ΧΑΡΟΚΟΠΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ

ΣΧΟΛΗ ΨΗΦΙΑΚΗΣ ΤΕΧΝΟΛΟΓΙΑΣ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΜΑΤΙΚΗΣ

Πτυχιακή εργασία

ΣΧΕΔΙΑΣΗ ΚΑΙ ΚΑΤΑΣΚΕΥΗ ΕΥΦΥΟΥΣ ΠΛΗΡΟΦΟΡΙΑΚΟΥ ΣΥΣΤΗΜΑΤΟΣ ΕΚΤΙΜΗΣΗΣ ΚΙΝΔΥΝΟΥ ΚΑΙ ΥΠΟΣΤΗΡΙΞΗΣ ΚΛΙΝΙΚΗΣ ΑΠΟΦΑΣΗΣ ΓΙΑ ΤΗΝ ΕΞΑΤΟΜΙΚΕΥΜΕΝΗ ΔΙΑΧΕΙΡΙΣΗ ΑΣΘΕΝΩΝ ΣΕ ΜΕΘ ΜΕ ΣΗΠΤΙΚΟ ΣΟΚ

Ιωάννης Μαυρομματάκης

Αθήνα, 14/06/2022

HAROKOPIO UNIVERSITY

DIGITAL TECHNOLOGY

INFORMATICS AND TELEMATICS

DESIGN AND CONSTRUCTION OF AN INTELLIGENT INFORMATION SYSTEM FOR RISK ASSESSMENT AND CLINICAL DECISION SUPPORT AIMING AT PERSONALIZED PATIENT ADMINISTRATION

Bachelor thesis

Ioannis Mavrommatakis

Athens, 14/06/2022

ΧΑΡΟΚΟΠΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ

ΣΧΟΛΗ ΨΗΦΙΑΚΗΣ ΤΕΧΝΟΛΟΓΙΑΣ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΜΑΤΙΚΗΣ

Τριμελής Εξεταστική Επιτροπή

Γεώργιος Δημητρακόπουλος
Αναπληρωτής Καθηγητής, Πληροφορική και Τηλεματική, Χαροκόπειο Πανεπιστήμιο

Χρήστος Μιχαλακέλης
Αναπληρωτής Καθηγητής, Πληροφορική και Τηλεματική, Χαροκόπειο Πανεπιστήμιο

Ηρακλής Βαρλάμης
Αναπληρωτής Καθηγητής, Πληροφορική και Τηλεματική, Χαροκόπειο Πανεπιστήμιο

Ο Ιωάννης Μαυρομματάκης

δηλώνω υπεύθυνα ότι:

1) Είμαι ο κάτοχος των πνευματικών δικαιωμάτων της πρωτότυπης αυτής εργασίας και από όσο γνωρίζω η εργασία μου δε συκοφαντεί πρόσωπα, ούτε προσβάλει τα πνευματικά δικαιώματα τρίτων.

2) Αποδέχομαι ότι η ΒΚΠ μπορεί, χωρίς να αλλάξει το περιεχόμενο της εργασίας μου, να τη διαθέσει σε ηλεκτρονική μορφή μέσα από τη ψηφιακή Βιβλιοθήκη της, να την αντιγράψει σε οποιοδήποτε μέσο ή/και σε οποιοδήποτε μορφότυπο καθώς και να κρατά περισσότερα από ένα αντίγραφα για λόγους συντήρησης και ασφάλειας.

3) Όπου υφίστανται δικαιώματα άλλων δημιουργών έχουν διασφαλιστεί όλες οι αναγκαίες άδειες χρήσης ενώ το αντίστοιχο υλικό είναι ευδιάκριτο στην υποβληθείσα εργασία.

Σελίδα για Αφιέρωση

~Στο Φάνη, στο Γιάννη και στο Δημήτρη, που με δίδαξαν το "χαμπουτζαλισμό".

~Στον Πέτρο, για την υπομονή και τη συμπαράσταση που μου επέδειξε τόσα χρόνια.

~Στη Χριστίνα, που δίχως τη συνεισφορά της, τώρα δε θα ήμουν εδώ για να πάρω πτυχίο.

~Στους γονείς μου, που χρωστάω τα πάντα.

Σελίδα για γνωμικό

"***Πάντες ἄνθρωποι τοῦ εἰδέναι ὀρέγονται φύσει.***"
~Αριστοτέλης

"***Η επιτυχία δεν είναι διαρκής, η αποτυχία δεν είναι ολέθρια. Το κουράγιο και η συνεχής προσπάθεια είναι που μετράνε.***"
~Ουΐνστον Τσώρτσιλ

"***Καλύτερα να είσαι πολεμιστής σε κήπο, παρά κηπουρός στο πεδίο της μάχης.***"
~Κινέζικο γνωμικό

ΕΥΧΑΡΙΣΤΙΕΣ

Ευχαριστώ ιδιαίτερα τους Ηλία Παναγιωτόπουλο και Γεώργιο Δημητρακόπουλο για τη στήριξη που μου παρείχαν στην εκπόνηση της εργασίας, καθώς και τον κοσμήτορα Θωμά Καμαλάκη που αποτελεί πάντα πρόθυμο μεσολαβητή στην επικοινωνία καθηγητών και φοιτητών.

ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

Περίληψη στα Ελληνικά

Στην εποχή μας, η ανάπτυξη της τεχνολογίας και ο όγκος των διαθέσιμων δεδομένων επεκτείνουν δραματικά τις δυνατότητες του ιατροφαρμακευτικού κλάδου και ενισχύουν την αντιμετώπιση των νοσημάτων εξασφαλίζοντας, έτσι, την αύξηση του προσδοκίμου επιβίωσης. Παράλληλα, στις μονάδες εντατικής θεραπείας (ΜΕΘ), όπου οι ασθενείς βρίσκονται σε κρίσιμη κατάσταση, ο φόρτος εργασίας εξακολουθεί να επιβαρύνει ιατρούς και νοσηλευτές και τα δεδομένα των ασθενών χρειάζονται ταχεία επεξεργασία για τη χορήγηση της κατάλληλης θεραπείας.

Εντούτοις, ένα ζήτημα το οποίο σχετίζεται με την επέκταση του ορίου ζωής και τις συνθήκες στις ΜΕΘ είναι τα σύνδρομα που σχετίζονται με τη σήψη. Η σήψη (μια απειλητική για την υγεία κατάσταση), παρά την εξέλιξη της βιοτεχνολογίας και της ιατρικής, συνεχίζει να απασχολεί την ανθρωπότητα, αφού συνιστά τη 10^{η} αιτία θανάτου παγκοσμίως με 700.000 θανάτους το χρόνο. Η σήψη τα τελευταία χρόνια αποτελεί συχνότερο φαινόμενο και ορισμένοι παράγοντες που επιδρούν σε αυτό είναι η γήρανση του πληθυσμού, η εξασφάλιση της επιβίωσης ασθενών με πολλαπλά χρόνια νοσήματα, οι χημειοθεραπείες και η επεμβατική ιατρική.

Μια λύση στο χώρο της ιατρικής που γνωρίζει αυξημένο ενδιαφέρον, είναι η χρήση συστημάτων υποστήριξης κλινικής απόφασης (CCMS), τα οποία παρέχουν ακριβέστερες προγνώσεις και μπορούν να επιβλέπουν την κατάσταση των ασθενών. Αντικείμενο, λοιπόν, της παρούσας πτυχιακής εργασίας είναι η ανάπτυξη ενός μικρής έκτασης και πλήρους στοίβας (full stack) πληροφοριακού συστήματος υποστήριξης κλινικής απόφασης, το οποίο θα διαχειρίζεται τα δεδομένα ασθενών που έχουν εισαχθεί σε ΜΕΘ με σήψη και με βάση αυτά, θα εξάγει συμπεράσματα για την πιθανότητα θνησιμότητας και θα προβάλει το επίπεδο ανάγκης χρήσης αγγειοσυσταλτικών (vasopressors) για την αντιμετώπιση της σήψης μέσω ενός εκπαιδευμένου νευρωνικού δικτύου.

Στην παρούσα αναφορά, αφού γίνονται σαφείς οι λόγοι ύπαρξης ενός τέτοιου συστήματος, παραθέτονται οι παραδοχές και η σχεδίαση των λειτουργιών σε μορφή σεναρίων χρήσης, αναλύονται οι τεχνολογίες που χρησιμοποιούνται, επεξηγείται ο κώδικας που αναπτύχθηκε, ενώ δεν παραλείπονται και οι δυσκολίες που προέκυψαν.

Λέξεις κλειδιά: [ευφυές σύστημα, μηχανική μάθηση, ιατρικά δεδομένα, διαχείριση γνώσης, ΜΕΘ]

Abstract ή Περίληψη στα Αγγλικά

Nowadays, the development of technology and the volume of available data dramatically expand the capabilities of the medical industry and enhance the treatment of diseases, thus ensuring an increase in life expectancy. At the same time, in intensive care units (ICUs), where patients are in critical condition, the workload continues to be borne by physicians and nurses and patient data need to be processed rapidly to provide appropriate treatment.

However, one issue related to life expectancy extension and conditions in ICUs is sepsis-related syndromes. Sepsis (a threat to health) despite the development of biotechnology and medicine, continues to concern humanity, as it is the 10th leading cause of death worldwide with 700,000 deaths per year. Sepsis has become more common in recent years and some of the factors that affect it are the aging of the population, ensuring the survival of patients with multiple chronic diseases, chemotherapy and invasive medicine.

One solution in the field of medicine that is of increasing interest is the use of clinical decision support systems (CCMS), which provide more accurate prognosis and can monitor the condition of patients. Therefore, object of this dissertation is the development of a small scale full-stack information clinical decision system, which will manage the data of patients admitted to ICU with sepsis and based on them, will extract conclusions about mortality probability and the levels of need for vasopressors' use through a trained neural network.

In the present report, after making the reasons for the existence of such a system clear, the assumptions and the design of the functions in the form of use-case scenarios are presented, the technologies used are analyzed, the code developed is explained, while the difficulties that arose are not omitted.

Keywords: [intelligent system, machine learning, medical data, knowledge management, ICU]

ΚΑΤΑΛΟΓΟΣ ΕΙΚΟΝΩΝ

ΚΑΤΑΛΟΓΟΣ ΠΙΝΑΚΩΝ

ΚΑΤΑΛΟΓΟΣ ΣΧΗΜΑΤΩΝ

ΣΥΝΤΟΜΟΓΡΑΦΙΕΣ/ΑΚΡΩΝΥΜΙΑ

CCMS	Cognitive Clinical Management Systems
ΜΕΘ	Μονάδες Εντατικής Θεραπείας
SOFA	Sequential Organ Failure Assessment
ΒΔ	Βάση Δεδομένων
ΣΔΒΔ	Σύστημα Διαχείρισης Βάσης Δεδομένων
HTML	Hyper Text Markup Language
CSS	Cascading Style Sheets
CRUD	Create Read Update Delete
UML	Unified Modeling Language
ΜΓΑ	Μήτρες Γενετικών Αλγορίθμων
ML	Machine Learning
ΤΝ	Τεχνητή Νοημοσύνη
ΤΝΔ	Τεχνητά Νευρωνικά Δίκτυα
ΜΤΣ	Μέσο Τετραγωνικό Σφάλμα
MVC	Model-View-Controller
DOM	Document Object Model
CSV	Comma Separated Values
JSON	JavaScript Notation Object
API	Application Programming Interface
REST	Representational State Transfer
ORM	Object Relational Mapping
ODM	Object Document Mapping
UI	User Interface
JWT	JSON Web Token

ΕΙΣΑΓΩΓΗ

Η πανδημία του κορωνοϊού ανέδειξε την αναγκαιότητα της ύπαρξης και της ορθής λειτουργίας των μονάδων εντατικής θεραπείας. Ένα συχνό πρόβλημα των ασθενών που νοσηλεύονται με λοιμώξεις, όπως ο κορωνοϊός (αν και συνήθως πρόκειται για βακτηριακές λοιμώξεις[1]), είναι η σήψη. Σύμφωνα με την Τρίτη Διεθνής Ομοφωνία Ορισμών της Σήψης (αποκαλούμενη Sepsis-3), τα όργανα του ασθενή παρουσιάζουν απειλητική για τη ζωή δυσλειτουργία έπειτα από αδυναμία του ανοσοποιητικού να αντισταθεί σε αυτές τις λοιμώξεις[23]. Εάν οι τοξίνες και οι μικροοργανισμοί απαντώνται μόνο στο αίμα, πρόκειται για σηψαιμία. Στη χειρότερη περίπτωση, που είναι το στάδιο του σηπτικού σοκ ή σηπτικής καταπληξίας· οι υποκείμενες διαταραχές της κυκλοφορίας και του κυτταρικού μεταβολισμού είναι τόσο προχωρημένες, ώστε να αυξάνουν τη θνητότητα σε σημαντικό βαθμό και η σήψη των οργάνων συνδυάζεται με κυκλοφοριακές, κυτταρικές και μεταβολικές επιπλοκές. Οι διαταραχές στην ομοιόσταση και στο μεταβολισμό οδηγούν σε σύνδρομο πολυοργανικής ανεπάρκειας (MODS) και τελικά στο θάνατο.

Για την αντιμετώπιση αυτής της κατάστασης, γίνεται άμεση χρήση αντιβιοτικών, αγγειοσυσταλτικών και κορτικοστεροειδών, ενώ δοκιμάζονται και άλλες θεραπείες[2]. Το σηπτικό σοκ συνιστά πρόβλημα δεκαετιών και επηρεάζει ακόμα και τις ανεπτυγμένες χώρες. Πράγματι, η θνησιμότητα του σηπτικού σοκ αγγίζει το 50%[3], ενώ υπάρχουν και μακροπρόθεσμες επιπλοκές, όπως εξασθένηση του ανοσοποιητικού, συχνές επαναμολύνσεις και επανεισαγωγές, υποβάθμιση της ποιότητας ζωής και μείωση του προσδόκιμου επιβίωσης[17].

Κρίσιμο στοιχείο στην αντιμετώπιση των επιπλοκών και στη διάσωση των ασθενών είναι η έγκαιρη ανίχνευση και χορήγηση θεραπειών[5]. Τα κριτήρια για την αναγνώριση της σήψης ανανεώνονται συνεχώς, ενώ τα τελευταία χρόνια, έχουν γίνει προσπάθειες τόσο για έγκαιρη αναγνώριση όσο και για πρόβλεψη της έκβασης της κατάστασης των ασθενών μέσω τεχνικών μηχανικής μάθησης[6][7]. Οι προσπάθειες αυτές έκαναν χρήση, συνήθως, μαθηματικών αλγορίθμων μηχανικής μάθησης με επίβλεψη, όπως μπεϊσιανά δίκτυα ή παλινδρομήσεις. Μια άλλη τεχνική για μηχανική μάθηση (με επίβλεψη) που έχει χρησιμοποιηθεί στις ΜΕΘ είναι και τα τεχνητά νευρωνικά δίκτυα [8][9][10].

Εντούτοις, οι έρευνες αυτές δεν εστιάζουν στην κατασκευή ενός οργανωμένου και ιεραρχικά δομημένου συστήματος που απευθύνεται σε ιατρικό προσωπικό για καθημερινή και ερευνητική χρήση. Το σύστημα που έχει αναπτυχθεί και αναλύεται στην εργασία αποθηκεύει και διαχειρίζεται δεδομένα ασθενών, ενώ μπορεί να εκπαιδεύεται με αντίστοιχες μεθόδους ώστε να πραγματοποιεί έγκαιρες προβλέψεις.

ΚΕΦ.1: Ένα σύστημα διαχείρισης γνώσης

1.1. Το πρόβλημα του σηπτικού σοκ

Τα τελευταία χρόνια παρατηρείται αύξηση των περιπτώσεων σήψης παγκοσμίως με αποτέλεσμα να αποτελεί την κυριότερη αιτία διακομιδής σε ΜΕΘ, την επικρατέστερη αιτία θανάτου στις ΜΕΘ, καθώς και τυπική κατάληξη ασθενών με λοιμώδη νοσήματα[11-22]. Δεν είναι γνωστό αν η πνευμονία από επιπλοκές του κορωνοϊού SARS-COV-2 στο διάστημα έξαρσης της πανδημίας έχει μεταβάλλει αυτά τα στατιστικά[25]. Τα συμπτώματα της σήψης μιμούνται αυτά ενός κρυολογήματος και περιλαμβάνουν πυρετό, ρίγη, γρήγορη αναπνοή, πόνους και εξανθήματα, ενώ στο στάδιο του σηπτικού σοκ, παρατηρούνται σύγχυση, δυσκολία στην αναπνοή, αραιή ούρηση και αποχρωματισμός των άκρων (κυάνωση).

Το σηπτικό σύνδρομο (ή σήψη μονολεκτικά) αποτελεί ένα από σημαντικότερα προβλήματα της παγκόσμιας υγείας. Κάθε χρόνο πλήττει πάνω από τριάντα εκατομμύρια (30.000.000) ανθρώπους παγκοσμίως και αποτελεί μία από τις κυριότερες αιτίες θανάτου στους βαρέως πάσχοντες ασθενείς, με πέντε εκατομμύρια τριακόσεις χιλιάδες (5.300.000) θανάτους ετησίως. Χαρακτηριστικά, στις Η.Π.Α, σύμφωνα με τον CDC, υπολογίζεται πως ο ετήσιος αριθμός των ενηλίκων που νοσηλεύεται με σηπτικό σύνδρομο ανέρχεται στο 1.5 – 1.7 εκατομμύρια, εκ των οποίων διακόσιες εβδομήντα χιλιάδες (270.000) καταλήγουν[24]. Το κόστος νοσηλείας υπολογίζεται ότι άγγιξε τα εξήντα εκατομμύρια (60.000.000,00$) δολάρια.

Επιπλέον, πρέπει να τονισθεί πως ακόμη και μετά από την επιτυχή αντιμετώπιση της σήψης, ο ασθενής μπορεί να αντιμετωπίσει μακροχρόνια προβλήματα υγείας, τόσο σωματικά, όσο και ψυχικά. Πάνω από το 50% των ασθενών που επιβίωσαν παρουσιάζουν μακροχρόνιες επιπτώσεις, όπως σωματικές και ψυχικές διαταραχές, επιπτώσεις στην ψυχολογία, εγκεφαλοπάθεια και αναπηρία. Πολλοί από αυτούς αδυνατούν να επιβιώσουν μόνοι τους και χρειάζονται μακροχρόνια νοσηλεία και φροντίδα[56]. Για τους παραπάνω λόγους, είναι εξαιρετικά σημαντικό να συνεχισθεί η έρευνα, ώστε να βρεθούν αποτελεσματικότεροι τρόποι έγκαιρης διάγνωσης και αποτελεσματικής θεραπείας.

Μέχρι στιγμής, δεν υπάρχει κάποιος «χρυσός κανόνας» για την πρόληψη και πρόβλεψη της σήψης και του σηπτικού σοκ. Η διάγνωση επιτυγχάνεται με συνδυασμό κλινικών χαρακτηριστικών και αποτελεσμάτων εργαστηριακών ελέγχων, όπως χαμηλή πίεση, πυρετός, δυσκολία στην αναπνοή κλπ. Ως κριτήριο διάγνωσης της σήψης χρησιμοποιείται η κλίμακα κλινικοεργαστηριακής αξιολόγησης SOFA (Sequential Organ Failure Assessment) Score. Ανοδική μεταβολή της τιμής αξιολόγησης του ασθενούς ≥ 2 ισοδυναμεί με διάγνωση οργανικής δυσλειτουργίας συμβατής με σήψη. Αν η αρχική τιμή SOFA (που εξαρτάται και από συν-νοσηρότητες του ασθενούς) δεν είναι γνωστή, τότε θεωρείται ίση με μηδέν. Επειδή η κλίμακα SOFA περιλαμβάνει την αξιολόγηση εργαστηριακών παραμέτρων, που δεν είναι πάντα εφικτό να είναι διαθέσιμες, θεσπίστηκε η απλούστερη κλινική κλίμακα q SOFA (quick SOFA). Αυτή απαιτεί 2 από τα εξής 3 κριτήρια :

- Συστολική Αρτηριακή Πίεση ≤ 100 mmHg (υπόταση)
- Αριθμό Αναπνοών > 22/min (ταχύπνεια)
- Διαταραχή επιπέδου συνείδησης

Η τιμή q SOFA ≥ 2 συσχετίστηκε με παρατεταμένη νοσηλεία στη ΜΕΘ και αυξημένη ενδονοσοκομειακή θνητότητα.

Η φύση ορισμένων ασθενειών, αλλά και οι διαφορές στα συμπτώματα των ασθενών δυσχεραίνουν την αξιοπιστία των αλγορίθμων πρόβλεψης. Επιπλέον, οι σύγχρονες τακτικές αντιμετώπισης, δηλαδή η χορήγηση αντιβιοτικών, αντιμυκητιασικών και η διατήρηση της ορθής λειτουργίας του αίματος και των ιστών είναι απαραίτητο να πραγματοποιούνται το συντομότερο δυνατό[26]. Άρα κρίνεται σκόπιμη η ανάπτυξη ενός αλγορίθμου που θα είναι ακριβέστερος ως προς τις ιδιότητες που οδηγούν έναν ασθενή σε σήψη και παράλληλα, με τη χρήση της τεχνολογίας, θα αξιοποιεί τα δεδομένα από περιπτώσεις ασθενών και θα εξάγει συμπεράσματα έγκαιρα πριν επιδεινωθεί η κατάσταση των ασθενών.

