1 Δυνατά Σημεία – Strengths

- Ευκολία στη χρήση από μη εξειδικευμένους χρήστες
- Η διαδικτυακή πλατφόρμα μπορεί να εγκατασταθεί σε τοπικό δίκτυο αλλά και στον παγκόσμιο ιστό
- Ο τρόπος κατασκευής του *API* προσφέρει τη δυνατότητα επέκτασης και σε επιπλέον εφαρμογές.
- Οι διαχειριστές της εφαρμογής έχουν πλήρη έλεγχο των χρηστών που χρησιμοποιούν την εφαρμογή, των σεναρίων που έχουν πρόσβαση οι χρήστες και των ρομπότ που εκτελούν τη διαδικασία.
- Συνεχής ενημέρωση κατά την εντολή εκτέλεσης σεναρίου.
- Δυνατότητα επεξεργασίας όλων των οντοτήτων σε περίπτωση λανθασμένης καταχώρησης.
- Δεν υπάρχει κάποιο παρόμοιο πρόγραμμα στην αγορά που να εξυπηρετεί τη συγκεκριμένη ανάγκη.
- Ευκολία εγκατάστασης

5.2.2 Αδύναμα Σημεία – Weaknesses

Σημαντική αδυναμία του συστήματος, είναι ότι δεν έχουν πραγματοποιηθεί δοκιμές σε πραγματικές συνθήκες. Για αυτόν τον λόγο, δε γνωρίζουμε τι είδους προβλήματα θα προκύψουν στο *front-end* από περιπτώσεις χρήσης των χρηστών. Επιπλέον, στη βάση δεδομένων και στο *API* δεν έχουν γίνει δοκιμές για μεγάλο όγκο διαχείρισης δεδομένων.

5.2.3 Ευκαιρίες – Opportunities

Για τη συγκεκριμένη παροχή υπηρεσιών δεν υπάρχει ανταγωνισμός στην αγορά και, λόγω της πανδημίας του *COVID-19*, προσφέρει έναν ενναλακτικό τρόπο εκμάθησης στους μαθητές. Επιπλέον, αν κατασκευαστούν ορισμένες επεκτάσεις θα δινόταν η ευκαιρία ενσωμάτωσης του συστήματος σε γραμμές παραγωγής του ιδιωτικού τομέα.

5.2.4 Απειλές – Threats

Η μεγαλύτερη απειλή που αντιμετωπίζει το σύστημα είναι ο μεγάλος όγκος δεδομένων, από πλήθος χρηστών που χρησιμοποιούν ταυτόχρονα τη πλατφόρμα. Καθώς δεν έχουν πραγματοποιηθεί δοκιμές υπό πραγματικές συνθήκες. Επιπλέον, απειλή μπορεί να θεωρηθεί το *hacking*. Τα προσωπικά δεδομένα είναι πια μέρος της αγοράς και υπάρχουν επιτήδειοι που προσπαθούν να υποκλέψουν αυτές τις πληροφορίες ώστε να τις εκμεταλλευτούν για προσωπικό όφελος.

Το σύστημα έχει κατασκευαστεί σύμφωνα με την αρχιτεκτονική *MEAN* και για τον λόγο αυτό, έχουν ληφθεί μέτρα αντιμετώπισης των *hackers*. Ωστόσο, ο κλάδος της πληροφορικής εξελίσσεται ραγδαία, οπότε δεν μπορεί να εξασφαλισθεί η ακεραιότητα ασφαλείας του συστήματος. Για να αποφευχθούν προβλήματα ασφαλείας χρειάζεται να γίνονται συχνές ενημερώσεις στην πλατφόρμα.