Τα σύγχρονα περιβάλλοντα των ΜΕΘ προσφέρουν ένα γόνιμο περιβάλλον, ώστε να αναπτυχθούν συστήματα κλινικής διαχείρισης, τα οποία θα υποβοηθούνται από την ανάπτυξη που γνωρίζουν οι δικτυακές υποδομές. Γύρω από αυτόν τον άξονα κινούνται τα Συστήματα Υποστήριξης Κλινικών Αποφάσεων (CCMSs), τα οποία μπορούν να προβλέψουν επικίνδυνες καταστάσεις καθώς και να βελτιώσουν την πορεία των ασθενών και τη φροντίδα που δέχονται. Τα συστήματα αυτά λειτουργούν με βάση την περισυλλογή δεδομένων από διάφορες πηγές, την επεξεργασία τους, την ενσωμάτωση γνώσεων και εμπειριών και τελικά τη λήψη των καταλληλότερων κλινικών αποφάσεων.

Όπως, προαναφέρθηκε και στην Εισαγωγή, έχουν πραγματοποιηθεί αρκετές προσπάθειες για την ανάπτυξη ενός «ευφυούς συστήματος» που σκοπό έχει την πρόληψη και πρόβλεψη της πορείας της σήψης. Μια από αυτές συνδύαζε τον αλγόριθμο του Kadri με δεδομένα από τη ΒΔ κλινικών δεδομένων MIMIC-III[7] με ελαφριά βελτίωση στην ακρίβεια πρόβλεψης. Μια πολύ ενδιαφέρουσα μελέτη, η οποία ήταν αρκετά κοντά στο τελικό προϊόν της εργασίας, ήταν ένα σύστημα κλινικής απόφασης που χρησιμοποιούσε πληθώρα τεχνικών μηχανικής μάθησης (μάλιστα δημιούργησαν την τεχνική με ΜΓΑ) και παρήχθη ένα σύστημα πλήρους στοίβας (full stack) σε ASP.NET[27].

1.2. Επιλογή μοντέλου μηχανικής μάθησης

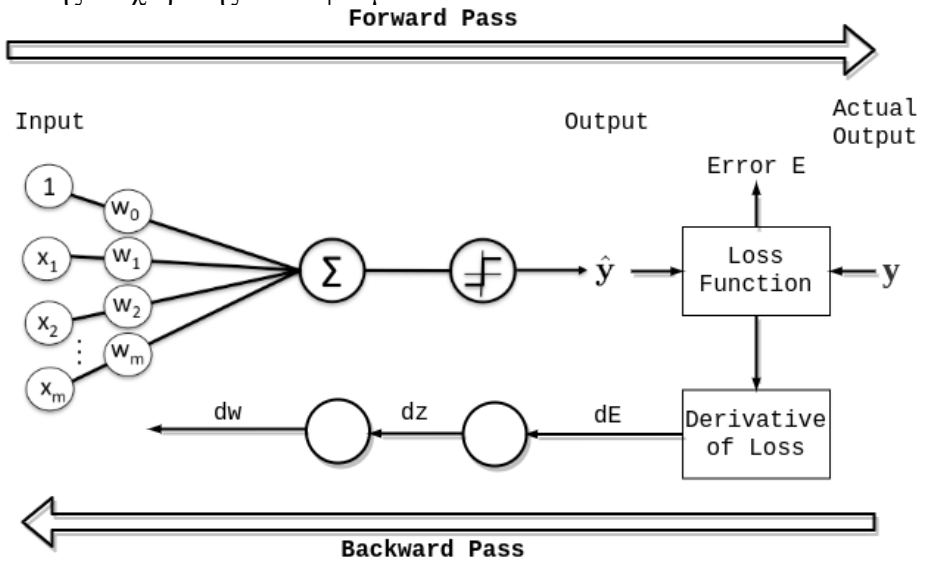
Βασικό στοιχείο ενός «νοήμονος συστήματος» το οποίο μπορεί να εξάγει συμπεράσματα είναι η ικανότητά του να μαθαίνει και να μπορεί να αξιολογεί την πληροφορία. Ένας τρόπος που μπορεί να επιτευχθεί αυτό είναι μέσω της μηχανικής μάθησης (machine learning, ML). Ως μηχανική μάθηση ορίζουμε τη διαδικασία κατασκευής ειδικών «έξυπνων» μεθόδων που εκπαιδεύουν μοντέλα με βάση ένα σύνολο δεδομένων[28]. Ασφαλώς υπάρχουν και άλλες τεχνικές εκπαίδευσης των μοντέλων τεχνητής νοημοσύνης, όπως είναι η βαθιά μάθηση (deep learning) και η μάθηση δίχως επίβλεψη (unsupervised learning), ωστόσο επειδή τα δεδομένα είναι ήδη κατηγοριοποιημένα και αναλυτικά (έτοιμα κλινικά δεδομένα ασθενών από πανεπιστήμιο), είναι προτιμότερη η εκπαίδευση του μοντέλου με τυπικές διαδικασίες μηχανικής μάθησης με επίβλεψη (supervised learning).

Μια αρκετά δημοφιλής επιλογή μάθησης με επίβλεψη και μηχανικής μάθησης γενικότερα, είναι η χρήση τεχνητών νευρωνικών δικτύων (ΤΝΔ). Τα νευρωνικά δίκτυα είναι συστήματα επεξεργασίας δεδομένων που αποτελούνται από έναν αριθμό τεχνητών νευρώνων (παρόμοια με έναν εγκέφαλο) που είναι δομικά οργανωμένος. Τα

χαρακτηριστικά των νευρωνικών δικτύων είναι η ικανότητά τους να μαθαίνουν μέσω παραδειγμάτων, δηλαδή εκπαιδεύονται με βάση έτοιμα δεδομένα που αναπαριστούν σχέσεις εισόδου-εξόδου, διαθέτουν μνήμη, είναι ανεκτικά σε σφάλματα και μπορούν να αναγνωρίσουν πρότυπα.

Η ανεκτικότητα στα σφάλματα και η αναγνώριση μοτίβων είναι μερικά από τα κυριότερα χαρακτηριστικά που βοήθησαν στην τελική επιλογή αυτής της μεθόδου. Ένα σημαντικό πλεονέκτημα σε σχέση με τις τεχνικές μάθησης με επίβλεψη, όπως οι παλινδρομήσεις ή η μάθηση Bayes είναι ότι δε γίνεται διαχωρισμός των δεδομένων σε κατηγορίες, κάτι που δεν προτιμάται σε περιπτώσεις ασθενών με πολλά κλινικά χαρακτηριστικά. Συνήθως στην εξαγωγή συμπερασμάτων πάνω σε ιατρικά δεδομένα προτιμώνται τα δέντρα απόφασης. Εντούτοις, το γεγονός ότι τα νευρωνικά δίκτυα συμπεριφέρονται ως αυτόνομες οντότητες διευκολύνει την ένταξή τους σε ένα ευρύτερο πληροφοριακό σύστημα, ενώ οι βιβλιοθήκες που παρέχουν την υλοποίησή των ΤΝΔ τυπικά περιλαμβάνουν έναν ικανοποιητικό αριθμό λειτουργιών και κατατοπιστική τεκμηρίωση. Ακόμη, ένα μοντέλο νευρωνικού δικτύου έχει το χαρακτηριστικό πλεονέκτημα ότι μπορεί να εκπαιδευτεί πολλές φορές μέσω «εποχών» (epochs) ή κύκλων εκπαίδευσης.

Ως εποχή ορίζουμε μια φάση εκπαίδευσης κατά την οποία εισάγονται μαζικά όλα τα δεδομένα εκπαίδευσης ή σε μικρότερα τμήματα που ονομάζονται «μίνι-δέσμες» (αν και γενικά το σύνολο που εφαρμόζεται ανά εποχή καλείται δέσμη), αθροίζεται για την κάθε είσοδο η μεταβολή στην τιμή των βαρών (τιμή που αλλοιώνει την είσοδο σε κάθε κόμβο του δικτύου) και στο τέλος αναπροσαρμόζονται τα βάρη. Ο αριθμός εποχών που απαιτούνται εξαρτάται από τον όγκο των δεδομένων και το ΜΤΣ (Μέσο τετραγωνικό σφάλμα, MSE) που προκύπτει κατά την εκπαίδευση, αν και συνήθως ξεπερνάει τις 100[29]. Η ύπαρξη κύκλων εκπαίδευσης διευκολύνει τη διαδικασία εκπαίδευσης διότι το σύνολο αληθινών δεδομένων από ασθενείς είναι σχετικά μικρό (~365), ωστόσο έχει το μειονέκτημα της αύξησης πολυπλοκότητας και σχετικά μεγαλύτερης κατανάλωσης μνήμης εξαιτίας του αριθμού των κόμβων, της συνολικής επεξεργασίας της εισόδου μέχρι την έξοδο, της μεταβολής των τιμών βάρους αλλά και της διαχείρισης των σφαλμάτων[30].



Εικόνα 1. *Οι εσωτερικές διαδικασίες ενός δικτύου ανά επανάληψη.*

Μια μορφή ΤΝΔ που χρησιμοποιείται αρκετά συχνά, είναι τα πρόσθιας τροφοδότησης (feedforward). Πρόκειται για τα απλούστερα δίκτυα, καθώς η πληροφορία που διέρχεται οδεύει μόνο στην έξοδο, οπότε είναι μονής κατεύθυνσης. Όπως διακρίνεται και στην *εικόνα 1*, ένα ΤΝΔ, ακόμα και στην πιο απλή μορφή του, εκτός από νευρώνες εισόδου και εξόδου, συχνά περιέχει και κρυφά επίπεδα ενδιάμεσα, τα οποία αυξάνουν τον αριθμό των κόμβων, με αποτέλεσμα το δίκτυο να εξάγει καλύτερα συμπεράσματα σε περιπτώσεις μεγάλου αριθμού εισόδου ή σε περιπτώσεις που δεν υπάρχει οργανωμένη διάταξη στα δεδομένα εισόδου (unordered categories). Ο αριθμός των κρυφών επιπέδων και των νευρώνων που περιέχουν ποικίλλει σε κάθε έργο και προσδιορίζεται κυρίως εμπειρικά βάσει του αριθμού σφαλμάτων που προκύπτουν. Ο λόγος χρήσης αυτού του τύπου δικτύου στην εφαρμογή έγκειται στην απλότητα και στην πληθώρα υλοποιήσεων που κυκλοφορούν ώστε να χρησιμοποιηθούν στην εφαρμογή, ενώ δεν απαιτείται ακρίβεια στην οργάνωση των δεδομένων εισόδου, όπου θα κρίνονταν επιτακτική η ανάγκη χρήσης ενός επαναλαμβανόμενου δικτύου (recurrent neural network)[31].

ΚΕΦ.2: Δημιουργία του συστήματος διαχείρισης ασθενών με σηπτικό σοκ

2.1. Ανάλυση και παραδοχές

Το σύστημα απευθύνεται, ως επί τω πλείστον, σε γιατρούς και νοσηλευτές ή και ερευνητές που επιθυμούν να αντλήσουν γνώση από το σύστημα για την πορεία περιστατικών που οδηγούν σε σηπτικό σοκ. Επομένως, οι χρήστες που θα μπορούν να συνδεθούν θα έχουν πρόσβαση σε όλες τις λειτουργίες του που αφορούν τη διαχείριση του λογαριασμού τους και τα στοιχεία των ασθενών. Οι κύριες λειτουργίες της εφαρμογής είναι η προσθήκη και διαχείριση ασθενών (με διαδικασίες CRUD), η επεξεργασία των στοιχείων του χρήστη και η ανάλυση των δεδομένων των ασθενών με γραφήματα και στατιστικές. Τα γραφήματα προκύπτουν με βάση τα δεδομένα που αποθηκεύονται στη ΒΔ ή από αποφάσεις του τμήματος που ασχολείται με τεχνητή νοημοσύνη. Η είσοδος στο σύστημα πραγματοποιείται με διαπιστευτήρια και αποτυχία εισόδου ή προσπέλαση απαγορευμένου πόρου θα συνοδεύεται με μήνυμα σφάλματος.

Έμπρακτος σκοπός του συστήματος είναι να διαχειρίζεται εύκολα και αποδοτικά τα δεδομένα των ασθενών και να απλοποιεί την αλληλεπίδραση με το χρήστη, ενώ η διαχείριση των λογαριασμών χρηστών είναι δευτερεύουσας σημασίας (αλλά εξακολουθεί να τις απασχολεί). Επίσης, όταν το σύστημα ολοκληρωθεί και είναι έτοιμο για χρήση, πρέπει να είναι συμβατό με νέα στοιχεία ασθενών και να έχει

δυνατότητα επέκτασης (scalability), ώστε να μπορούν οι υπεύθυνοι διαχείρισης και εγκατάστασής του να το συντηρήσουν ή να προσθέσουν νέες λειτουργίες.

Επομένως, θα είναι προσπελάσιμες από το χρήστη οι εξής λειτουργίες:

- Διαχείριση ασθενών
- Αναλυτική παρακολούθηση δεδομένων ασθενών
- Προσθήκη δεδομένων αληθινού χρόνου ασθενή που νοσηλεύεται
- Διαχείριση λογαριασμού

Επιπλέον, προχώρησα σε ορισμένες παραδοχές, έχοντας πάντα στο νου την επεκτασιμότητα, την απόδοση και την ευχρηστία και με σκοπό την τοποθέτηση ορίων κατά την ανάπτυξη της εφαρμογής. Αρχικά, ο χρήστης δεν μπορεί να διαγράψει οριστικά ούτε τους ασθενείς, αλλά ούτε και τα δεδομένα πραγματικού χρόνου που εισάγονται (Delete). Αυτό συμβαίνει ώστε να μη χάνονται τα δεδομένα ασθενών, να μπορούν να μελετηθούν μελλοντικά και ενδεχομένως να εξαχθούν συμπεράσματα, ενώ ενδέχεται να αποτελεί καλή στρατηγική να καταγράφεται η επίσκεψη των ασθενών σε ΜΕΘ (όπως προαναφέρθηκε, οι επανεισαγωγές σε μακροχρόνια κλίμακα αποτελούν συχνό φαινόμενο). Μπορεί, ωστόσο, να δηλώσει ότι ο ασθενής έχει επανέλθει και δε νοσηλεύεται στη ΜΕΘ. Έπειτα, αξίζει να σημειωθεί ότι για λόγους ανωνυμίας και αποδοτικότητας, οι χρήστες δε διαθέτουν εικόνες προφίλ για το λογαριασμό τους. Ασφαλώς, αυτή η λειτουργία μπορεί να προστεθεί ανά πάσα στιγμή.

Ενδεικτικά, το σύνολο των λειτουργιών μπορεί να συνοψιστεί μέσω του διαγράμματος περιπτώσεων χρήσης της UML (πρότυπο με τεχνικές δημιουργίας διαγραμμάτων για συστήματα) ως εξής[32]:

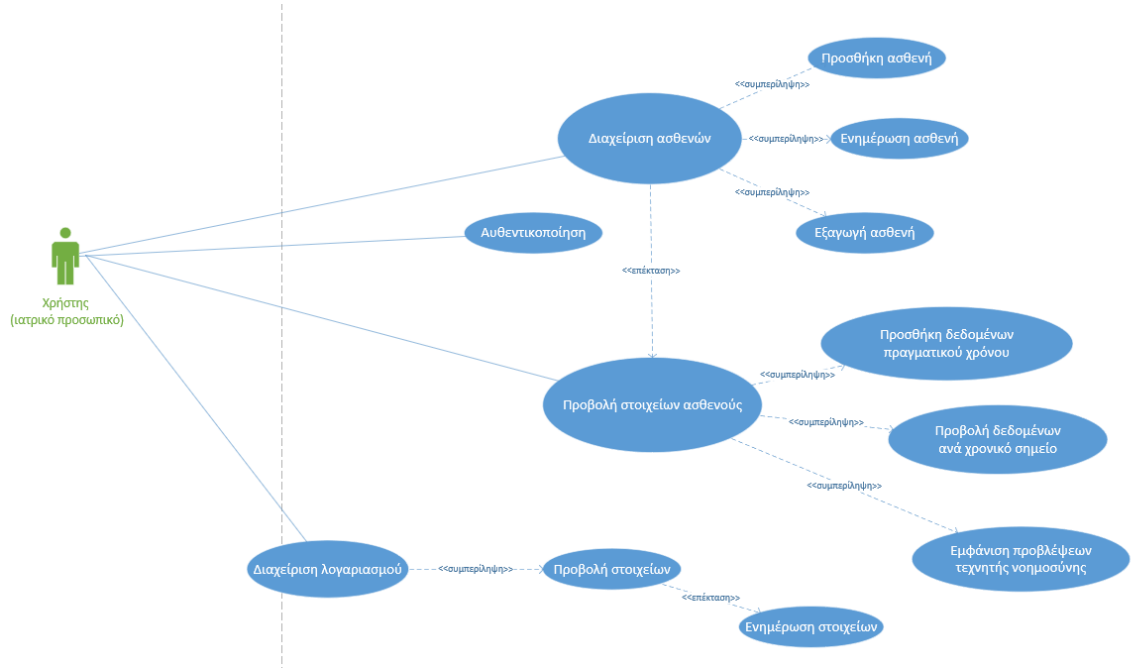


Figure 1. *Διάγραμμα των περιπτώσεων χρήσης του συστήματος*

2.2. Τα τμήματα και οι τεχνολογίες του συστήματος

Από τις παραδοχές, προκύπτει ότι η τελική μορφή του συστήματος με βάση τις τεχνολογίες που χρησιμοποιούνται είναι η εξής:

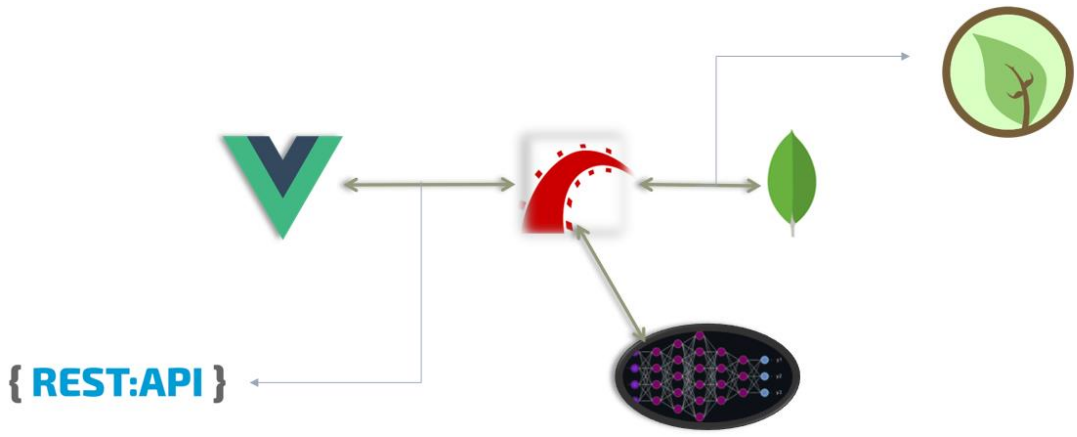


Figure 2. *Το γενικό σύνολο τεχνολογιών που εφαρμόζονται στο σύστημα*

Συνοπτικά, τα λογότυπα που εμφανίζονται στην εικόνα αντιπροσωπεύουν τις τεχνολογίες που χρησιμοποιήθηκαν και συμβολίζουν τα επιμέρους τμήματα του συστήματος, τα οποία είναι:

- **Το τμήμα του εξυπηρετητή**
- **Το τμήμα της διεπαφής χρήστη**
- **Το τμήμα τεχνητής νοημοσύνης**
- **Η βάση δεδομένων**
- **Οι διεπαφές επικοινωνίας**

Μια πιο αναλυτική απεικόνιση που θα περιλάμβανε τη συμβολή του χρήστη και θα αναδείκνυε το ρόλο της κάθε τεχνολογίας και πλαισίου θα ήταν ως εξής:

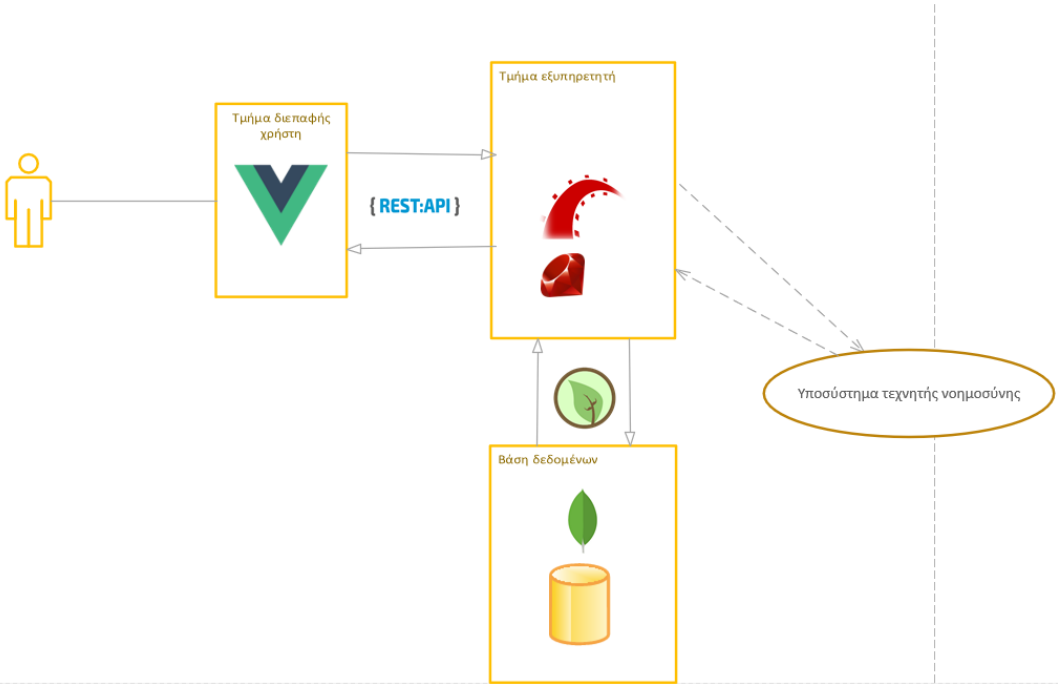


Figure 3. *Το αναλυτικό σύνολο τεχνολογιών που εφαρμόζονται στο σύστημα*

Θα αποσαφηνίσουμε τις λειτουργίες που επιτελούν τα τμήματα επακριβώς και έτσι θα γίνει κατανοητό ποιες τεχνολογίες συμβολίζονται από τα λογότυπα σε κάθε τμήμα.