5.3 Μελλοντικές Επεκτάσεις

Κάθε ιστοσελίδα/διαδικτυακή πλατφόρμα αναβαθμίζεται συνεχώς, ώστε να ενημερώνονται οι βιβλιοθήκες με τις τελευταίες εκδόσεις και να προστίθενται επιπλέον λειτουργίες σε αυτή. Με αυτόν τον τρόπο, οι εφαρμογές μπορούν να ακολουθούν την τεχνολογία της εποχής και να επιτυγχάνεται μεγαλύτερη ασφάλεια. Η συγκεκριμένη διαδικτυακή εφαρμογή έχει περιθώρια βελτίωσης και ο τρόπος σχεδίασης, με τον οποίο έχει υλοποιηθεί προσφέρει ευκολία σε αυτήν την περίπτωση.

- **State-Management**: Καθώς η εφαρμογή θεωρείται *one-page application*, τα δεδομένα χάνονται κάθε φορά που γίνεται επανάληψη φόρτωσης της σελίδας και ο χρήστης αναγκάζεται να εκτελεί τις ίδιες διαδικασίες για να επανέλθει στην προηγούμενη κατάσταση. Υλοποιώντας *state-management*, αυτή η ανάγκη αντιμετωπίζεται.
- **Αναγνώριση επιπλέον συστημάτων**: Με αυτόν τον τρόπο, το σύστημα θα μπορεί να παρέχει τις λειτουργίες του και σε περισσότερα ρομποτικά συστήματα ή μηχανισμούς αυτοματοποίησης.
- **Ενσωμάτωση τεχνολογίας WebRTC**: Προσθέτοντας δυνατότητες *WebRTC* στην παρούσα πλατφόρμα, μπορούν να δοθούν στον χρήστη επιπλέον λειτουργίες όπως αυτή της αναπαραγωγής οπτικοακουστικού υλικού από ενσωματωμένη κάμερα του ρομπότ σε πραγματικό χρόνο.

5.4 Συμπεράσματα

Μπορούν να προκύψουν ποικίλα συμπεράσματα από την παρούσα διπλωματική εργασία. Κυρίως αποδεικνύεται ο μεγάλος βαθμός χρησιμότητας που μπορεί να προσφέρει το σύστημα σε καθηγητές, οι οποίοι δεν κατέχουν την απαραίτητη τεχνογνωσία να διδάξουν τις βασικές λειτουργίες και τους μηχανισμούς ενός ρομπότ σε μαθητές της δευτεροβάθμιας εκπαίδευσης.

Για τη διευκόλυνση και ασφαλέστερη χρήση της πλατφόρμας, παρέχεται πληθώρα επιλογών στους διαχειριστές που αφορούν τη διαχείριση και τον έλεγχο των ρομπότ, των σεναρίων και των χρηστών.

Τέλος, αναλύθηκαν εκτενώς τα προτερήματα των τεχνολογιών που χρησιμοποιούνται από τη *MEAN* αρχιτεκτονική.

5.5 Σύνοψη 5ου Κεφαλαίου

Η παρούσα διπλωματική εργασία ολοκληρώνεται με το πέμπτο κεφάλαιο. Σε αυτό το κεφάλαιο γίνεται μία συνοπτική ανάλυση του περιεχομένου της διαδικτυακής πλατφόρμας μέσω ενός μοντέλου *SWOT*. Παρουσιάζονται επιγραμματικά τα πλεονεκτήματα και τα μειονεκτήματα που προσφέρει η υλοποίηση αυτής της εργασίας σε συνδυασμό με τις ευκαιρίες που προσφέρει αλλά και τις απειλές που μπορεί να δεχτεί. Επιπλέον, δίνονται κάποιες πιθανές μελλοντικές επεκτάσεις που θα ωφελήσουν την πλατφόρμα. Το κεφάλαιο αυτό ολοκληρώνεται με μία συμπερασματική αναφορά της διπλωματικής εργασίας.

Παράρτημα

Για να εγκατασταθεί η διαδικτυακή εφαρμογή και να επιτευχθεί σωστά η λειτουργία της θα πρέπει να τηρηθούν κάποιες προϋποθέσεις. Αρχικά κρίνεται απαραίτητη η εγκατάσταση κατάλληλων βιβλιοθηκών λογισμικού που δεν υπάρχουν, όπως η Node JS και MySQL server.