2.2.1 Το τμήμα του εξυπηρετητή

Οι διακομιστές ιστοσελίδων αποτελούν την καρδιά του σύγχρονου Ιστού. Κύρια λειτουργία τους είναι η εξυπηρέτηση των αιτημάτων που προέρχονται από το χρήστη

και η ορθή απόκριση σε αυτά. Επιπλέον, οποιεσδήποτε λειτουργίες απαιτούνται από το σύστημα, εκτελούνται από το διακομιστή (τουλάχιστον σε βασικό επίπεδο). Συχνά αποκαλούνται το «πίσω μέρος» (back-end) μιας εφαρμογής ιστού. Η εφαρμογή που αναπτύχθηκε στην παρούσα εργασία δεν αποτελεί εξαίρεση. Έτσι, λοιπόν, χρειάστηκε ιδιαίτερη προσοχή στο πώς θα διαμορφωνόταν ο κύριος εξυπηρετητής που θα διαχειρίζεται τα αιτήματα του τμήματος διεπαφής χρήστη («μπροστινό μέρος» ή front-end αντίστοιχα), θα επικοινωνεί με τη ΒΔ και θα προσκομίζει τις βασικές πληροφορίες του συστήματος, θα θέτει σε λειτουργία, θα αρχικοποιεί και θα προβάλλει τις απαντήσεις του τμήματος τεχνητής νοημοσύνης, θα εκτελεί διάφορες τυπικές εργασίες, θα ανιχνεύει και διαχειρίζεται τα σφάλματα και θα προστατεύει το σύστημα από εξωτερικές επιθέσεις.

Για να μπορέσει ένα τέτοιο σύστημα να αποκτήσει υπόσταση, χρησιμοποιούνται τεχνολογίες πλαισίων λογισμικού (frameworks) που εξειδικεύονται στο να εκπληρώσουν το σκοπό αυτό. Στην πραγματικότητα, ένας εξυπηρετητής ιστού υπάρχει και δίχως τη χρήση εξειδικευμένων πλαισίων εφαρμογών. Στη σύγχρονη εποχή, όμως, θεωρούνται αναγκαία, καθώς οργανώνουν, αυτοματοποιούν και διευκολύνουν κατά πολύ τη λειτουργία των εξυπηρετητών που τελικά διαχειρίζονται τα εξωτερικά αιτήματα[33]. Αυτή η δυνατότητα προέρχεται από μια αρχή του προγραμματισμού που ονομάζεται «αντιστροφή ελέγχου» (Inversion Of Control, IoC) σύμφωνα με την οποία, ένα πρόγραμμα δεν εκτελείται με το συνήθη ακολουθιακό τρόπο, αλλά μεταφέροντας τον έλεγχο εκτέλεσης σε κάποια κεντρική διεργασία[34].

Υπάρχουν αμέτρητα πλαίσια λογισμικού για τη διαχείριση των εξυπηρετητών. Αυτό οφείλεται σε ένα βαθμό στο ότι υπάρχουν αντίστοιχα πολλές γλώσσες με μεγάλο μερίδιο αγοράς. Μπορούμε, συνεπώς, να θεωρήσουμε ότι για κάθε δημοφιλή γλώσσα αντιστοιχεί και ένα πλαίσιο λογισμικού. Επομένως, ένας καθοριστικός παράγοντας που ωθεί τον προγραμματιστή στην επιλογή συγκεκριμένου πλαισίου είναι η γλώσσα με την οποία έχει επιλέξει να ασχοληθεί ή έχει οικειότητα. Ασφαλώς υπάρχουν και άλλοι παράγοντες που επηρεάζουν μια τέτοια απόφαση, όπως η απόδοση, η συμβατότητα, οι παρεχόμενες λειτουργίες, το επίπεδο συνεργασίας με άλλες εφαρμογές κ.α.

Στην περίπτωση του συγκεκριμένου συστήματος, πρόκειται για μια εφαρμογή που διαχειρίζεται δεδομένα ασθενών με πολλαπλά χαρακτηριστικά, οφείλει να αποκρίνεται γρήγορα, να αλληλεπιδρά με λιγότερο τεχνολογικά καταρτισμένους χρήστες, να διαθέτει σταθερότητα και υποστήριξη, να είναι απλά οργανωμένο (και παράλληλα πλήρες), ασφαλές και ευέλικτο, να υπάρχει η δυνατότητα επέκτασής του, αλλά και να μπορεί να υποστηρίζει λειτουργίες τεχνητής νοημοσύνης. Ορισμένα υποψήφια πλαίσια που θα μπορούσαν να αντιστοιχούν σε αυτό το προφίλ είναι το Spring Boot, το Django (στην πραγματικότητα πρόκειται για MTV πλαίσιο), το NestJS, το Express.js και το ASP.NET. Αυτό που, όμως, επικράτησε στο τέλος ήταν η Ruby on Rails. Ας δούμε γιατί.

Η Ruby on Rails είναι πλαίσιο γραμμένο στην γλώσσα Ruby και στοχεύει στην ανάπτυξη εφαρμογών ιστού πλήρους στοίβας με έμφαση στον ευανάγνωστο και οργανωμένο κώδικα, μέσω του μοτίβου μοντέλου-όψης-ελεγκτή (MVC). Ένα πλαίσιο που ακολουθεί το μοτίβο MVC αναπαριστά και αποθηκεύει τα δεδομένα ως μοντέλα (Μ από το “Model”), παρουσιάζει τα αποτελέσματα στο χρήστη (V από το “View”) και διαχειρίζεται την είσοδο, τα μοντέλα και τις όψεις μέσω των ελεγκτών (C από το “Controller”). Τα πλαίσια που στηρίζονται κατεξοχήν σε αυτό το μοτίβο, όπως η Rails, το ASP.NET και το Spring MVC κυριαρχούν στη βιομηχανία ανάπτυξης εφαρμογών ιστοσελίδων για σχεδόν δυο δεκαετίες[35]. Ορισμένα από τα πλεονεκτήματα του MVC αρχιτεκτονικού σχεδιασμού είναι η αυτονομία των

τμημάτων μεταξύ τους, η διαχείριση των εξαρτήσεων και η ενσωμάτωση του κώδικα στην HTML στις όψεις.

Ο λόγοι που την επέλεξα ως πλαίσιο ανάπτυξης της εφαρμογής είναι η προσωπική εξοικείωση με την Ruby on Rails και την Ruby γενικότερα, η γρήγορη και εύκολη δημιουργία νέων έργων που πραγματοποιούν βασικές λειτουργίες χωρίς περαιτέρω ρυθμίσεις, η οργανωμένη δομή των καταλόγων του έργου, η πλούσια, επαρκής και κατανοητή τεκμηρίωση, αλλά και η ευελιξία που παρέχεται από το σύνολο των διαθέσιμων πακέτων (τα οποία ονομάζονται "gems"), γραμμένα σε Ruby. Βασικές λειτουργίες, όπως η διαχείριση της επιλεγμένης βάσης δεδομένων, η ασφάλεια του συστήματος, η χρήση υποσυστήματος τεχνητής νοημοσύνης, δε θα μπορούσαν να επιτευχθούν δίχως τα αναγκαία gems που μπορούν να αξιοποιηθούν από την Rails. Μια άλλη επιλογή που κινείται σε αυτά τα πλαίσια είναι το NestJS, το οποίο προσφέρει κι αυτό εργαλεία τεχνητής νοημοσύνης σε μια γλώσσα που αναπτύσσεται ραγδαία, την JavaScript. Θα δούμε, όμως, ποιοι ήταν οι λόγοι που απορρίφθηκε σε παρακάτω κεφάλαιο.

2.2.2 Το τμήμα διεπαφής με το χρήστη (Όψεις)

Όπως προαναφέρθηκε, τα πλαίσια MVC προσφέρουν, εκτός από διαχείριση των δεδομένων και της E/E, τη δυνατότητα για αναπαράσταση της πληροφορίας και του ελέγχου στο χρήστη. Η Rails προσφέρει ειδικά γι'αυτόν το σκοπό τα πρότυπα ERB ("Embedded RuBy") τα οποία είναι στατικές σελίδες HTML που ενσωματώνουν ειδικές εντολές για επικοινωνία με το Rails πλαίσιο. Τα πρότυπα δεν είναι αποκλειστική τακτική της Rails, καθώς το συναντάμε και σε άλλα πλαίσια, πέραν των προαναφερόμενων που ακολουθούν το MVC. Τα οφέλη των όψεων γραμμένα σε μορφή προτύπων περιλαμβάνουν την άμεση επικοινωνία και πλήρη αξιοποίηση των δυνατοτήτων του «πίσω μέρους», τον εμπλουτισμό της ήδη υπάρχουσας σύνταξης HTML, ενώ αρκετά συχνά, παρέχεται επαρκής τεκμηρίωση από τις ομάδες που συντηρούν τα εκάστοτε πλαίσια.

Τα τελευταία 10 χρόνια έχουν εμφανιστεί πλαίσια λογισμικού τα οποία δεν απευθύνονται στους διακομιστές των ιστοσελίδων, αλλά είναι οι ίδιες οι ιστοσελίδες. Πρόκειται για τα πλαίσια ανάπτυξης λογισμικού «μπροστινής όψης» τα οποία συμπεριφέρονται σαν διακομιστές (δέχονται αιτήματα σε συγκεκριμένη δικτυακή θύρα, οργανώνουν διαδρομές, κ.α.) και συνήθως βασίζονται στο Node.js, ένα περιβάλλον χρόνου εκτέλεσης (runtime) που επιτρέπει στη γλώσσα JavaScript να εκτελείται σε ένα λειτουργικό σύστημα όπως θα εκτελούνταν σε έναν περιηγητή (που είναι και το βασικό της περιβάλλον εκτέλεσης). Σε αντίθεση με τα πρότυπα, τα πλαίσια ιστοσελίδων δεν εμπλουτίζουν απλώς των κώδικα HTML ούτε εκτελούν εντολές μιας γλώσσας σε διαδικτυακό περιβάλλον, αλλά επιτρέπουν τη συγγραφή σελίδων με παραπάνω ετικέτες (παράλληλα με τη χρήση HTML) με αποτέλεσμα να θυμίζουν μια νέα γλώσσα σήμανσης (Markup Language). Επιπρόσθετα, αρκετά από αυτά τα πλαίσια περιλαμβάνουν πολύ περισσότερες δυνατότητες από ό,τι οι σελίδες με στατικά έγγραφα, κάτι που εισάγει μια ξεχωριστή λογική ανάπτυξης εφαρμογών.

Ένα χαρακτηριστικό που διευρύνει τις δυνατότητες των ιστοσελίδων, αλλά περιπλέκει και την ανάπτυξή τους, είναι το μοντέλο των ασύγχρονων «άγκιστρων κύκλου ζωής» (lifecycle hooks) το οποίο προκύπτει από την JavaScript και χωρίζει τη διαδικασία εκφόρτωσης του DOM (Μοντέλο Αντικειμένου Εγγράφου) σε φάσεις. Τα πλαίσια που πραγματοποιούν ασύγχρονες αλλαγές στο DOM της σελίδας,

ονομάζονται «αντιδραστικά» (reactive) και το προγραμματιστικό παράδειγμα «αντιδραστικός προγραμματισμός». Πλαίσια που φορτώνουν μόνο μια σελίδα DOM και την ανανεώνουν αντιδραστικά είναι τα React.js, Angular και Vue.js. Από αυτά επιλέχθηκε το Vue για την εργασία, αφού μπορεί να χρησιμοποιηθεί αποτελεσματικά τόσο σε έργα ανεξαρτήτως μεγέθους και σκοπού, ενώ διαθέτει ικανοποιητικό αριθμό λειτουργιών.

Το Vue.js αυτοπροσδιορίζεται ως το «προοδευτικό πλαίσιο» (progressive framework)[36], δηλαδή επιτρέπει τη διαρκή ανάπτυξη των έργων ασχέτως του μεγέθους τους, ακολουθώντας το μοντέλο της επεκτασιμότητας που έχει τονιστεί ήδη ότι απαιτείται για το συγκεκριμένο σύστημα. Εκτός από ασύγχρονη ανανέωση DOM, το Vue περιλαμβάνει δυνατότητα προσθήκης επαναχρησιμοποιούμενων συστατικών στοιχείων (components), μενού μεθόδων και μεταβλητών ώστε να χρησιμοποιηθούν στα στοιχεία αυτά, εντολές για διαχείριση των μεταβλητών, των μεθόδων και των γεγονότων σε κάθε σημείο της σελίδας, κεντρική κατασκευάστρια μέθοδο στην οποία προσθέτονται επιλογές, συστατικά στοιχεία και κλάσεις JavaScript που έχουν γραφτεί σε άλλα πηγαία αρχεία και χρησιμεύουν ως πρόσθετα. Σημαντικό κριτήριο επιλογής συνιστά το γεγονός ότι η σύνταξη του Vue είναι απλή και απαιτεί συγκριτικά λιγότερο χρόνο εκμάθησης από τα υπόλοιπα πλαίσια[37].

Είναι όμως το Vue και το κάθε πλαίσιο ιστοσελίδας καλύτερο από τις λύσεις των MVC; Στην πραγματικότητα αυτό αποτελεί περισσότερο ζήτημα προτίμησης, αν και για πιο μικρά έργα (και ιδιαίτερα αν πρόκειται για Rails), ίσως η επιλογή των προτύπων να είναι προτιμότερη. Στην περίπτωσή μας, μια εφαρμογή που διαχειρίζεται δεδομένα και αποβλέπει στη σταδιακή προσθήκη επιπλέον δυνατοτήτων επωφελείται περισσότερο από την ανάπτυξη ενός API. Θα δούμε παρακάτω πώς η αναπαράσταση των δεδομένων, εν τέλει επηρεάζει ολόκληρη τη δομή του έργου.

2.2.3 Το τμήμα τεχνητής νοημοσύνης

Το τμήμα τεχνητής νοημοσύνης αναλαμβάνει την αρχικοποίηση, την εκπαίδευση, τον υπολογισμό και την ανάλυση που πραγματοποιούνται στο «πίσω μέρος» του συστήματος. Στην πραγματικότητα, αυτό που χαρακτηρίζουμε ως «τμήμα», είναι ένα σύνολο από επιμέρους λειτουργίες που τις αναλαμβάνει μια κλάση στο πλαίσιο Rails σε συνεργασία με ορισμένες βιβλιοθήκες. Αρχικά, ο χρήστης που μεταβαίνει στη σελίδα για τα στατιστικά και τις προγνώσεις, αρχικοποιεί το σύστημα τεχνητής νοημοσύνης με μια κλήση στους ελεγκτές του συστήματος που είναι υπεύθυνοι για τη λειτουργία της τεχνητής νοημοσύνης. Η αρχικοποίηση περιλαμβάνει την ανάκληση ενός έτοιμου και εκπαιδευμένου ΤΝΔ αποθηκευμένο σε ένα αρχείο () και αν αυτό το αρχείο δεν υπάρχει, τότε εκπαιδεύεται και δημιουργείται ένα νέο ΤΝΔ. Η εκπαίδευση του ΤΝΔ γίνεται με βάση ένα αρχείο τύπου CSV (Comma Separated Values) που περιέχει τα αποθηκευμένα στοιχεία 364 ασθενών από το κέντρο υγείας και πανεπιστήμιο του Proctor στο Ιλινόις των ΗΠΑ[4]. Συνεπώς, μια βιβλιοθήκη που χρησιμοποιείται αφορά την ανάγνωση και εκφόρτωση δεδομένων από αρχεία .csv. Οι γραμμές και οι στήλες του αρχείου ομαδοποιούνται σε εισόδους και εξόδους, όπως προϋποθέτει η διαδικασία εκπαίδευσης με επίβλεψη ενός ΤΝΔ και στη συνέχεια, το στιγμιότυπο του εκπαιδευμένου δικτύου αποθηκεύεται σε ένα ειδικό αρχείο προς ανάγνωση για τη μελλοντική ανάκληση του δικτύου.

Ο όρος «στιγμιότυπο δικτύου» προκύπτει από το γεγονός ότι αρχικοποιείται και αποθηκεύεται ένα αντικείμενο στο περιβάλλον της Rails. Το αντικείμενο αυτό προέρχεται από τη βιβλιοθήκη του ΤΝΔ εντός της οποίας υλοποιούνται όλες αυτές οι

λειτουργίες. Πρόκειται για τη βιβλιοθήκη "ruby-fann" που περιέχει κλάσεις και μεθόδους για τη δημιουργία και διαχείριση τέτοιων δικτύων. Το "fann" υποδηλώνει ότι βασίζεται στη διαδεδομένη βιβλιοθήκη FANN, οπότε πρόκειται για δίκτυο πρόσθιας τροφοδότησης, όπως προαναφέρθηκε σε προηγούμενο κεφάλαιο. Το δίκτυο είναι υλοποιημένο κατά κύριο μέρος σε γλώσσα C, αφού και η FANN υλοποιείται σε αυτήν[38] και οι δημιουργοί του το προσάρμοσαν έτσι ώστε να λειτουργεί σε Ruby (wrapper)[39].

Το δίκτυο, είτε έχει δημιουργηθεί κατά τη φάση εκπαίδευσης, είτε έχει ανακληθεί από αρχείο, μπορεί να επιστρέψει τιμές εξόδου με βάση κάποιες τιμές εισόδου. Στην εφαρμογή, η διαχείριση της βιβλιοθήκης και του δικτύου πραγματοποιείται σε μια ξεχωριστή κλάση στον κατάλογο του έργου, η οποία λειτουργεί ως singleton (δημιουργείται ένα μόνο στιγμιότυπο κλάσης που υποστηρίζει έναν αριθμό λειτουργιών σε όλη την έκταση του εφαρμογής[40]). Σε επόμενο κεφάλαιο παρέχεται η ακριβής ανάλυση της λειτουργίας αυτού του αυτόνομου αντικειμένου που πρακτικά υλοποιεί το τμήμα τεχνητής νοημοσύνης.

2.2.4 Η βάση δεδομένων

Πλέον, οι βάσεις δεδομένων αποτελούν αναπόσπαστο τμήμα μιας τυπικής εφαρμογής ιστού και ιδιαίτερα εάν η εφαρμογή αυτή διατηρεί στοιχεία χρηστών. Η εφαρμογή μου περιλαμβάνει τόσο δεδομένα χρηστών, όσο και δεδομένα ασθενών, πραγματικού και μη χρόνου. Οι σχεσιακές ΒΔ συνιστούν μια συνηθισμένη επιλογή για την αποθήκευση των δεδομένων. Οι σχεσιακές βάσεις ακολουθούν το σχεσιακό μοντέλο που χαρακτηρίζεται από τις σχέσεις -πίνακες ή σύνολα από αντικείμενα που διαθέτουν παρόμοιες ιδιότητες- στις οποίες εναποθέτονται πλειάδες (γραμμές) σχετικών αντικειμένων (στήλες) και κάθε πλειάδα διαθέτει ένα μοναδικό αναγνωριστικό κλειδί (primary key)[41]. Το πρόβλημα με τις σχεσιακές ΒΔ είναι ότι παρουσιάζουν δυσκολίες όταν αυξάνεται ο όγκος των δεδομένων (λόγω αυξημένης πολυπλοκότητας), κάτι που μετατρέπεται σε εμπόδιο όταν η εφαρμογή αποσκοπεί στη διαρκή βελτίωση, ενώ δεν υπάρχει ευελιξία, διότι στηρίζονται κατά κύριο λόγο στον αρχικό σχεδιασμό των σχέσεων και των αναφορών μέσω των κλειδιών.

Μια εναλλακτική μορφή σχεδιασμού ΒΔ που προτάθηκε το 2009 ήταν η NoSQL[42], η οποία απέρριπτε το πλήρες οργανωμένο μοντέλο των σχέσεων και την SQL ως γλώσσα διαχείρισης αιτημάτων. Ως NoSQL χαρακτηρίζονται οι ΒΔ που ακολουθούν διαφορετικά σχεδιαστικά πρότυπα, αλλά αποκλείουν το σχεσιακό μοντέλο, όπως οι βάσεις κλειδιού-τιμής (key-value), γράφων (graph), ευρείας στήλης (wide column) και εγγράφων (document). Στην εφαρμογή χρησιμοποιήθηκε η ανοιχτού κώδικα ΒΔ εγγράφων MongoDB.

Το MongoDB είναι μια ΒΔ (και όχι ΣΔΒΔ, όπως λ.χ. η PostgreSQL) που αποθηκεύει τις οντότητες σε έγγραφα αντί για σχέσεις/πίνακες. Συγκεκριμένα, το MongoDB χωρίζει τη ΒΔ σε «συλλογές» (collections) που είναι κάτι αντίστοιχο με τους πίνακες στις σχεσιακές ΒΔ. Αντί για πλειάδες, κάθε συλλογή περιέχει έγγραφα σε μορφή παρόμοια με JSON. Η σημασιολογία JSON (JavaScript Notation Object) είναι σημαντική, καθώς χρησιμοποιείται ευρέως στο σύγχρονο Διαδίκτυο, λόγω της εξαιρετικά απλής σύνταξής της για την επισήμανση προγραμματιστικών αντικειμένων (πίνακες, αντικείμενα και μεταβλητές)[43]. Η σημασιολογία JSON είναι το μυστικό συστατικό πίσω από την επικοινωνία των τμημάτων, όπως θα εξηγήσουμε στο επόμενο υποκεφάλαιο. Το MongoDB αποθηκεύει τα έγγραφα σε μια συγγενική σημασιολογία, την BSON[44] για τον απλούστατο λόγο ότι η μορφή αυτών των

αρχείων είναι δυαδική, σε σύγκριση με τα αρχεία .json που είναι κωδικοποιημένες συμβολοσειρές, επομένως η διαχείρισή τους από μια ΒΔ και τα συστήματα που την χρησιμοποιούν είναι σαφώς ευκολότερη.

2.2.5 Οι διεπαφές επικοινωνίας

Μέχρι στιγμής, έχουν αναφερθεί δυο διαφορετικά πλαίσια που χρησιμοποιούνται στην ίδια εφαρμογή, ο τύπος της ΒΔ, καθώς και μια βιβλιοθήκη που περιέχει υλοποίηση ΤΝΔ. Το ερώτημα που προκύπτει είναι «Πώς επικοινωνούν μεταξύ τους;».

Προαναφέρθηκε η σημασιολογία JSON ως το βασικό στοιχείο της επικοινωνίας μεταξύ των τμημάτων. Όμως αυτό είναι το ένα κομμάτι. Το άλλο είναι η αποστολή και η λήψη των εγγράφων που διαθέτουν αυτήν την κωδικοποίηση. Έχουμε της αναφέρει ότι ένα σύστημα διαχείρισης «πίσω μέρους» είναι υπεύθυνο για τη διαχείριση των αιτημάτων και των σφαλμάτων. Επομένως, μια συνήθης περίπτωση διασύνδεσης πλαισίου διακομιστή με πλαισίου όψεων είναι να ανταλλάσσουν αιτήματα τα οποία ακολουθούν τη σημασιολογία JSON. Π.χ. ο χρήστης υποβάλει μια φόρμα για προσθήκη νέου ασθενή με τη βοήθεια του πλαισίου όψεων (Vue.js) και έπειτα αυτό καταχωρεί τα στοιχεία σε ένα αντικείμενο Request, το οποίο επιτρέπει την αποστολή αιτημάτων μέσω του HTTP πρωτοκόλλου σε μια διεύθυνση[45]. Τα στοιχεία ακολουθούν τη συντομογραφία JSON (κάτι σχετικά εύκολο, δεδομένου ότι το Vue χρησιμοποιεί τη γλώσσα JavaScript) και στη συνέχεια αποστέλλονται στο πλαίσιο διακομιστή Rails. Η Rails αποθηκεύει το αίτημα σε έναν πίνακα κατακερματισμών (hash array) που ονομάζεται “params”[56]. Οι κατακερματισμοί στις πρόσφατες εκδόσεις της Ruby, αν χρησιμοποιούν «σύμβολα» θυμίζουν μια συμβολοσειρά JSON (ή μάλλον BSON, αφού πρόκειται για αντικείμενα στη μνήμη):

```
:030 >   h1 = {name: "Canh"}
=> {:name=>"Canh"}
:031 > h2 = {name: "Canh"}
=> {:name=>"Canh"}
```

Εικόνα 2. *Δημιουργία κατακερματισμών στην Ruby*

Αφού το αίτημα του χρήστη επεξεργαστεί, το πλαίσιο της Rails οφείλει να επιστρέψει μια απόκριση σχετικά με την πορεία του αιτήματος (αν ήταν επιτυχές ή προέκυψε σφάλμα). Αυτή η απόκριση πρέπει να είναι αντίστοιχης μορφής με το αίτημα, οπότε αποστέλλεται σε μορφή JSON στο πλαίσιο όψης.

Μια εύλογη απορία θα ήταν το πώς επιτυγχάνεται η αποστολή των μηνυμάτων μεταξύ των δυο πλαισίων σε πρακτικό επίπεδο. Αυτό είναι κάτι που ορίζεται από τον προγραμματιστή και είναι αναγκαίο να ρυθμιστεί και στα δυο πλαίσια. Πρωτίστως, έχουμε αναφέρει ότι τα αντιδραστικά πλαίσια όψεων (της και τα πλαίσια διακομιστών) δημιουργούν υποδοχές (sockets) δικτύου μέσω της διεύθυνσης και της θύρας. Έτσι, τα αιτήματα μπορούν να εντοπίσουν το όνομα του πλαισίου στο δίκτυο. Όμως, εκτός από την κεντρική διεύθυνση (hostname), πρέπει κάθε αίτημα να πραγματοποιεί μια διαφορετική λειτουργία. Επομένως, υπάρχουν και διαδρομές ώστε το κάθε αίτημα να ξεχωρίζει από τα υπόλοιπα. Η διαδικασία αντιστοίχισης

διαδρομών με είδη αιτημάτων και η ανταλλαγή πληροφοριών μέσω πόρων (JSON ή XML) ονομάζεται REST (μεταφορά αναπαραστατικής κατάστασης) και οι διεπαφές επικοινωνίας, δηλαδή οι διαδρομές με το όνομα του εκάστοτε εξυπηρετητή, RESTful API[57]. Συνεπώς, υπάρχουν δυο αληθινοί εξυπηρετητές· αυτός που διαχειρίζεται το «πίσω μέρος» και αυτός που αναλαμβάνει την επικοινωνία με το χρήστη.

Ωστόσο, η αρχιτεκτονική REST (που είναι η πρωταρχική αρχιτεκτονική στο σύγχρονο Διαδίκτυο), αφορά την επικοινωνία σε δικτυακό επίπεδο. Η επικοινωνία με τη ΒΔ απαιτεί άλλου είδους διεπαφές. Έχουμε αναφερθεί σε πλαίσια ανάπτυξης λογισμικού για όψεις και διακομιστές. Μια ακόμη κατηγορία πλαισίων αφορά τη διαχείριση των ΒΔ και συχνά αποκαλούνται χαρτογράφοι (mappers). Έτσι, λοιπόν, έχουμε τα πλαίσια ORM (Object Relational Mapping)[46] για τις σχεσιακές ΒΔ και τα ODM (Object Document Mapping)[47] για τις ΒΔ βασισμένες στα έγγραφα. Και τα δυο είδη πλαισίων είναι υπεύθυνα για την προσθήκη αφαιρετικότητας στη διαχείριση των ΒΔ από πλευράς κώδικα, την παροχή και υποστήριξη ενός API που πραγματοποιεί «συναλλαγές» (transactions) με τη ΒΔ και την ολοκληρωτική δημιουργία και διαχείριση οντοτήτων εντός μιας ΒΔ.

Ένα παράδειγμα πλαισίου πλήρως υποστηριζόμενο και συμβατό με Rails, είναι το Mongoid, το οποίο είναι ODM και διευκολύνει δραματικά τη διαχείριση των αντικειμένων τόσο στην MongoDB βάση, όσο και στην ίδια την Rails. Εγκαθίσταται ως εξάρτηση gem όπως και η βιβλιοθήκη για το νευρωνικό δίκτυο. Θα εξηγήσουμε αναλυτικότερα πώς ακριβώς διευκολύνεται η πραγματοποίηση CRUD διαδικασιών για τη διαχείριση των δεδομένων στο επόμενο κεφάλαιο.

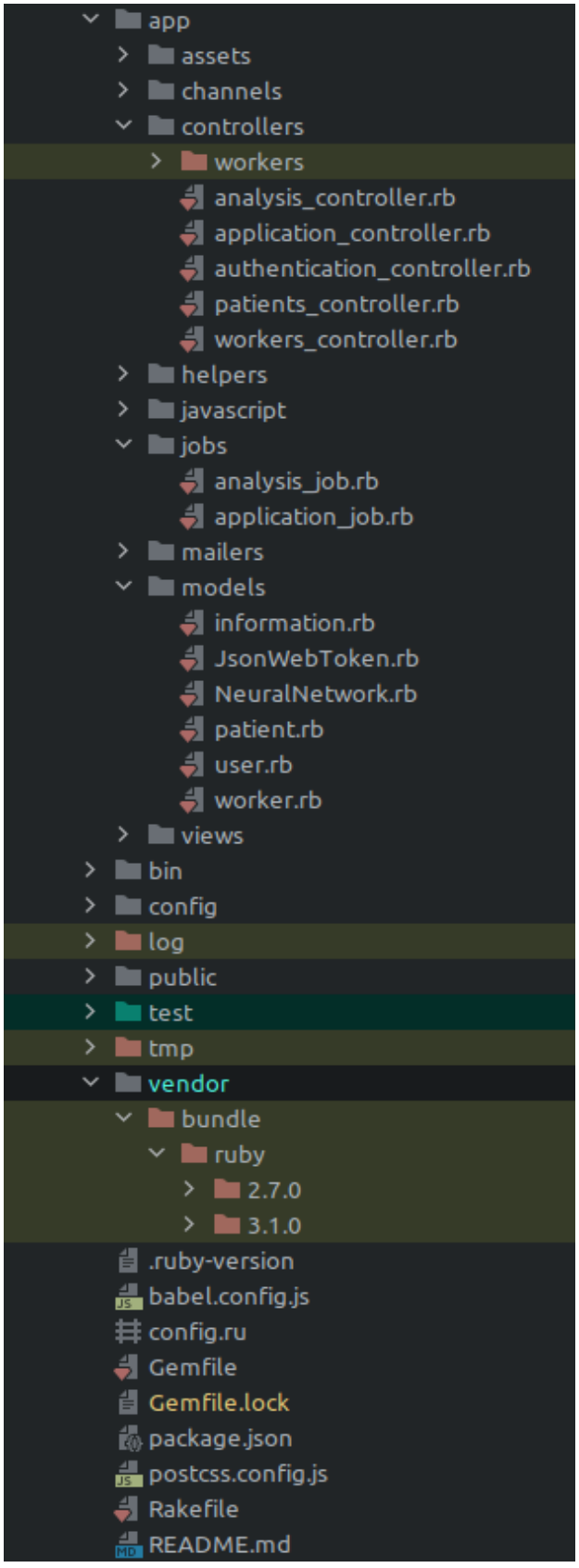
2.3. Διαδικασία ανάπτυξης

2.3.1 Οργάνωση της Rails

Εφόσον ολοκληρώθηκε η διεξοδική έρευνα και επιλέχθηκαν τα κατάλληλα εργαλεία, σειρά είχε η διασύνδεσή τους για την υλοποίηση του συστήματος. Εφόσον το υποσύστημα του εξυπηρετητή κρίνεται ως το βασικότερο, ξεκίνησα αναπτύσσοντας το «πίσω μέρος» της εφαρμογής μέσω της Ruby on Rails. Η δημιουργία ενός νέου έργου αποτελεί εύκολη υπόθεση στην Rails, αφού με την εντολή

rails new ICU-Knowledge-Manager -O

δημιουργείται ο κατάλογος “ICU-Knowledge-Manager” που περιέχει τα αναγκαία αρχεία για τη λειτουργία του πλαισίου και την ανάπτυξη της εφαρμογής. Η επιλογή -O στο τέλος της εντολής αποκλείει τη χρήση προεπιλεγμένης ΒΔ (στην Rails είναι η SQLite) ώστε να συνδυαστεί μετέπειτα με την MongoDB.



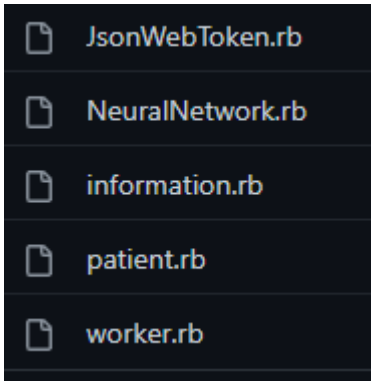
Εικόνα 3. *Η δομή του «πίσω μέρους» της εφαρμογής σε Rails*

Όπως φαίνεται και στην εικόνα, υπάρχουν φάκελοι που περιέχουν αρχεία ρυθμίσεων και αρχεία Ruby στον κατάλογο /app. Στον πίνακα αναγράφονται συνοπτικά οι κατάλογοι και η χρησιμότητά τους[48]:

Πίνακας 1: Βασικοί κατάλογοι του έργου για το πλαίσιο της Rails

/app	Κεντρικός κατάλογος του έργου που περιέχει τα στοιχεία που συγκροτούν το MVC.
/app/assets	Περιέχει τα στατικά αρχεία που χρειάζονται για την απεικόνιση των όψεων (CSS, JavaScript, SASS).
/app/controllers	Αποθήκευση όλων των ελεγκτών που συγκροτούν τη διεπαφή επικοινωνίας με το διακομιστή.
/app/helpers	Βοηθητικές συναρτήσεις για το MVC.
/app/jobs	Εργασίες που εκτελούνται στο παρασκήνιο ανά τακτό χρονικό διάστημα.
/app/mailers	Διαδικασίες σχετικές με τη διαχείριση ηλεκτρονικού ταχυδρομείου.
/app/models	Δεδομένα και μοντέλα που αποθηκεύονται στη ΒΔ.
/app/views	Περιέχονται τα πρότυπα που προβάλλονται στο χρήστη.
/bin	Εκτελέσιμα αρχεία που επιτρέπουν τη διαχείριση του πλαισίου της Rails.
/config	Διάφορα αρχεία ρυθμίσεων (επικοινωνία με ΒΔ, διαδρομές, διαχείριση πόρων πολλαπλής προέλευσης, κλπ.
/public	Περιέχει στατικά αρχεία και μεταγλωττισμένα στοιχεία.
/test	Αρχεία για δοκιμές που μπορούν να εφαρμοστούν στην εφαρμογή.
Gemfile	Το αρχείο με το οποίο ικανοποιούνται οι εξαρτήσεις του έργου και του πλαισίου και γι'αυτό κρίνεται απαραίτητο.
/vendor	Περιέχει τα gems που εγκαταστάθηκαν μέσω Gemfile για κάθε έκδοση της Ruby.
/tmp	Προσωρινά αρχεία και μνήμη cache.
/log	Αρχεία καταγραφών της εφαρμογής.

Ας εξετάσουμε, λοιπόν, τα αρχεία εντός του καταλόγου /app που παρέχουν τη λειτουργικότητα της εφαρμογής. Τα μοντέλα εντός του /app/models αντιστοιχούν στις οντότητες που αποθηκεύονται στη ΒΔ ή χρησιμοποιούνται βοηθητικά στην εκτέλεση ορισμένων λειτουργιών της εφαρμογής.



Εικόνα 4. *Τα περιεχόμενα του καταλόγου των μοντέλων*

Τα “worker”, “patient” και“information”, αντιστοιχούν στις οντότητες χρήστη, ασθενή και ιατρικά δεδομένα ασθενή αντίστοιχα και διαθέτουν τα εξής πεδία:

```ruby
class Worker
  include Mongoid::Document
  include ActiveModel::SecurePassword

  field :username, type: String, default: 'user'
  field :name, type: String
  field :surname, type: String
  field :age, type: Integer
  field :domain, type: String
  field :email, type: String, default: ""
  field :phone, type: String
  field :address, type: String
  field :password_digest
  has_secure_password
  # field :admin, type: Boolean, default: false
end
```

```ruby
class Patient
  include Mongoid::Document
  include Mongoid::Attributes::Dynamic

  field :AMKA, type: String
  field :name, type: String
  field :surname, type: String
  field :age, type: Integer
  field :gender, type: String
  field :description, type: String
  field :hospitalized, type: Boolean
end
```

```ruby
class Information
  include Mongoid::Document

  field :AMKA, type: String
  field :conditions, type: Array, default: []
  field :PaO2, type: Array, default: []
  field :FiO2, type: Array, default: []
  field :PLT, type: Array, default: []
  field :BIL, type: Array, default: []
  field :MAP, type: Array, default: []
  field :CR, type: Array, default: []
  field :UoP, type: Array, default: []
  field :PCT, type: BigDecimal
  field :ventilation, type: Boolean
  field :GCS, type: Integer
  field :location, type: Integer
  field :inflammation, type: Integer
  field :organism, type: Integer
  field :dates, type: Array, default: []
end
```

Εικόνα 5. *Τα πεδία των κλάσεων που αντιπροσωπεύουν τα μοντέλα της ΒΔ*

Παρατηρούμε ότι τα αρχεία αυτά είναι απλές κλάσεις της Ruby που όμως περιέχουν αρθρώματα από τη βιβλιοθήκη Mongoid (τα αρθρώματα ή modules παρέχουν ένα σύνολο από υλοποιημένες μεθόδους και εφαρμόζονται σε μια κλάση με την εντολή include[49]) η οποία προσφέρει έτοιμες μεθόδους για τη διαχείριση των οντοτήτων-εγγράφων στη ΒΔ. Τα πεδία αντιστοιχούν σε τύπους ακριβώς όπως ένα έγγραφο στην MongoDB και ορισμένα πεδία έχουν προεπιλεγμένες τιμές.

Το νευρωνικό δίκτυο υλοποιείται μέσω της κλάσης "NeuralNetwork" που είναι wrapper (κλάση που διαχειρίζεται υπάρχουσες λειτουργίες με πιο αφαιρετικό και εύκολο τρόπο για τον προγραμματιστή) του ruby-fann και υλοποιεί το τμήμα ΤΝ που προαναφέραμε.

```ruby
class NeuralNetwork
  include Singleton

  attr_accessor :instance

  def initialize
    super
    @path = 'app/assets/training'
    if File.exist? @path
      @instance = RubyFann::Standard.new(filename: @path)
      # @network.get_total_connections
    else
      @error = AnalysisJob.perform_now
      puts @error.inspect
      # return error if error
      @instance = RubyFann::Standard.new(filename: @path)
    end
  end

  def outcomes(data)
    puts @error.inspect
    # puts @error.is_a? Exception
    return @error if @error.is_a? Exception
    result = @instance.run data
    puts result.inspect
=begin
    result[1] = Float(result[1]).round
    result[2] = Float(result[2]).round
=end
    result
  end

end
```

Εικόνα 6. *Η κλάση για τη διαχείριση ΤΝ*

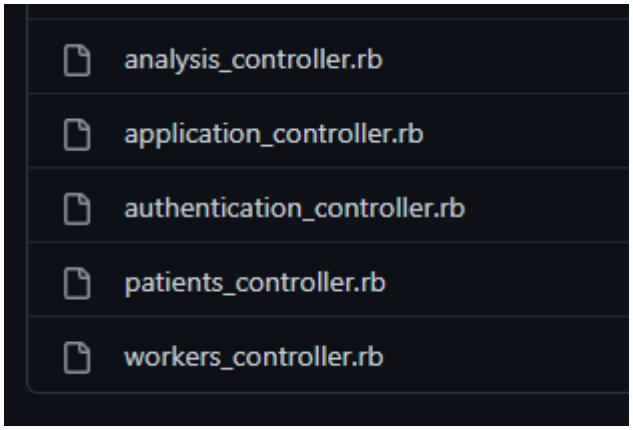
Η κλάση περιέχει δυο μεθόδους και μια μεταβλητή στιγμιοτύπου, ενώ ενσωματώνει το άρθρωμα “Singleton” που της επιτρέπει να καλείται μόνο μια φορά ώστε να προσφέρει τις απαιτούμενες υπηρεσίες. Οι υπηρεσίες αυτές προέρχονται από τις δυο αυτές μεθόδους, την “initialize” που είναι κατασκευάστρια συνάρτηση και αρχικοποιεί το αντικείμενο singleton και την “outcomes” που εξάγει τις πιθανότητες που υπολογίζει το ΤΝΔ με βάση τα δεδομένα εισόδου (παράμετρος “data”). Κατά τη δημιουργία του αντικειμένου, ελέγχεται εάν υπάρχει αρχείο με προρυθμισμένο ΤΝΔ και εάν δεν υπάρχει, δημιουργείται νέο από την εργασία “AnalysisJob” στο /app/jobs.

```
inputs = []
output = []

begin
  CSV.foreach('app/assets/real_clinical_data_septic_prediction.csv', headers: true) do |row|
    inputs.push [row[1], row[2], row[3], row[4], row[5], row[6], row[7], row[8], row[9], row[10], row[11], row[12],
                 row[13], row[14], row[15], row[16], row[17], row[18],
                 row[19], row[20], row[21], row[22], row[23], row[24], row[25]]
    #x_data.push( [row[0].to_f, row[1].to_f] )
    output.push [ row[29], row[30]]
  end
rescue Exception => error
  return error
end


set = (13...20).to_a
# set.push 2

inputs.map! do |x|
  i=0
  x.map! do |j|
    i+=1
    if set.include? i
      j.to_f
    else
      j.to_i
    end
  end
end

begin
  output.each do |i|
    i.map!(&:to_i)
  end
end
  output.each do |x|
    x.map!(&:to_f)
    x.map! do |i|
      i = i - 1.0
    end
  end
  # inputs.each do |i|
  #   puts i.inspect
  #   puts i[2].class
  # end
  # output.each { |i| puts i.inspect }

  train = RubyFann::TrainData.new(inputs: inputs, desired_outputs: output)
  fann = RubyFann::Standard.new(num_inputs: 25, hidden_neurons: [8], num_outputs: 2)
  fann.train_on_data(train, 50000, 100, 0.01) # 1000 max_epochs, 10 errors between reports and 0.1 desired MSE (mean-squared-error)
  fann.save('app/assets/training')
```

Εικόνα 7. *Η κλάση της εργασίας που εκπαιδεύει και αποθηκεύει το ΤΝΔ*

Η εργασία, εν ολίγοις, εντοπίζει ένα αρχείο τύπου CSV (εάν δεν υπάρχει, εκπέμπει σφάλμα), αποθηκεύει σε πίνακες τις εισόδους και τις εξόδους από τα δεδομένα των ασθενών και αφού τα μετατρέψει σε συμβατούς τύπους, δημιουργεί ένα ΤΝΔ και αφού το εκπαιδεύσει, το αποθηκεύει σε αρχείο. Έπειτα, από το αρχείο, φορτώνεται το νέο ΤΝΔ. Οι ρυθμίσεις του δικτύου αποτελούν τις βέλτιστες (με βάση εμπειρικές μετρήσεις), αφού τα ΜΤΣ ήταν τάξεως του 1% . Τέλος, το μοντέλο "JsonWebToken" περιέχει μεθόδους κωδικοποίησης και αποκωδικοποίησης για το JWT στο σύστημα αυθεντικοποίησης που θα εξηγήσω παρακάτω.