Παρακάτω αναφέρονται τα βήματα που απαιτούνται για την ορθή εγκατάσταση του συστήματος σε τοπικό δίκτυο:

1. Αντιγραφή του φακέλου EduRobot σε φάκελο της επιλογής σας.
2. Παραχώρηση δικαιωμάτων 755 στον κατάλογο και τα περιεχόμενα του.
3. Στον MySQL server δημιουργείστε τη βάση δεδομένων που επιθυμείτε να επικοινωνεί η σελίδα
4. Περιηγηθείτε στο αρχείο back-end/util/environment.js και τροποποιείστε τις παραμέτρους database, username, password, port και host. Σε περίπτωση που χρησιμοποιείται σύστημα Linux τροποποιείστε και τις παραμέτρους socketPath.
5. Για να συνδεθείτε με SMTP υπηρεσία της επιλογής σας επεξεργαστείτε τις παραμέτρους του object mail (from, host, port, user, pass, web_token_key).
6. Περιηγηθείτε χρησιμοποιώντας τερματική κονσόλα στο σημείο ./edurobot/back-end και εκτελέστε τις εντολές npm install και npm start. Αφήστε ανοιχτή τη κονσόλα
7. Περιηγηθείτε στο αρχείο ./front-end/src/app/shared/requestURL.ts και τροποποιείστε τη παράμετρο baseURL βάση των ρυθμίσεων που έγιναν στο back-end.
8. Περιηγηθείτε χρησιμοποιώντας τερματική κονσόλα στο σημείο ./edurobot/front-end και εκτελέστε τις εντολές npm install και npm start.
9. Όταν ολοκληρωθεί η επεξεργασία των αρχείων θα εντοπίσετε στο τέλος τη διεύθυνση URL που εμφανίζεται η σελίδα.
10. Χρησιμοποιώντας τα κατάλληλα username/password μπορείτε να έχετε πρόσβαση στις λειτουργίες της πλατφόρμας (**username**: admin / **password**: 1234567). Προτείνεται η αλλαγή του μετά την εγκατάσταση του συστήματος.

Βιβλιογραφία

[1] Hyundai Robotics H!-RMS, Ιστότοπος, http://www.hyundai-robotics.com/enghrms.html, Φεβρουάριος 2021

[2] Παγκόσμιος Ιστός, Ιστότοπος, http://repfiles.kallipos.gr/, Νοέμβριος 2020

[3] Προγραμματισμός Διαδικτύου, Ιστότοπος, https://repository.kallipos.gr/, Νοέμβριος 2020

[4] Αρχιτεκτονική MEAN Stack, Ιστότοπος, https://evincedev.com/blog/mean-stack-architecture/, Νοέμβριος 2020

[5] Node JS, Ιστότοπος, https://nodejs.org/api/, Νοέμβριος 2020

[6] Angular Framework, Ισότοπος, https://angular-university.io/, Νοέμβριος 2020

[7] Γλώσσες Προγραμματισμού, Ιστότοπος, https://library.uowm.gr/?lang=el, Νοέμβριος 2020

[8] Γλώσσα Προγραμματισμού SCSS, Ιστότοπος, https://sass-lang.com/documentation, Νοέμβριος 2020

[9] Γλώσσα Προγραμματισμού Typescript, Ιστότοπος, https://www.typescriptlang.org/docs/, Νοέμβριος 2020

[10] Angular Material, Ιστότοπος, https://material.angular.io/guides, Νοέμβριος 2020

[11] MySQL, Ιστότοπος, https://www.mysql.com/products/workbench/, Νοέμβριος 2020

[12] Visual Studio Code, Ιστότοπος, https://code.visualstudio.com/, Νοέμβριος 2020

[13] GitKraken, Ιστότοπος, https://www.gitkraken.com/learn/git/best-practices, Νοέμβριος 2020