Οι ελεγκτές που βρίσκονται στο "/app/controllers" υλοποιούν το τμήμα της REST διεπαφής και είναι οι:



Εικόνα 8. *Τα περιεχόμενα του καταλόγου των ελεγκτών*

Ο "ApplicationController" παρέχεται από την Rails και συνήθως σε αυτόιν προσθέτονται μέθοδοι που εκτελούνται πριν από τους υπόλοιπους ελεγκτές. Οι "PatientController", "WorkerController" και "AnalysisController" περιέχουν τις μεθόδους που πραγματοποιούν όλες τις λειτουργίες της εφαρμογής.

```ruby
def index
  @patients = Patient.all
  render json: @patients.as_json
end

# GET /patients/1 or /patients/1.json
def show
  @patient = Patient.find_by AMKA: params[:id]
  render json: @patient, status: 200
end

# GET /patients/new
def new
  @patient = Patient.new
end

# GET /patients/1/edit
def edit
end

# POST /patients or /patients.json
def create
  @patient = Patient.new(patient_params)

  #respond_to do |format|
  if @patient.save
    # format.html { redirect_to @patient, notice: "Patient was successfully created." }
    render json: 'Ο ασθενής καταχωρήθηκε', status: 200
  else
     render json: @patient.errors, status: :unprocessable_entity
  end
  #end
end

# PATCH/PUT /patients/1 or /patients/1.json
def update
  #respond_to do |format|
  begin
    @patient = Patient.find params[:id]
    @patient.update patient_params
  rescue NoMethodError
    render json: 'No patient found', status: :unprocessable_entity
  else
    render json: 'Status updated', status: 200
  end
  # end
end

def delete
  begin
    @patient = Patient.find_by(id: params[:patient_id])
    @patient.update_attributes!(hospitalized: false)
  rescue NoMethodError
    render json: 'No patient found', status: :unprocessable_entity
```

```ruby
def index
  @workers = Worker.all
  render json: @workers.as_json
end

# GET /workers/1 or /workers/1.json
def show
  @worker = Worker.find params[:id]
  render json: @worker.as_json
end

# GET /workers/new
def new
  @worker = Worker.new
end

# GET /workers/1/edit
def edit
end

# POST /workers or /workers.json
def create
  @worker = Worker.new(worker_params)
  respond_to do |format|
    if @worker.save
      # format.html { redirect_to @worker, notice: "Worker was successfully created." }
      format.json { render :show, status: :created, location: @worker }
    else
      # format.html { render :new, status: :unprocessable_entity }
      format.json { render json: @worker.errors, status: :unprocessable_entity }
    end
  end
end

# PATCH/PUT /workers/1 or /workers/1.json
def update
  @worker = Worker.find params[:id]
  if @worker.update(worker_params)
    #format.html { redirect_to @worker, notice: "Worker was successfully updated." }
    render json: "Αποθηκεύτηκαν οι αλλαγές", status: 200
  else
    render json: "Αδύνατη η επεξεργασία", status: 400
  end
end

# DELETE /workers/1 or /workers/1.json
def destroy
  @worker.destroy
  respond_to do |format|
    format.html { redirect_to workers_url, notice: "Worker was successfully destroyed." }
    format.json { head :no_content }
  end
end
```

```ruby
def show
  @data = Information.find_by AMKA: params[:id]
  render json: @data, status: 200
end

def create
  @data = Information.create! first_params
  @data.add_to_set dates: Time.now.strftime("%Y-%m-%dT%H:%M:%S")
  puts @data.inspect
  render json: 'Ξεκίνησε καταγραφή δεδομένων', status: 200
end

def update
    @data = Information.find_by AMKA: params[:id]

    @data.add_to_set PaO2: data_params[:PaO2] unless data_params[:PaO2].nil?
    @data.add_to_set FiO2: data_params[:FiO2] unless data_params[:FiO2].nil?
    @data.add_to_set PLT: data_params[:PLT] unless data_params[:PLT].nil?
    @data.add_to_set BIL: data_params[:BIL] unless data_params[:BIL].nil?
    #@data.add_to_set GCS: data_params[:GCS] unless data_params[:GCS].nil?
    @data.add_to_set MAP: data_params[:MAP] unless data_params[:MAP].nil?
    @data.add_to_set CR: data_params[:CR] unless data_params[:CR].nil?
    @data.add_to_set UoP: data_params[:UoP] unless data_params[:UoP].nil?
    @data.add_to_set dates: Time.now.strftime("%Y-%m-%dT%H:%M:%S")
    @data.update PCT: data_params[:PCT] unless data_params[:PCT].nil?

    render json: 'Τα δεδομένα ανανεώθηκαν', status: 200
end

def calculate
  # @network = NeuralNetwork.instance
  puts Network.inspect
  @data = Information.find_by AMKA: params[:analysis_id]
  @patient = Patient.find_by AMKA: params[:analysis_id]
  input = data_filter @data, @patient
  puts input.inspect
  output = Network.outcomes input
  if output.is_a? Exception
    render json: "Σφάλμα συστήματος τεχνητής νοημοσύνης: #{output}", status: 500
  else
    prediction = filter_results output
    render json: prediction, status: 200
  end
end
```

Εικόνα 9.α. *Οι μέθοδοι εντός των κλάσεων που υλοποιούν τους ελεγκτές*

Οι λειτουργίες της εφαρμογής, λοιπόν, είναι ως επί τω πλείστον CRUD και παρέχουν στο χρήστη τη δυνατότητα, μέσω των όψεων του Vue που επικοινωνούν με το REST API, δηλαδή τους ελεγκτές, να διαχειριστούν τα δεδομένα με αφαιρετικό τρόπο. Τα δεδομένα που πρόκειται να προστεθούν στη ΒΔ, «φιλτράρονται» από μεθόδους-φίλτρα και από αυτές τις μεθόδους οι κύριες μέθοδοι που επιτελούν μια διαδικασία αντλούν τα δεδομένα με τη λογική κλειδιού-τιμής.

```
  # Only allow a list of trusted parameters through.
def patient_params
  params.require(:patient).permit(:AMKA, :name, :surname, :age, :description, :hospitalized, :gender)
end

def data_params
  params.permit( :PaO2, :FiO2, :PLT, :BIL, :MAP , :CR , :UoP)
end

def first_params
 params.permit( :AMKA, :ventilation, :location, :inflammation, :organism, :PCT, { :conditions => [] }, :GCS)
end

 # Only allow a list of trusted parameters through.
 def worker_params
   params.permit(:username, :password, :name, :surname, :age, :email, :domain)
 end
```

Εικόνα 10. *Οι μέθοδοι που επιλέγουν τα επιτρεπόμενα πεδία του αιτήματος JSON*

Στη συνέχεια, αυτές οι τιμές που λαμβάνονται από τις μεθόδους-φίλτρα ελέγχονται για την περίπτωση που είναι κενές (τιμή nil). Εάν δεν είναι, τότε προσθέτονται σε ένα αντικείμενο που έχει δημιουργηθεί (ή ανασυρθεί από τη ΒΔ, αν πρόκειται για ενημέρωση στοιχείων) και αφού προστεθούν στο αντικείμενο, εκείνο είναι έτοιμο για εγγραφή στη ΒΔ.

Ίσως να προσέξατε ότι κάποια πεδία στα μοντέλα είναι πίνακες. Αυτό επιτρέπει την καταγραφή ιστορικού για τις τιμές πραγματικού χρόνου. Για παράδειγμα, τιμές όπως η αρτηριακή πίεση αλλάζουν διαρκώς και θα ήταν χρήσιμο στο χρήστη να παρακολουθεί τις τιμές για κάθε χρόνο. Η καταγραφή του χρόνου είναι της μορφής «Χρονιά-μήνας-ημέρα Ώρα:λεπτά:δευτερόλεπτα» και λαμβάνει χώρα σε δυο μεθόδους του AnalysisController: Τις create και update.

```ruby
def create
  @data = Information.create! first_params
  @data.add_to_set dates: Time.now.strftime("%Y-%m-%d %H:%M:%S")
  puts @data.inspect
  render json: 'Ξεκίνησε καταγραφή δεδομένων', status: 200
end

def update
    @data = Information.find_by AMKA: params[:id]

    @data.add_to_set PaO2: data_params[:PaO2] unless data_params[:PaO2].nil?
    @data.add_to_set FiO2: data_params[:FiO2] unless data_params[:FiO2].nil?
    @data.add_to_set PLT: data_params[:PLT] unless data_params[:PLT].nil?
    @data.add_to_set BIL: data_params[:BIL] unless data_params[:BIL].nil?
    #@data.add_to_set GCS: data_params[:GCS] unless data_params[:GCS].nil?
    @data.add_to_set MAP: data_params[:MAP] unless data_params[:MAP].nil?
    @data.add_to_set CR: data_params[:CR] unless data_params[:CR].nil?
    @data.add_to_set UoP: data_params[:UoP] unless data_params[:UoP].nil?
    @data.add_to_set dates: Time.now.strftime("%Y-%m-%d %H:%M:%S")
    @data.update PCT: data_params[:PCT] unless data_params[:PCT].nil?

    render json: 'Τα δεδομένα ανανεώθηκαν', status: 200
end
```

Εικόνα 9.β. *Οι μέθοδοι δημιουργίας και ανανέωσης δεδομένων ασθενή*

Το ΤΝΔ, ωστόσο, δε λαμβάνει τις προηγούμενες τιμές που είναι αποθηκευμένες στον πίνακα, διότι αυτό θα δυσκόλευε τη διατήρηση της αλληλεξάρτησης εισόδου και εξόδου με βάση τα δεδομένα εκπαίδευσης.

Υπάρχουν και κάποια επιπλέον φίλτρα, τα οποία αφορούν το wrapper νευρωνικού δικτύου που αναφέρθηκε και οργανώνουν τα δεδομένα ώστε η “outcomes”μέθοδος να μπορεί να τα αναγνώσει ή προτού σταλθούν στο Vue μέσω του API:

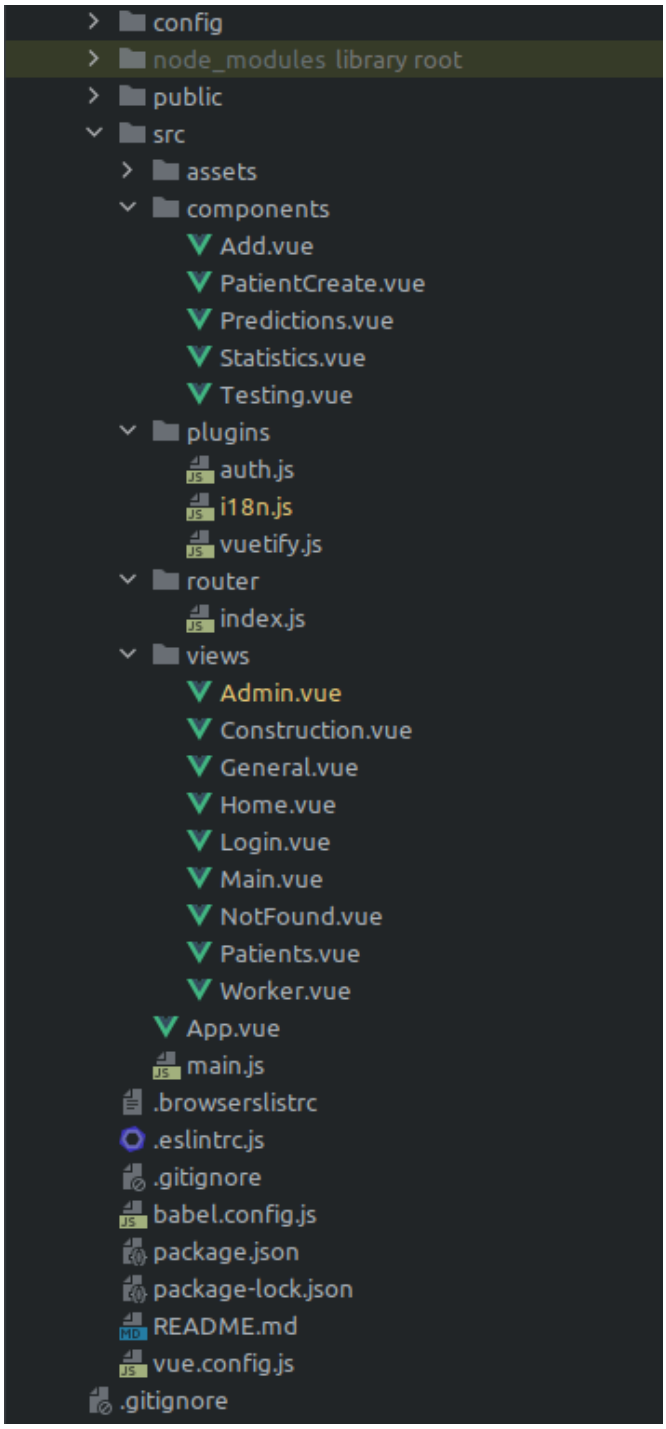
```ruby
def data_filter(data, patient)
  array = []
  if patient.gender == 'Αντρας'
    gender = 1
  else
    gender = 2
  end
  if data.ventilation
    ventilation = 1
  else
    ventilation = 2
  end
  conditions = conditions_filter(data.conditions)
  array.push patient.age.to_i, gender, data.PCT.to_f, conditions[0], conditions[1], conditions[2], conditions[3], conditions[4], conditions[5], conditions[6],
             conditions[7], conditions[8], conditions[9], data.PaO2[-1].to_f, data.FiO2[-1].to_f, data.PLT[-1].to_f, data.BIL[-1].to_f, data.GCS, data.MAP[-1].to_f,
             data.CR[-1].to_f, data.UoP[-1].to_i,   ventilation,  data.location.to_i, data.inflammation.to_i, data.organism.to_i

end
#.each_with_index {|value, i| array[i+3] = value }
def conditions_filter(conditions)
  list = ['Διαβήτης', 'Υπέρταση', 'Χρόνια αποφρακτική πνευμονοπάθεια', 'Στεφανιαία νόσος', 'Συμφορητική καρδιακή ανεπάρκεια', 'Χρόνια νεφρική νόσος με διάλυση',
      'Χρόνια νεφρική νόσος χωρίς διάλυση', 'Κολπική μαρμαρυγή', 'Πνευμονικό εμβολή', 'Καρκίνος']
  conditions = list.filter_map do |condition|
    if conditions.include? condition
      condition = 1
    else
      condition = 2
    end
  end
  puts conditions.inspect
  conditions
end

def filter_results(results)
  results[0] = 1.0 - results[0]
  results.map! do |x|
    x = x * 100
    x = x.round(2)
  end
end
```

Εικόνα 11. *Μέθοδοι που επεξεργάζονται τα δεδομένα ΤΝ*

Όσον αφορά τους καταλόγους “/app/views” και “/app/assets”, αυτοί δεν επιτελούν κάποια λειτουργία, διότι το έργο τους το αναλαμβάνει το Vue.js πλαίσιο.

2.3.2 Οργάνωση και τεχνολογίες του Vue.js

Η δομή ενός έργου σε Vue.js είναι εμφανώς διαφορετική σε σχέση με την οργάνωση της Rails:



Εικόνα 12. *Η δομή του «μπροστινού μέρους» της εφαρμογής σε Vue.js*

Υπάρχουν κατάλογοι με αρχεία ρυθμίσεων (“/config”), στατικά στοιχεία (“/src/assets”) και κατάλογοι με εξαρτήσεις (“/node_modules”), ενώ οι εξαρτήσεις καθορίζονται από το package.json, όπως θα γινόταν και με ένα Gemfile. Ωστόσο, το Vue.js δεν είναι ένα MVC πλαίσιο, οπότε δεν υπάρχουν ελεγκτές και μοντέλα. Ο κατάλογος “/routes” περιέχει ένα αρχείο “index.js” το οποίο περιέχει τις διαδρομές για μετάβαση στις τοποθεσίες της ιστοσελίδας (παρόμοια λογική με το REST API). Υπάρχει, όμως, ο κατάλογος “/components”. Η λογική του Vue.js έχουμε αναφέρει ότι στηρίζεται στα επαναχρησιμοποιήσιμα στοιχεία (components) τα οποία αποθηκεύονται σε αυτόν τον κατάλογο. Υπάρχουν και κάποιοι επιπλέον κατάλογοι που τους έχω δημιουργήσει, οι “/src/plugins” και “/src/views” που περιέχουν τα πρόσθετα που χρησιμοποιούνται εντός της εφαρμογής και μεγάλα στοιχεία που απεικονίζουν ολόκληρο το DOM.

Παρατηρούμε δυο αρχεία τα οποία βρίσκονται εκτός του καταλόγου των συστατικών στοιχείων και είναι τα “App.vue” και “main.js” που αρχικοποιούν την εφαρμογή, προσθέτουν δυνατότητες και ρυθμίσεις και διαχειρίζονται τα συστατικά στοιχεία με ιεραρχική οργάνωση. Η δομή ενός στοιχείου (αρχείου .vue) ορίζεται ως

- Πρότυπο (template)
- Σενάριο εκτέλεσης (script)
- Εμφάνιση (style)

```
<template>
  <v-app>
    <v-card
    dark
    tile
  >
    <v-toolbar height="70px" dense>
      <v-toolbar-title><v-btn plain text to="/" color="white"> ICU-Manager</v-btn></v-toolbar-title>

      <v-spacer></v-spacer>
      <v-menu v-if="isAuthenticated" open-on-hover offset-y transition="slide-y-transition" bottom>
        <template v-slot:activator="{ on }">
          <v-btn dark text v-on="on">
            Μενου
          </v-btn>
        </template>
        <v-list>
          <v-list-item to="/">
            <v-icon>mdi-home</v-icon>
            <v-list-item-title>Αρχική</v-list-item-title>
          </v-list-item>
          <v-list-item to="/patients">
            <v-icon>mdi-bed</v-icon>

            <v-list-item-title>Ασθενείς</v-list-item-title>
          </v-list-item>

          <v-list-item to="/worker">
            <v-icon>mdi-shield-account</v-icon>
            <v-list-item-title>Ο λογαριασμός μου</v-list-item-title>
          </v-list-item>
          <v-list-item @click="logout">
            <v-icon>mdi-logout-variant</v-icon>

            <v-list-item-title>Αποσύνδεση</v-list-item-title>
          </v-list-item>
        </v-list>
      </v-menu>
    </v-toolbar>
    </v-card>
    <v-container class="spacing-playground py-12">
    <router-view> </router-view>
    </v-container>
    <div class="footer">

        {{ new Date().getFullYear() }} — <strong>ICU-Manager</strong>
    </div>
  </v-app>
</template>
```

Εικόνα 13. *Η δομή ενός αρχείου .vue για την ανάπτυξη όψεων/στοιχείων*

Εντός του σεναρίου εκτέλεσης, με βάση το “options API”, υπάρχει μια κύρια μέθοδος με επιλογές:

```
<script>

export default {
  name: "App",
  data: () => ({
  }),
  computed: {
    isAuthenticated() {
      return this.$store.getters.isAuthenticated
    }
  },
  methods: {
    async logout() {
      await this.$store.commit('LogOut')
      await this.$router.push('/login')
    },
  },
};
</script>

<style>
.footer {
  position:fixed;
  bottom:0;
  color: azure;
  background-color: #333333;
  display: inline-block;
  text-align: center;
  width: 100%;
  height: 33px;
}
</style>
```

Εικόνα 14. *Τυπικό παράδειγμα χρήσης του “options API”*

Παρατηρούμε για μια ακόμα φορά την οργάνωση κλειδιού-τιμής που αιτιολογεί την ονομασία “options”, αφού εξάγονται διάφορες «επιλογές», όπως το όνομα, οι μεταβλητές εντός της εμβέλειας του στοιχείου, μέθοδοι που επιστρέφουν διαφορετικές τιμές με βάση την αλλαγή κατάσταση και υλοποιήσεις μεθόδων. Αυτές οι επιλογές είναι μόλις οι μισές από αυτές που διαθέτονται. Υπάρχουν οι επιλογές για προσθήκη επιπλέον στοιχείων εντός του ίδιου του στοιχείου (η δενδροειδής ιεραρχία που αναφέρθηκε μεταξύ των στοιχείων)[50], επιλογές για χρήση μεταβλητών που προέρχονται από ανώτερο στοιχείο (props)[51], επιλογή για χρήση αντιδραστικών μεταβλητών (watchers)[52] και επιλογές για κάθε άγκιστρο του κύκλου ζωής[53].

Η αρχικοποίηση του κυρίως αντικειμένου Vue λαμβάνει μέρος εντός του αρχείου “main.js”:

```
Vue.use(VueApexCharts)
Vue.component('apexchart', VueApexCharts)

new Vue({
  el: "#app",
  router,
  vuetify,
  store,
  // i18n,
  render: h => h(App)
}).$mount('#app')
```

Εικόνα 15. *Οι επιλογές που δίνονται στην κατασκευάστρια συνάρτηση του Vue αντικειμένου*

Οι επιλογές και τα στοιχεία καταχωρούνται στο αντικείμενο και έτσι είναι προσβάσιμες σε ολόκληρη την εμβέλεια της εφαρμογής. Τα πρόσθετα που χρησιμοποιούνται ως επιλογές είναι το router για τη διαχείριση των διαδρομών από τα στοιχεία, το Vuetify που προέρχεται από την ομώνυμη βιβλιοθήκη και παρέχει έτοιμα στοιχεία τα οποία βελτιώνουν το UI (εμφάνιση και αισθητική) και βοηθούν το χρήστη να αλληλεπιδράσει αποτελεσματικότερα με την εφαρμογή, ενώ το store το οποίο προέρχεται από το αρχείο-πρόσθετο "auth.js" βασίζεται στη βιβλιοθήκη Vuex που στοχεύει στην οργανωμένη διαχείριση της κατάστασης του πλαισίου και την προστασία των καθολικών μεταβλητών από επιπλοκές της αντιδραστικότητας. Επιπλέον, προστίθεται ένα στοιχείο για χρήση σε ολόκληρη την εφαρμογή, το "VueApexCharts" από τη βιβλιοθήκη "Apexcharts", το οποίο αναλαμβάνει τη σχεδίαση και διαχείριση γραφημάτων που βελτιώνουν και εξωραΐζουν την προβολή των στατιστικών.

Παρόμοια με την Rails, έτσι και το Vue.js απαιτεί ένα διακομιστή για να μπορεί να λειτουργήσει και εκτελείται στο δίκτυο στη θύρα 8000 (η Rails στην 3000· πρέπει να είναι διαφορετικές) με τη βοήθεια του διαχειριστή πακέτων NPM. Ο NPM (διαχειριστής πακέτων που συνεργάζεται με το Node.js[54]) διαχειρίζεται επίσης τα πακέτα και τις εξαρτήσεις τους που αναγράφονται στο "package.json". Έτσι, η εντολή

```
> npm init vue@latest
```

εκτελεί ένα σενάριο με ερωτήσεις μέσω γραμμής εντολών για τον τρόπο διαμόρφωσης του έργου και οι εντολές

```
> npm install
> npm run dev
```

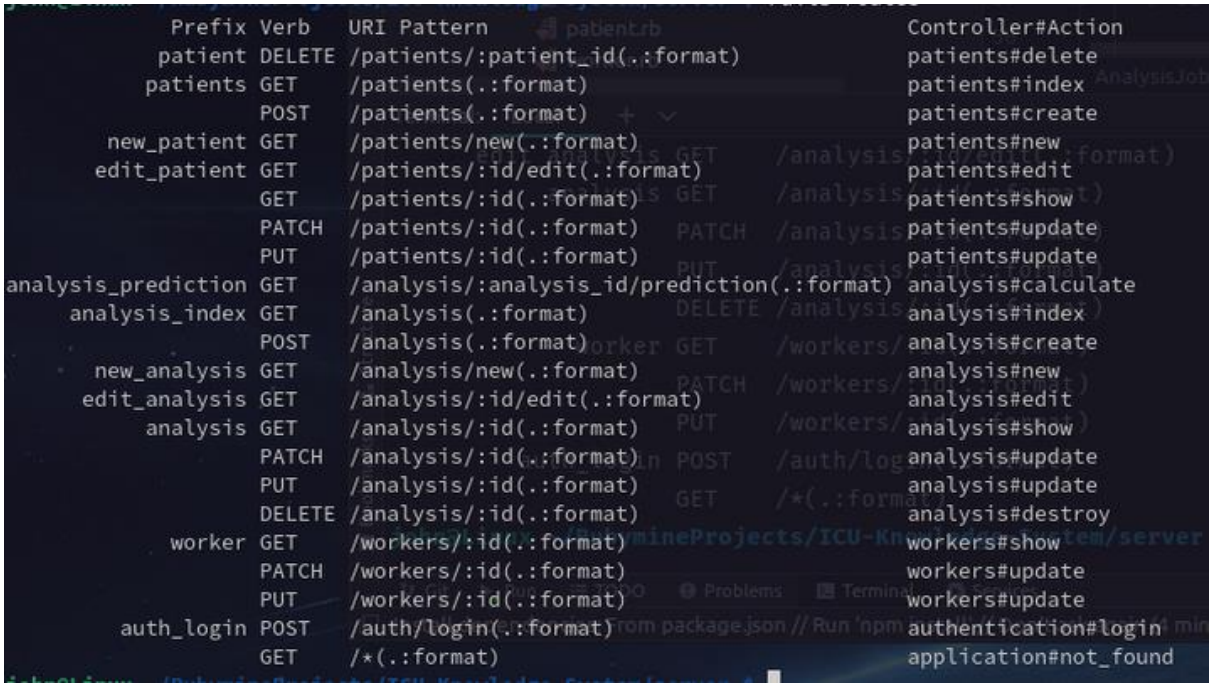
ικανοποιούν τις εξαρτήσεις και στη συνέχεια εκκινούν το διακομιστή που εκτελεί το πλαίσιο.

2.3.3 Διαδρομές της εφαρμογής

Οι διαθέσιμες διαδρομές για το REST API προκύπτουν εάν δώσουμε την εντολή

rails routes

και είναι οι εξής:



Prefix	Verb	URI Pattern	Controller#Action
patient	DELETE	/patients/:patient_id(.:format)	patients#delete
patients	GET	/patients(.:format)	patients#index
	POST	/patients(.:format)	patients#create
new_patient	GET	/patients/new(.:format)	patients#new
edit_patient	GET	/patients/:id/edit(.:format)	patients#edit
	GET	/patients/:id(.:format)	patients#show
	PATCH	/patients/:id(.:format)	patients#update
	PUT	/patients/:id(.:format)	patients#update
analysis_prediction	GET	/analysis/:analysis_id/prediction(.:format)	analysis#calculate
analysis_index	GET	/analysis(.:format)	analysis#index
	POST	/analysis(.:format)	analysis#create
new_analysis	GET	/analysis/new(.:format)	analysis#new
edit_analysis	GET	/analysis/:id/edit(.:format)	analysis#edit
analysis	GET	/analysis/:id(.:format)	analysis#show
	PATCH	/analysis/:id(.:format)	analysis#update
	PUT	/analysis/:id(.:format)	analysis#update
	DELETE	/analysis/:id(.:format)	analysis#destroy
worker	GET	/workers/:id(.:format)	workers#show
	PATCH	/workers/:id(.:format)	workers#update
	PUT	/workers/:id(.:format)	workers#update
auth_login	POST	/auth/login(.:format)	authentication#login
	GET	/*(.:format)	application#not_found

Εικόνα 16. *Οι διαδρομές του πλαισίου της Rails*

Οι διαδρομές των ιστοσελίδων (URL paths) βρίσκονται στο index.js στον κατάλογο “/src/router”:

```
{
  path: '/',
  name: 'Index',
  component: Main,
  meta: { requiresAuth: true }
},
{
  path: '/patients',
  name: 'View patients',
  component: Patients,
  props: true,
  meta: { requiresAuth: true }
},
{
  path: '/patients/create',
  name: 'Create patient',
  component: PatientCreate,
  meta: { requiresAuth: true }
},
{
  path: '/worker',
  name: 'Employees profile page',
  component: Worker,
  meta: { requiresAuth: true }
},
/*{
  path: '/admin',
  name: "Administrator's page",
  component: Admin
},*/
{
  path: '/analysis/:AMKA',
  name: 'statistics',
  component: General,
  props: true,
  meta: { requiresAuth: true }
},
{
  path: '/construct',
  component: Construction,
  name: 'Under construction',
  meta: { requiresAuth: true }
},
{
  path: '/login',
  name: "Login",
  component: Login,
  //meta: { guest: true },
},
{
  path: '*',
  name: 'NotFound',
  component: NotFound
}
```

Εικόνα 17. *Οι διαδρομές του πλαισίου Vue.js*

2.3.4 Ασφάλεια του συστήματος

Αναφορικά με την ασφάλεια του συνολικού συστήματος («πίσω» και «μπροστά» μέρη), χρησιμοποιείται η τεχνική του JWT, σύμφωνα με την οποία, οποιοδήποτε αίτημα στο REST API ικανοποιείται εάν και μόνο εάν συνοδεύεται από μια συμβολοσειρά (token) που έχει κωδικοποιηθεί και εγκριθεί από τον ίδιο το

διακομιστή (την Rails). Η υλοποίηση αυτής της τεχνικής στην εφαρμογή έχει ως εξής: Κάθε κλάση ελεγκτή περιέχει τη μακροεντολή

```
before_action :authorize_request
```

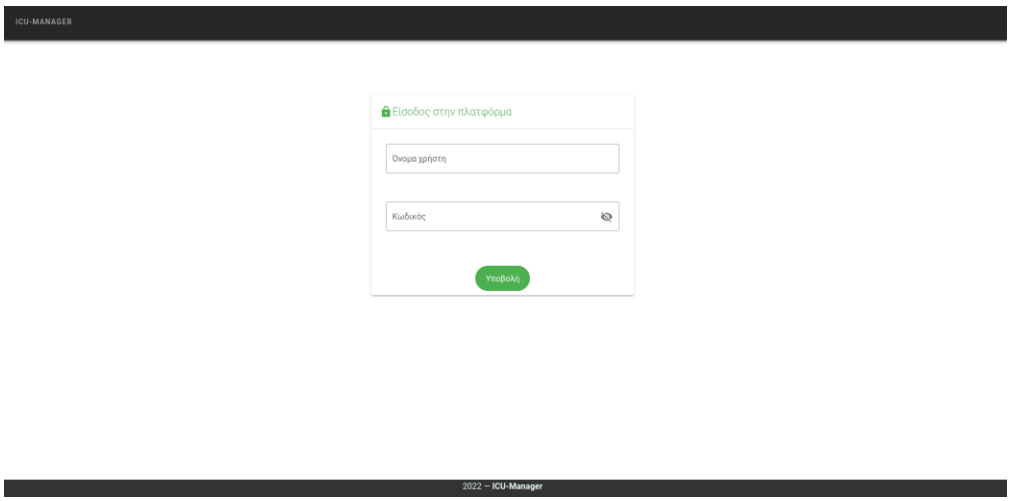
η οποία ανακατευθύνει το πλαίσιο στη μέθοδο

```
def authorize_request
  header = request.headers['Authorization']
  header = header.split(' ').last if header
  begin
    @decoded = JsonWebToken.decode(header)
    @current_user = Worker.find(@decoded[:user_id])
  rescue JWT::DecodeError => e
    render json: { errors: e.message }, status: :unauthorized
  rescue ActiveRecord::RecordNotFound => e
    render json: { errors: e.message }, status: :unauthorized
  end
end
```

Εικόνα 18. *Η μέθοδος για την εξουσιοδότηση χρήστη*

του ApplicationController και η μέθοδος αυτή ελέγχει εάν το αίτημα περιέχει ένα αποδεκτά κωδικοποιημένο JWT μέσω της decode από το μοντέλο JsonWebToken που έχουμε δει. Εάν η αποκωδικοποίηση είναι επιτυχής, τότε το αίτημα είναι δεκτό και μπορεί να πραγματοποιηθεί η ζητούμενη ενέργεια, αλλιώς προκύπτει σφάλμα αυθεντικοποίησης.

Πώς όμως κάποιος αποκτά το JWT σύμβολο; Μέσω της εισόδου στο σύστημα, την οποία αναλαμβάνει το στοιχείο εισόδου “Login” στο Vue.js που προτρέπει το χρήστη να εισάγει το όνομα χρήστη και τον κωδικό που του αποδόθηκαν:



Εικόνα 19. *Προτροπή εισόδου στοιχείων σύνδεσης*

Το στοιχείο αυτό καλεί μέθοδο και στέλνει αίτημα σύνδεσης στην Rails στη διαδρομή “/auth/login”.

```ruby
# POST /auth/login
def login
  @user = Worker.find_by username: login_params[:username]
  if @user&.authenticate(login_params[:password])
    token = JsonWebToken.encode(user_id: @user._id)
    # time = Time.now + 24.hours.to_i
    # exp: time.strftime("%m-%d-%Y %H:%M"),
    render json: { token: token, id: @user._id.to_s }, status: :ok
  else
    render json: { error: 'Login failed' }, status: :unauthorized
  end
end
```

Εικόνα 20. *Η μέθοδος για τη σύνδεση του χρήστη*

εκεί το όνομα και ο κωδικός που δόθηκαν από το χρήστη ελέγχονται με αυτά που είναι αποθηκευμένα στη ΒΔ και εάν υπάρχουν, κωδικοποιείται ένα νέο JWT και στέλνεται στο Vue.js, ειδάλλως υπάρχει σφάλμα εισόδου και ο χρήστης ειδοποιείται με μήνυμα. Το Vue αποθηκεύει το σύμβολο που έλαβε σε κρυφή μνήμη του περιηγητή και το διαχειρίζεται μέσω του Vuex. Το Vuex παρέχει έναν ικανοποιητικό αριθμό μεθόδων και επιλογών για τη διαχείριση του συνδέσεων και της αυθεντικοποίησης. Η κλήση του γίνεται στο αρχείο “auth.js” με τη δημιουργία νέου αντικειμένου Store (παρόμοια με το αντικείμενο Vue):

```
const token = localStorage.getItem('accessToken');
const user = localStorage.getItem('user');

const initialState = token
  ? { loggedIn: true , user:user,loginError:null }
  : { loggedIn: false , user: null,loginError:null };

const store = new Vuex.Store({
  namespaced: true,
  state: initialState,
  mutations: {
    loginSuccess(state, id) {
      state.loggedIn = true;
      state.user = id;
      state.loginError=null
    },
    loginFailure(state) {
      state.loggedIn = false;
      state.loginError = "Invalid username or password !"
      state.user = null;
      console.log(this.state)
    },
    LogOut(state){
      localStorage.clear()
      state.loggedIn= false
      state.user = null
    }
  },
  getters: {
    isAuthenticated: (state) => state.loggedIn,
  },

  actions: {
    doLogin({ commit }, loginData) {
      axios.post('http://localhost:3000/auth/login', {
        ...loginData
      })
      .then((response) => {
        localStorage.setItem('accessToken', response.data.token)
        localStorage.setItem('user', response.data.id)
        commit('loginSuccess', response.data.id);
        router.push('/').then()
      })
      .catch(() => {
          commit('loginFailure');
          console.log(this.state.loggedIn)
      })
    },
    fetchAccessToken({ commit }) {
      commit('updateAccessToken', localStorage.getItem('accessToken'));
    }
  }
})
```

Εικόνα 21. *Τα περιεχόμενα του αρχείου auth.js*

Κάθε αίτημα, λοιπόν, με τη βοήθεια του Vuex «προετοιμάζεται» από το Vue προτού αποσταλεί στο REST API. Για παράδειγμα, για τη δημιουργία ασθενή, έχουμε:

```
return {
  headers: {headers:{
      'Authorization': `Basic ${localStorage.getItem("accessToken")}`
    }
  },
```

και

```
axios
  .post("http://localhost:3000/patients", this.patient, this.headers)
  .then(() => {
    if (this.information){
      this.information.AMKA = this.patient.AMKA
      axios.post("http://localhost:3000/analysis", this.information, this.headers)
      .then(() => {
        this.$router.push({
          name: "View patients",
          params: { message: "Προστέθηκε ασθενής " },
        })
        this.snackbar = true
      })
      .catch((err)=>{
        this.$router.push({
          name: "View patients",
          params: { message: err },
        })
        this.snackbar = true
      });
    }
  })
```

Εδώ πρέπει να σημειωθεί ότι συμβάλλει και το ίδιο το Vue στην προστασία των πόρων, καθώς πριν από την φόρτωση μιας σελίδας ελέγχεται η ύπαρξη του JWT:

```
router.beforeEach((to, from, next) => {

  let isAuthenticated = store.state.loggedIn
  console.log(isAuthenticated)
  if (!isAuthenticated && to.matched.some(record => record.meta.requiresAuth)) {
    next('/login')
  } else {
    next()
  }
})
```

Όταν δεν υπάρχει, τα αιτήματα ανακατευθύνονται στην όψη για προτροπή εισόδου.

ΚΕΦ.3: Αποτελέσματα και παρατηρήσεις

3.1. Σχέση με εξόρυξη δεδομένων

Επικρατεί η αντίληψη πως η διαδικασία εκπαίδευσης μοντέλων για την αναζήτηση σε ΒΔ και την ανακάλυψη μοτίβων συνιστά εξόρυξη δεδομένων. Στην πραγματικότητα, όμως, παρά την άμεση σύνδεση εννοιών όπως η μηχανική μάθηση, η εξόρυξη δεδομένων και η ανακάλυψη γνώσης από δεδομένα (KDD), η σχέση τους είναι περισσότερο ιεραρχική.

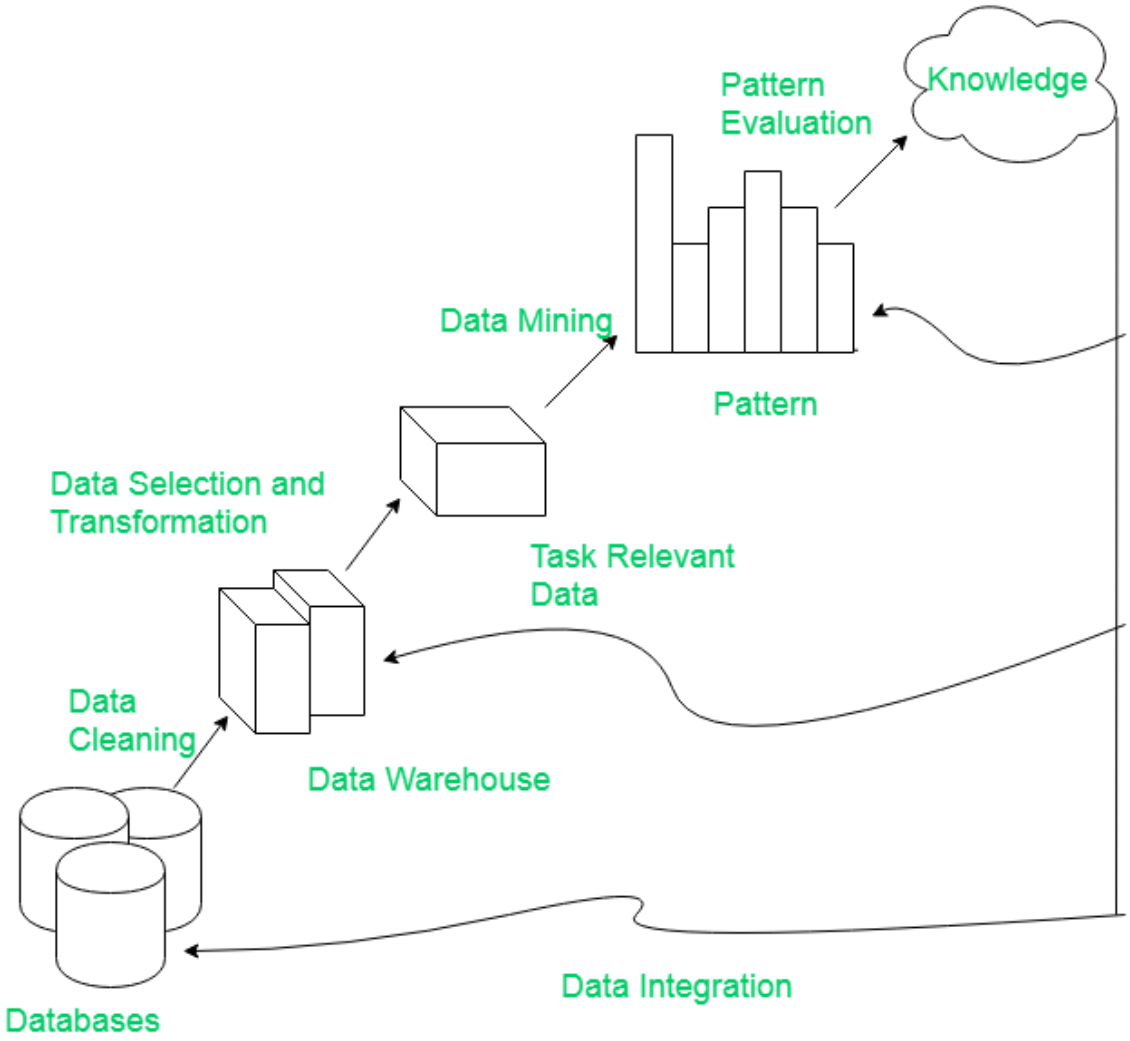


Figure 4. *Φάσεις της διαδικασίας ανακάλυψης γνώσης*

Όπως διακρίνεται και στο σχήμα, η εξόρυξη δεδομένων αποτελεί ένα κομμάτι της ευρύτερης διαδικασίας ανακάλυψης γνώσης από υπάρχοντα δεδομένα αποθηκευμένα σε ΒΔ. Αντίστοιχα, η εξόρυξη δεδομένων είναι μια διαδικασία που περιλαμβάνει τεχνικές μηχανικής μάθησης στη φαρέτρα της. Η διαφορά έγκειται στο ότι η μηχανική μάθηση αναπτύσσει και βελτιώνει τα μοντέλα και την ακρίβειά τους ή

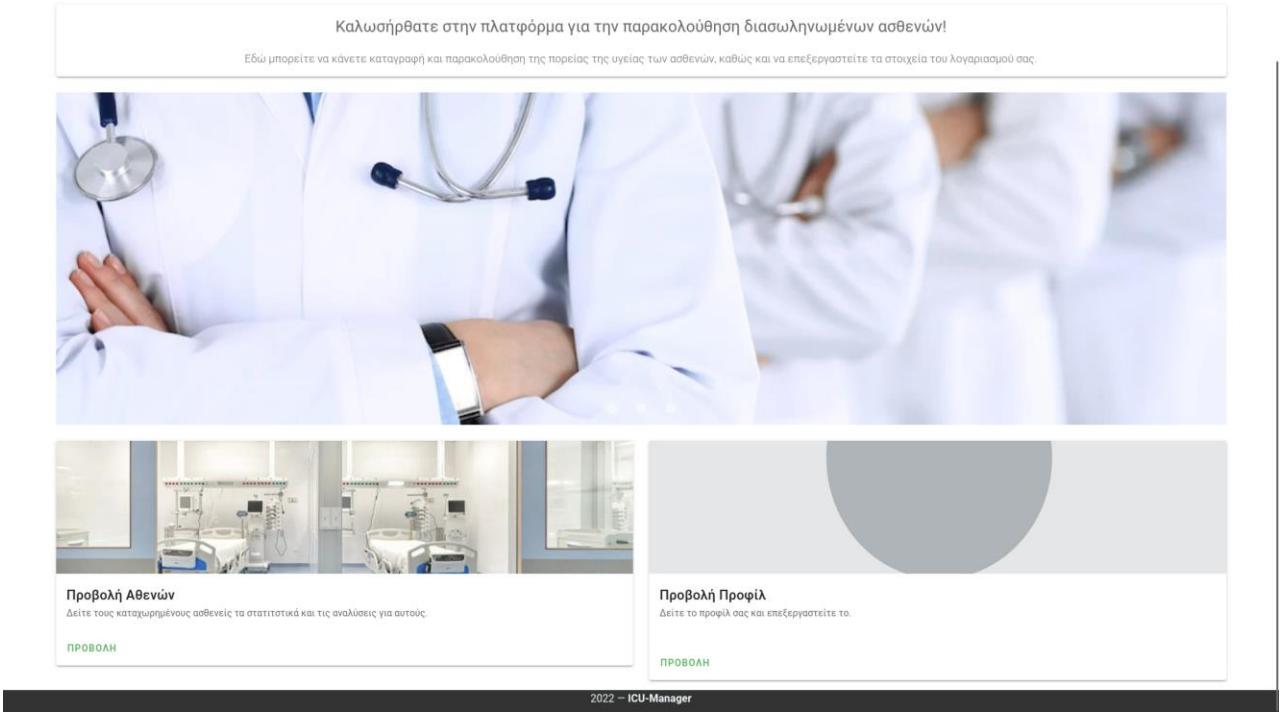
προσθέτει νέες τεχνικές και αλγορίθμους, ενώ η εξόρυξη εστιάζει στη βελτίωση της απόδοσης και της επεκτασιμότητας του τρόπου με τον οποίο ανακαλύπτονται νέα στοιχεία στις ΒΔ και προτείνει τρόπους με τους οποίους μπορεί να γίνει διαχείριση των δεδομένων και των τύπων τους. Η εφαρμογή που εξετάζεται προβάλει ορισμένα συμπεράσματα με βάση την εκπαίδευση ενός μοντέλου ΤΝΔ πάνω σε υπάρχοντα δεδομένα. Εντούτοις, μπορεί να παρατηρήσει κανείς ότι:

- Το σύνολο δεδομένων είναι αρκετά μικρό
- Υπάρχουν έτοιμα κριτήρια για την πορεία των ασθενών που ενεργοποιούν τις ειδοποιήσεις
- Το σύστημα μπορεί να εκπαιδευτεί από μελλοντικές περιπτώσεις

Από αυτά τα χαρακτηριστικά προκύπτει ότι δεν πρόκειται για ένα σύστημα που ανακαλύπτει νέα ιατρικά κριτήρια, αλλά βασίζεται σε ήδη υπάρχοντα, επομένως πρόκειται για ένα ευφυές σύστημα υποστήριξης κλινικής απόφασης έγκαιρης ειδοποίησης.

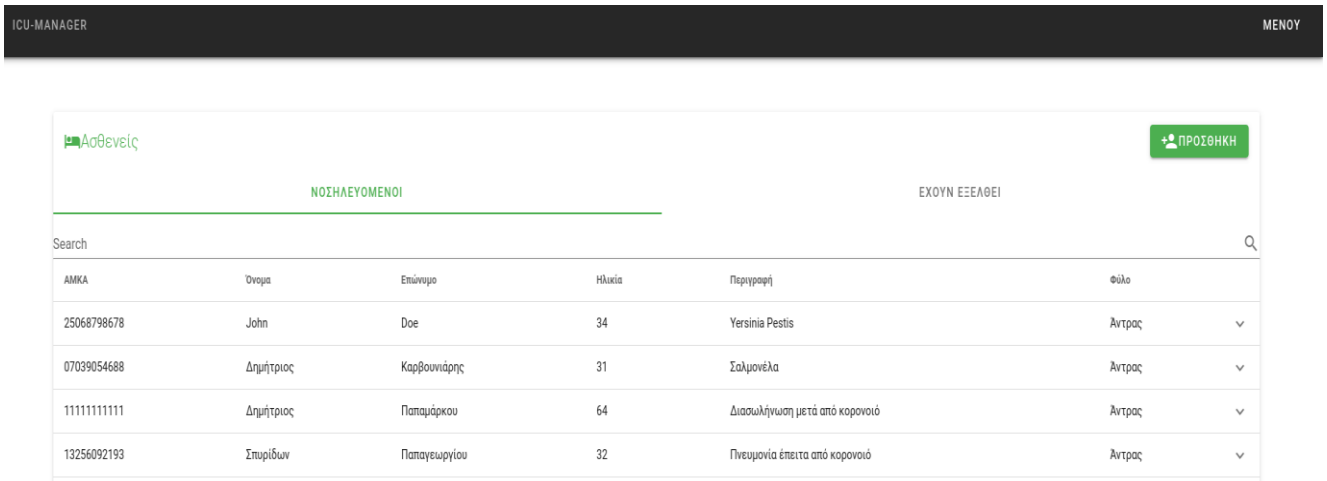
3.2. Τελική απεικόνιση

Με την ολοκλήρωση της εφαρμογής, γίνεται διαθέσιμη στο χρήστη η τελική διεπαφή με την οποία θα αλληλεπιδρά:

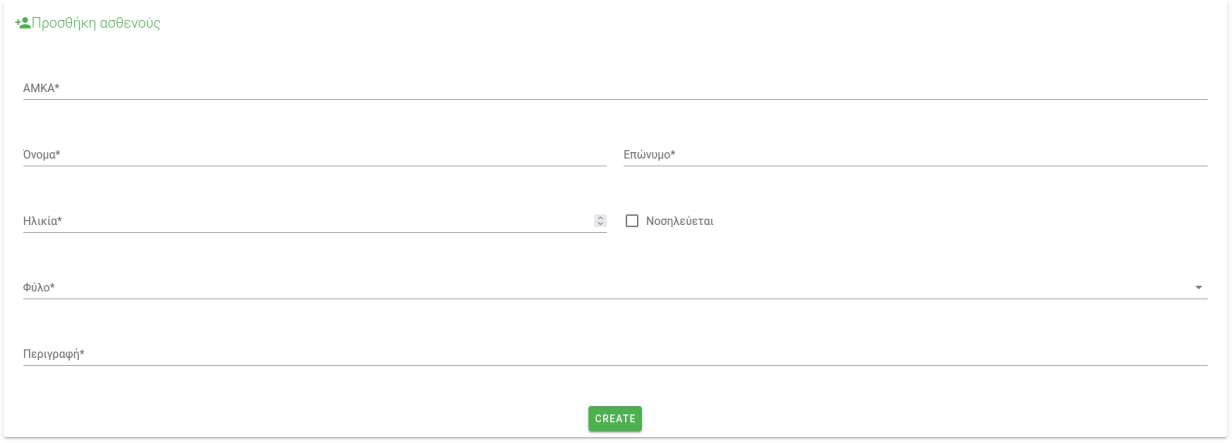


Εικόνα 22. *Η αρχική σελίδα της εφαρμογής*

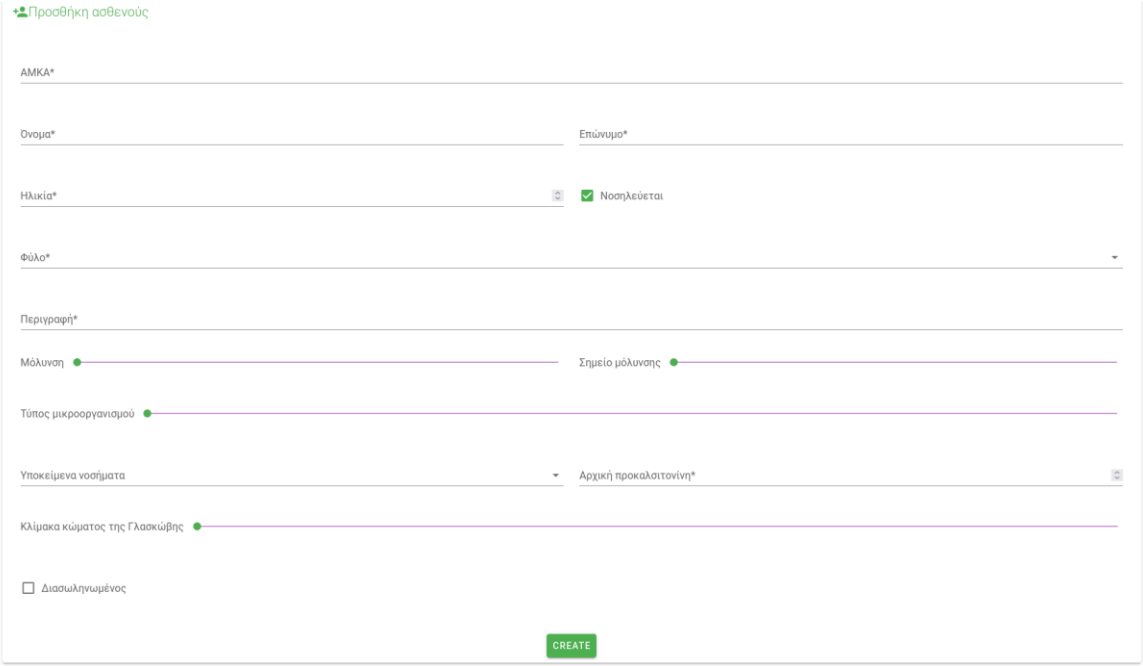
Διαχείριση ασθενών:



Εικόνα 23. *Προβολή των ασθενών, είτε βρίσκονται σε ΜΕΘ είτε έχουν εξέλθει*

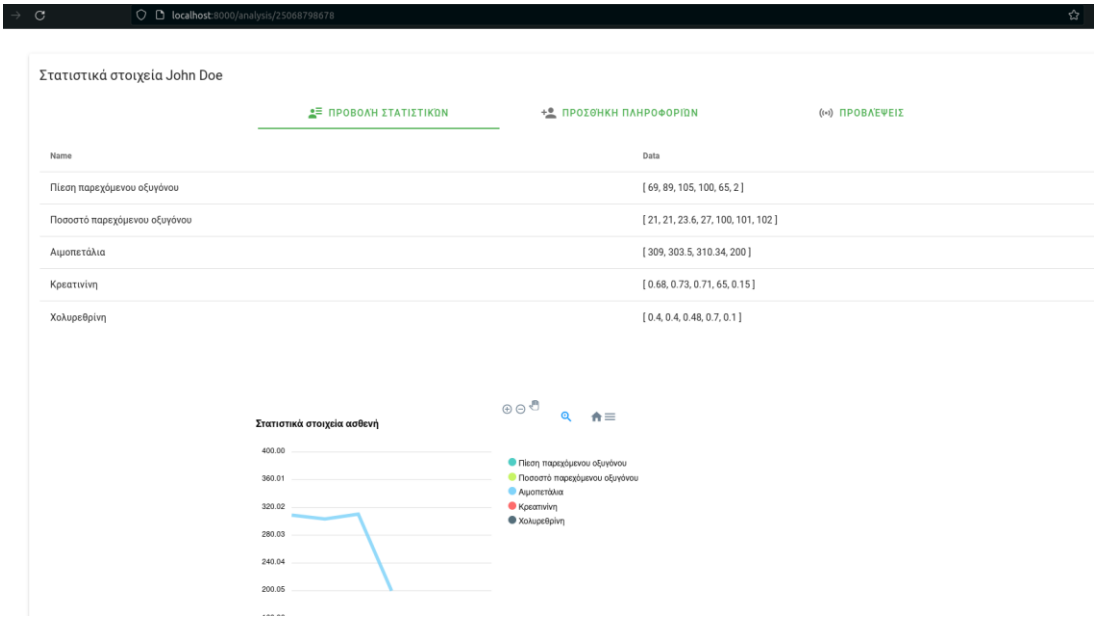


Εικόνα 24. *Προσθήκη νέου ασθενή*

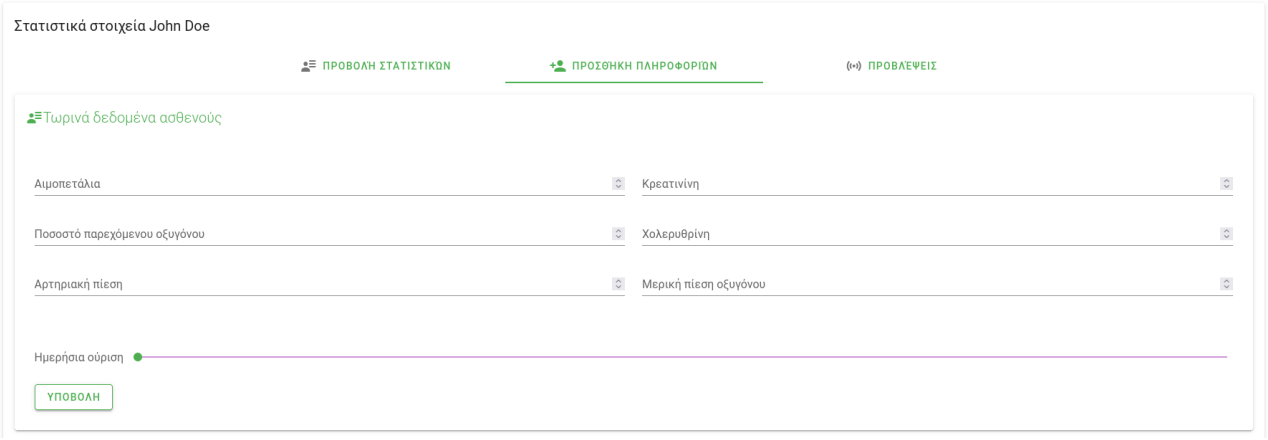


Εικόνα 25. *Εάν νοσηλεύεται και δεν προστίθεται για διατήρηση ιστορικού, απαιτούνται επιπλέον παράμετροι*

Προβολή των αναλυτικών για τους ασθενείς:

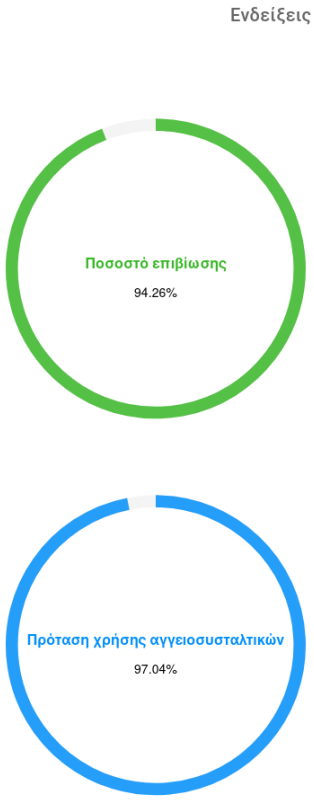


Εικόνα 26. *Καρτέλα με πίνακα και διάγραμμα προβολής της πορείας των ουσιών του ασθενή στο χρόνο*



Εικόνα 27. *Καρτέλα για την προσθήκη νέων δεδομένων πραγματικού χρόνου του ασθενή*

Εισάγοντας τα δεδομένα του ασθενή από το σύνολο εκπαίδευσης, παρατηρούμε ότι προκύπτουν παρόμοια αποτελέσματα με μορφή ποσοστών πιθανοτήτων:



Εικόνα 28. *Η τελευταία καρτέλα προβάλει δυο διαγράμματα με προβλέψεις του ΤΝΔ*

Αυτά τα αποτελέσματα προκύπτουν από αυτές τις δυο μεθόδους:

```
def filter_results(results)
  results[0] = 1.0 - results[0]
  results.map! do |x|
    x = x * 100
    x = x.round(2)
  end
end
```

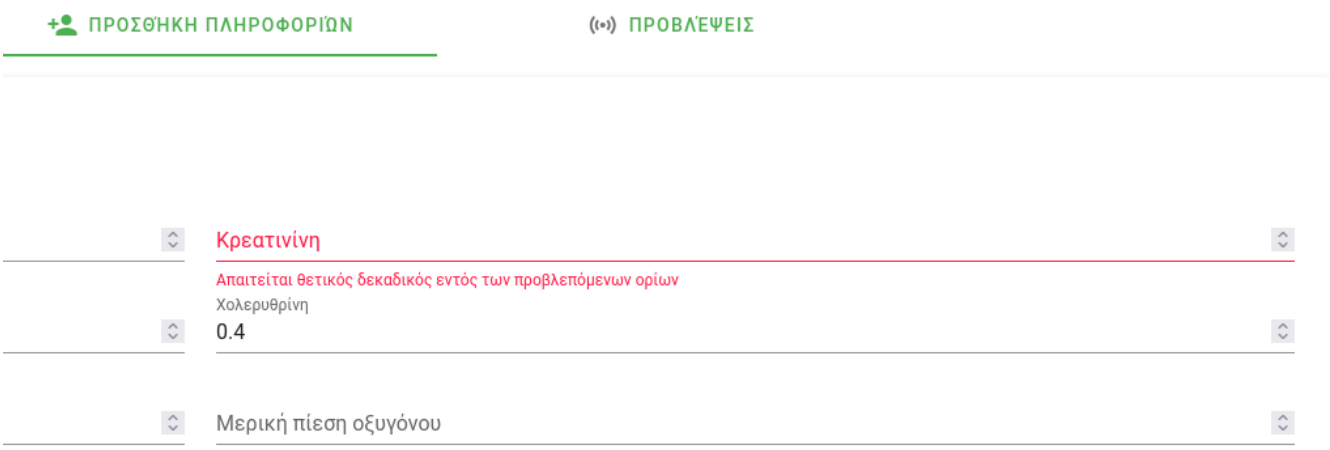
```
updateChart() {
  this.series1 = [this.response[0]]
  this.series2 = [this.response[1]]

  let color;
  if (this.series1[0] < 30.0) color = '#de1037'
  else if (this.series1[0] < 60.0) color = '#f1b626'
  else color = '#38b726'

  this.mortality = {

    colors: [color],
    series: this.series1,
    plotOptions: {
      radialBar: {
        hollow: {
          size: 80,
        }
      },
    },
    labels: ['Πιθανότητα επιβίωσης'],
  }
  this.vasopressors = {
    series: this.series2,
    // series: [{data: this.response[1]}],
    plotOptions: {
      radialBar: {
        hollow: {
          size: 80,
        }
      },
    },
    labels: ['Πρόταση χρήσης αγγειοσυσταλτικών'],
  }
}
```

Εικόνα 29. *Μέθοδοι που μετατρέπουν το αποτέλεσμα του δικτύου σε ποσοστά*

Η πρώτη προέρχεται από το περιβάλλον Rails και υπολογίζει το συμπλήρωμα της θνητότητας (πιθανότητα επιβίωσης) και το μετατρέπει σε ποσοστό, ενώ η μέθοδος του Vue το μετατρέπει σε γράφημα (το χρώμα μεταβάλλεται με βάση τον κίνδυνο που διατρέχει ο ασθενής).

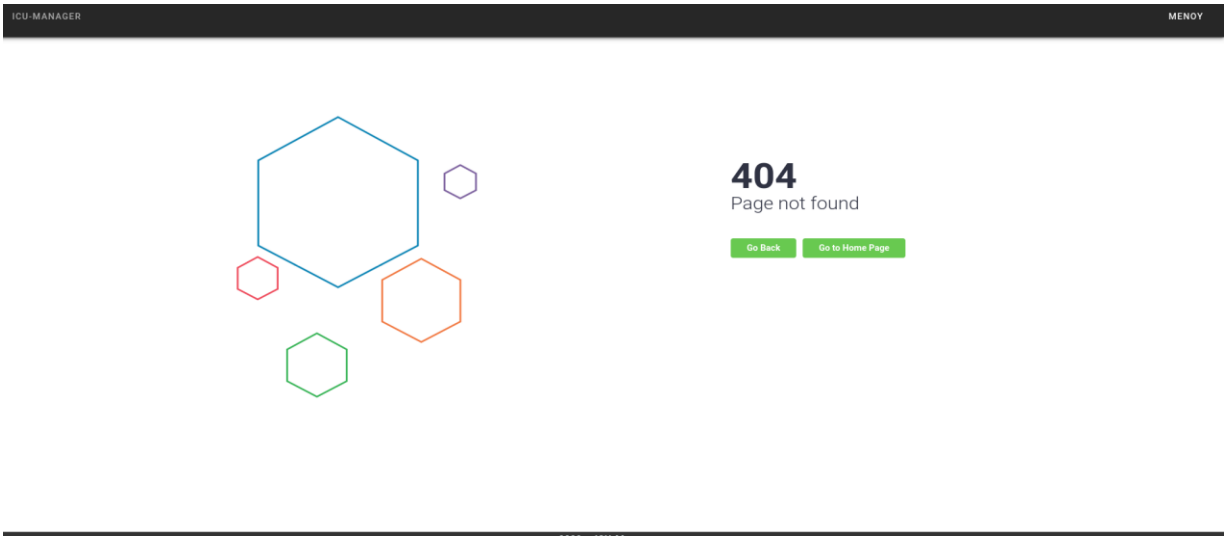
Στα στοιχεία που πραγματοποιείται προσθήκη δεδομένων, υπάρχει ο αναγκαίος έλεγχος εισόδου:

Εικόνα 30. *Έλεγχοι κατά την προσθήκη ασθενούς*



Εικόνα 31. *Έλεγχοι κατά την προσθήκη δεδομένων αληθινού χρόνου*

Εάν κάποια διαδρομή δεν υπάρχει, εμφανίζεται ενδεικτικό σφάλμα:



Εικόνα 32. *Όψη για την περίπτωση που δε βρέθηκε ο ζητούμενος πόρος (404)*

3.3. Λειτουργία και δεδομένα εισόδου

Σε αυτό το σημείο θα ήταν χρήσιμο να αναφερθούν οι επιλογές που μπορεί να δεχθεί το σύστημα για κάθε πεδίο του ασθενή, αλλά και τα πεδία που τον περιγράφουν στη ΒΔ.

Πίνακας 2. Τα δεδομένα που εισάγει ο χρήστης

Όνομα	Σύμβολο	Τιμές	Περιγραφή
Περιγραφή	**description**	Κείμενο	Ο κύριος παράγοντας που οδήγησε τον ασθενή για νοσηλεία.
Μόλυνση	**inflammation**	[1-3]	1=Βακτηριαιμία, 2=Τοπικό σημείο, 3=Και τα δυο
Σημείο μόλυνσης	**location**	[1-7]	1 = PNA, 2 = UTI, 3 = MSK, 4 = Εγκέφαλος, 5 = Καρδιά, 6 = GI, 7 = Απροσδιόριστο
Τύπος μικροοργανισμού	**organism**	[1-4]	1 = Gram θετικός, 2 = Gram αρνητικός, 3 =απροσδιόριστο, 4 = ικός
Κλίμακα κώματος Γλασκόβης	**GCS**	[2-15]	Νευρολογική κλίμακα για τον προσδιορισμό του κώματος.
Αιμοπετάλια	**PLT**	[3-726]	Επίπεδο αιμοπεταλίων στο αίμα.
Κρεατινίνη	**CR**	[0,15-15,1]	Επίπεδο κρεατινίνης στο αίμα.
Ποσοστό παρεχόμενου οξυγόνου	**FiO2**	[21-262]	Το ποσοστό του οξυγόνου που παρέχεται στον ασθενή (κυρίως όταν δέχεται αερισμό).
Χολερυθρίνη	**BIL**	[0,1-22,5]	Επίπεδο χολερυθρίνης στο αίμα.
Αρτηριακή πίεση	**MAP**	[9-138]	Η μέση αρτηριακή πίεση κατά το μονό καρδιακό κύκλο.
Μερική πίεση οξυγόνου	**PaO2**	[2-595]	Η πίεση του οξυγόνου στις αρτηρίες για να διαπιστωθεί η απορρόφησή του από τους πνεύμονες.
Ημερήσια ούρηση	**UoP**	[1-3]	Τα ml ούρων του ασθενή εντός ημέρας (1=500+, 2=200-500, 3= 200-).

Αυτά τα σύμβολα εμφανίζονται σε διάφορα σημεία της εφαρμογής διότι έτσι πραγματοποιείται ευκολότερα η ανταλλαγή των δεδομένων ανάμεσα στα δυο πλαίσια.

3.4. Δυσκολίες και επιπλοκές

Ο Τόμας Έντισον είχε πει ότι «Δεν απέτυχα. Απλώς ανακάλυψα 10.000 τρόπους που δε λειτουργούν». Αυτή η εργασία δεν αποτελεί εξαίρεση, καθώς πραγματοποίησα δοκιμές με τη χρήση διαφορετικών πλαισίων, συστημάτων και βιβλιοθηκών μέχρι να προκύψει ένα σύστημα με την απαιτούμενη λειτουργικότητα. Για παράδειγμα, η πρωταρχική κίνηση ανάπτυξης του συστήματος ήταν η εγγραφή μοντέλων και υπηρεσιών για επικοινωνία με τη ΒΔ στο πλαίσιο του NestJS. Έχουμε αναφέρει ότι το πλαίσιο NestJS βασίζεται στην JavaScript (Node.js συγκεκριμένα), αν και για τη συγγραφή επικροτείται η χρήση της TypeScript, παρόμοιας γλώσσας που χρησιμοποιεί δηλωμένους τύπους (strongly typed) για τις μεταβλητές της. Το NestJS επέτρεπε τη δημιουργία εφαρμογών διακομιστή και ήταν αρκετά συνεργάσιμο με την αποθήκευση και ανάκληση μοντέλων από τη ΒΔ μέσω του Mongoose, της ODM αντίστοιχου του Mongoid., ενώ για τις λειτουργίες CRUD απαιτούνταν η δημιουργία υπηρεσιών που θα τις υλοποιούσαν. Το πρόβλημα που παρουσιάστηκε αφορούσε αρχικά τη χρήση του νευρωνικού δικτύου από τη βιβλιοθήκη Brain.js[55], η οποία παρόλο που βασιζόταν στην TypeScript, εμφάνιζε προβλήματα συμβατότητας με το πλαίσιο (ενδεχομένως οι τύποι κλάσεων δεν υλοποιούνταν σωστά). Επίσης, η διαχείριση σφαλμάτων στην TypeScript ήταν πολύ πιο δύσκολη (επιστροφή σφάλματος στο χρήστη μέσω REST). Η Rails δεν παρουσιάζει κανένα από αυτά τα ζητήματα.

Πέραν του διακομιστή, η σχεδίαση των όψεων έχει επίσης περάσει από ραγδαίες αλλαγές, από την αρχή της ανάπτυξης, μέχρι και τη στιγμή που γράφονται αυτές οι σελίδες. Στην αρχή χρησιμοποιούσα έτοιμα πρότυπα ιστοσελίδων που περιέχουν τα ίδια στοιχεία HTML με συγκεκριμένη σχεδίαση CSS και σενάρια JavaScript, αντί για βιβλιοθήκες ανάπτυξης UI. Ένα ολοκληρωμένο σύστημα, όμως, που συνιστά εξατομικευμένη λύση σε ένα πρόβλημα είναι προτιμότερο να αναπτύσσεται εκ του μηδενός, με πολύ λίγα στοιχεία παρμένα από άλλα έργα. Επιπλέον, ενδέχεται να υπάρχουν επιπλοκές όσον αφορά την άδεια χρήσης και τα πνευματικά δικαιώματα.

Τέλος, κάτι που οφείλω να σημειώσω είναι πως το δείγμα των ασθενών πιθανότατα είναι αρκετά μικρό για την αποτελεσματικότητα του δικτύου. Το σύνολο, ασφαλώς, μπορεί να αυξηθεί μακροχρόνια, με την προσθήκη έτοιμων δεδομένων ή πραγματικού χρόνου από ασθενείς που νοσηλεύονται. Ένας ικανοποιητικός αριθμός δεδομένων για την εκπαίδευση του μοντέλου θα ήταν τουλάχιστον 5.000-10.000[58]. Ευελπιστώ ότι αυτό το σύνολο θα είναι διαθέσιμο στο μέλλον με την αναγνώριση του έργου.

ΣΥΜΠΕΡΑΣΜΑΤΑ

Εν κατακλείδι, παρά τη μικρή έκταση του συστήματος, αποτελεί ικανοποιητική ένδειξη συνεργασίας ανάμεσα στις νέες τεχνολογίες που κυκλοφορούν και εξελίσσονται διαρκώς στον τομέα της πληροφορικής και στο στιβαρό κλάδο της ιατρικής ο οποίος εξελίσσεται εξίσου ραγδαία και εξυπηρετεί τον άνθρωπο από αρχαιοτάτων χρόνων. Οι δυνατότητες του συστήματος είναι απεριόριστες και είναι βέβαιο ότι οι τεχνολογίες που την αποτελούν μπορούν να εξελιχθούν παραπάνω. Ήδη κυκλοφορούν οι εκδόσεις 7 για την Rails και 3 για το Vue.js (οι εκδόσεις που χρησιμοποιούνται είναι 6 και 2, αντίστοιχα). Από πλευράς εφαρμογής, θα μπορούσε να βελτιστοποιηθεί η βιβλιοθήκη τεχνητής νοημοσύνης ή να χρησιμοποιούνταν ένα πιο αποτελεσματικό μοντέλο για τις προβλέψεις του συστήματος. Επιπρόσθετα, κρίνεται επιτακτικό να προβλέπονται ακόμα περισσότερα στοιχεία, όπως τύποι φαρμάκων που πρέπει να λάβει ο ασθενής, πιθανές ημέρες παραμονής ή και προβλεπόμενες τιμές των στοιχείων του ασθενή για τις επόμενες ώρες (χρονοσειρές). Σε αυτό θα συνεισέφερε η χρήση πιο αποτελεσματικών διαγραμμάτων (έναντι των βασισμένων σε apexcharts) τα οποία θα ενσωματώνονται ομαλότερα στο πλαίσιο Vue. Παρά την οργανωμένη προβολή των αποτελεσμάτων, ήταν δύσκολα στη χρήση και προκαλούσαν επιπλοκές κατά την ανάπτυξη των όψεων, ενώ για άγνωστο λόγο δεν αξιοποιούν πλήρως τις δυνατότητες που αναφέρονται στην τεκμηρίωση σε περιβάλλον Vue. Στον ίδιο άξονα βελτίωσης της αισθητικής και της ευχρηστίας θα μπορούσε να κινηθεί και η χρήση εναλλακτικής βιβλιοθήκης για σχεδίαση στοιχείων UI, όπως το Quasar[59] ή το Vue-material[60].

Ενδεχομένως να βελτίωνε την ποιότητα της εφαρμογής και να διευκόλυνε το ιατρικό προσωπικό η εγκατάσταση επικοινωνίας με αισθητήρες που θα κατέγραφαν άμεσα τα στοιχεία των ασθενών, αντί για το τωρινό μοντέλο καταγραφής μέσω γραφικού περιβάλλοντος· αν και κάτι τέτοιο θα απαιτούσε τη συμβατότητα των τύπων δεδομένων που καταγράφουν οι αισθητήρες και των μοντέλων κλάσεων που έχουν οριστεί. Επιπλέον θα έπρεπε τα δεδομένα να καταγράφονται και σε μια κοινή βάση δεδομένων από την οποία μετέπειτα θα έπρεπε να λαμβάνει μέρος διαρκής ανάγνωση των δεδομένων. Εναλλακτικά, οι αισθητήρες θα μπορούσαν να επικοινωνούν με το API της εφαρμογής δικτυακά και να διατηρηθεί το τωρινό μοντέλο του συστήματος σε σημαντικό βαθμό.

Εντούτοις, απαιτούνται αρκετές δοκιμές ώστε να φανεί η αποτελεσματικότητα σε αληθινά περιστατικά και κυρίως, όπως έχει ήδη αναφερθεί, να αυξηθεί το σύνολο δεδομένων για την εκπαίδευση του συστήματος. Εν τέλει, σε τωρινή φάση, προκύπτει μια εφαρμογή η οποία είναι εύχρηστη και μπορεί να πραγματοποιήσει ενδεικτικές προβλέψεις καθιστώντας την ιδανική για αξιολόγηση των ενδείξεών της, ενώ παράλληλα παρέχεται ένα οργανωμένο REST API που μπορεί να χρησιμοποιηθεί βοηθητικά με άλλες υπάρχουσες ιατρικές εφαρμογές.

ΒΙΒΛΙΟΓΡΑΦΙΑ

4	Yunus I, Fasih A, Wang Y. The use of procalcitonin in the determination of severity of sepsis, patient outcomes andinfection characteristics. S1 Table – Data collection sheet. https://doi.org/10.1371/journal.pone.0206527.s001
3	Kumar, V.; Abbas, A.K.; Fausto, N.; et al., eds. (2007). *Robbins Basic Pathology* (8th ed.). Saunders, Elsevier..
26	MacArthur RD, Miller M, Albertson T, et al. Adequacy of early empiric antibiotic treatment and survival in severe sepsis: experience from the MONARCS trial. Clin Infect Dis. 2004;38:284-288. Doi:10.1086/379825.
	Βλαχαβάς Ιωάννης, Κεφαλάς Πέτρος, Βασιλειάδης Νικόλαος, Κόκκορας Φώτης, Σακελλαρίου Ηλίας. *Τεχνητή νοημοσύνη*. Εκδόσεις Πανεπιστημίου Μακεδονίας
	Jiawei Han, Micheline Kamber, Jian Pei. Data mining, concepts and teachings.
32	Λειτουργική μοντελοποίηση από το *Ανάλυση & σχεδιασμός συστημάτων* με τη UML 2.0 των Alan Dennis, Barbara Haley Wixom, David Tegarden. (σ. 244-249).
5	Genga K.R.[a] · Russell J.A.[a, b] Update of Sepsis in the Intensive Care Unit
6	Clinical decision support system to assess the risk of sepsis using Tree Augmented Bayesian networks and electronic medical record data Akash Gupta , Tieming Liu and Scott Shepherd
7	A Data-Driven Approach to Predicting Septic Shock in the Intensive Care Unit Christopher R Yee, Niven R Narain, Viatcheslav R Akmaev and Vijetha Vemulapalli
8	Use of a Neural Network as a Predictive Instrument for Length of Stay in the Intensive Care Unit Following Cardiac Surgery Author links open overlay panelJack V.TuMichael R.J.Guerriere
9	Predicting hospital mortality for patients in the intensive care unit: A comparison of artificial neural networks with logistic

	regression models Clermont, Gilles MD, CM, MSc; Angus, Derek C. MB, ChB, MPH; DiRusso, Stephen M. MD, PhD; Griffin, Martin MS; Linde-Zwirble, Walter T.
10	A comparison of ICU mortality prediction using the APACHE II scoring system and artificial neural networks L. S. S. Wong,J. D. Young
11	Hall MJ, Williams SN, DeFrances CJ, Golosinskiy A: Inpatient care for septicemia or sepsis: a challenge for patients and hospitals. NCHS Data Brief 2011; 62: 1–8.
12	Hoyert DL, Xu J: Deaths: preliminary data for 2011. Natl Vital Stat Rep 2012; 61: 1–51.
13	Navaneelan T, Alam S, Peters PA, Phillips O: Deaths involving sepsis in Canada. Health at a Glance. 2016, http://www.statcan.gc.ca/pub/82-624-x/2016001/article/14308-eng.htm.
14	Perner A, Gordon AC, De Backer D, et al: Sepsis: frontiers in diagnosis, resuscitation and antibiotic therapy. Intensive Care Med 2016; 42: 1958–1969.
15	Chang DW, Tseng CH, Shapiro MF: Rehospitalizations following sepsis: common and costly. Crit Care Med 2015; 43: 2085–2093.
16	Donnelly JP, Hohmann SF, Wang HE: Unplanned readmissions after hospitalization for severe sepsis at academic medical centeraffiliated hospitals. Crit Care Med 2015; 43: 1916–1927.
17	Goodwin AJ, Rice DA, Simpson KN, Ford DW: Frequency, cost, and risk factors of readmissions among severe sepsis survivors. Crit Care Med 2015; 43: 738–746.
18	Jones TK, Fuchs BD, Small DS, et al: Postacute care use and hospital readmission after sepsis. Ann Am Thorac Soc 2015; 12: 904–913.
19	Liu V, Lei X, Prescott HC, et al: Hospital readmission

	and healthcare utilization following sepsis in community settings. J Hosp Med 2014; 9: 502–507.
20	Mayr FB, Talisa VB, Balakumar V, et al: Proportion and cost of unplanned 30-day readmissions after sepsis compared with other medical conditions. JAMA 2017; 317: 530–531.
21	Ortego A, Gaieski DF, Fuchs BD, et al: Hospital-based acute care use in survivors of septic shock. Crit Care Med 2015; 43: 729–737.
22	Prescott HC, Langa KM, Liu V, Escobar GJ, Iwashyna TJ: Increased 1-year healthcare use in survivors of severe sepsis. Am J Respir Crit Care Med 2014; 190: 62–69.
25	Causes and timing of death in critically ill COVID-19 patients Damien Contou, Radj Cally, Florence Sarfati, Paul Desaint, Megan Fraissé & Gaëtan Plantefève
56	Howell MD, Davis AM, ''Management of Sepsis and Septic Shock'', JAMA Clinical Guidelines Synopsis, vol. 317, no. 8, 2017, pp. 847-848.
	https://www.dotconferences.com/2016/12/evan-you-reactivity-in-frontend-javascript-frameworks
27	Παναγιώτης Μπούντρης. Ανάπτυξη αλγορίθμων υπολογιστικής νοημοσύνης και ευφυών συστημάτων εκτίμησης κινδύνου και υποστήριξης κλινικής απόφασης για την έγκαιρη διάγνωση και την εξατομικευμένη διαχείριση γυναικών με ενδοεπιθηλιακές αλλοιώσεις τραχήλου μήτρας. Εθνικό Μετσόβιο Πολυτεχνείο, Αθήνα
35	https://www.youtube.com/watch?v=8FQ4zW_F_Iw
	https://www.atsu.edu/faculty/chamberlain/website/lectures/lecture/sepsis2007.htm
1	https://en.wikipedia.org/wiki/Sepsis
23	https://iepivlepsi.gr/μια-σύγχρονη-ματιά-πάνω-στη-σήψη/
24	https://www.nigms.nih.gov/education/fact-sheets/Pages/sepsis.aspx
28	https://en.wikipedia.org/wiki/Machine_learning
29	https://www.codespeedy.com/how-to-choose-number-of-epochs-to-train-a-neural-network-in-keras/
30	https://www.baeldung.com/cs/neural-net-advantages-disadvantages
31	http://blog.josephwilk.net/ruby/recurrent-neural-networks-in-ruby.html
33	https://blog.back4app.com/backend-

	frameworks/#Why_use_a_backend_framework
34	https://en.wikipedia.org/wiki/Inversion_of_control
	https://rubyonrails.org
	https://el.wikipedia.org/wiki/Model-view-controller
	https://www.brainvire.com/six-benefits-of-using-mvc-model-for-effective-web-application-development/
	https://github.com/ruby/erb
36	https://v2.vuejs.org/v2/guide/
37	https://www.codeinwp.com/blog/angular-vs-vue-vs-react/#learning-curve
38	https://github.com/libfann/fann
39	https://github.com/tangledpath/ruby-fann
40	https://en.wikipedia.org/wiki/Singleton_pattern
41	https://en.wikipedia.org/wiki/Relational_model
42	https://dione.lib.unipi.gr/xmlui/bitstream/handle/unipi/11931/Kokologos_ME1713.pdf?sequence=1&isAllowed=y
43	https://www.json.org/json-en.html
	https://www.mongodb.com/docs/manual/introduction/
44	https://www.mongodb.com/json-and-bson
45	https://developer.mozilla.org/en-US/docs/Web/API/Request/body
56	https://api.rubyonrails.org/classes/ActionController/Parameters.html
57	https://en.wikipedia.org/wiki/Representational_state_transfer
46	https://www.killerphp.com/articles/what-are-orm-frameworks/
47	https://www.treefrogframework.org/en/user-guide/model/object-document-mapping-on-mongodb.html
48	https://guides.rubyonrails.org/getting_started.html#creating-the-blog-application
49	https://ruby-doc.org/core-2.5.0/Module.html
50	https://v2.vuejs.org/v2/guide/components-registration.html
51	https://v2.vuejs.org/v2/guide/components-props.html
52	https://v2.vuejs.org/v2/guide/computed.html
53	https://v2.vuejs.org/v2/guide/instance.html#Instance-Lifecycle-Hooks
54	https://www.npmjs.com
55	https://github.com/BrainJS/brain.js#brainjs
58	https://machinelearningmastery.com/impact-of-dataset-size-on-deep-learning-model-skill-and-performance-estimates/
59	https://quasar.dev
60	https://www.creative-tim.com/vuematerial/