ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ – ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Πρόγραμμα Μεταπτυχιακών Σπουδών «Πληροφορική»

Μεταπτυχιακή Διατριβή

Τίτλος Διατριβής	**Κατασκευή ιστοσελίδας διαχείρισης προσωπικού προφίλ υγείας (Frontend) σε Vue.js και Flask.** **Development of a website for managing personal healthprofiles (Frontend) in Vue.js and Flask.**
Ονοματεπώνυμο Φοιτητή	**Σαχίνη Ειρήνη**
Πατρώνυμο	**Μεχντί**
Αριθμός Μητρώου	**ΜΠΠΛ 20071**
Επιβλέπων	**Αλέπης Ευθύμιος, Καθηγητής**

Ημερομηνία Παράδοσης **Μάιος 2024**

Τριμελής Εξεταστική Επιτροπή

Ευθύμιος Αλέπης Καθηγητής	Μαρία Βίρβου Καθηγήτρια	Κωνσταντίνος Πατσάκης Αν. Καθηγητής

Περιεχόμενα

Πρόλογος

Η παρούσας διατριβής είναι η ολοκληρωμένη κάλυψη ενός πακέτου ανάπτυξης εφαρμογών development delivery, από την αρχική ανάγκη έως την πρόταση βελτιώσεων. Περιλαμβάνει την ανάλυση, τον σχεδιασμό, την υλοποίηση και τον έλεγχο της εφαρμογής. Η επιλογή του κλάδου της υγείας ως βάσης για την ανάπτυξη της εφαρμογής έγινε λόγω της δυνατότητας να ενσωματωθούν νέες τεχνολογίες που θα βοηθήσουν τους χρήστες να διαχειρίζονται καλύτερα την υγεία τους και το ιατρικό προσωπικό να έχει μια καλύτερη πρόσβαση και καταγραφή των ιατρικών ιστορικών των ασθενών.

Στην αρχική φάση, γίνεται μια λεπτομερής ανάλυση του θεωρητικού υποβάθρου που σχετίζεται με την επιλογή μιας αρχιτεκτονικής πολλαπλών επιπέδων (n-tier) και μιας σχεσιακής βάσης δεδομένων. Η εφαρμογή σχεδιάζεται με έμφαση στην ευκολία ανάπτυξης και επέκτασης, χωρίς υψηλό κόστος για τους προγραμματιστές. Στο στάδιο των απαιτήσεων, προστίθεται ένα τμήμα ενός εγγράφου που περιγράφει εμπορικές προδιαγραφές για όλα τα επίπεδα της εφαρμογής και τις υποστηρικτικές υπηρεσίες.

Κατά τη διάρκεια της ανάλυσης, γίνεται ανάλυση της λειτουργίας της εφαρμογής και παρέχονται διαγράμματα ροής και αναλύονται πιθανές προβληματικές καταστάσεις. Παρουσιάζονται επίσης διαγράμματα αλληλεπίδρασης των στοιχείων της εφαρμογής και αναλύονται οι επιλογές τεχνολογίας. Στη συνέχεια, στο στάδιο του σχεδιασμού, παρουσιάζονται οι αρχιτεκτονικές λύσεις και οι προδιαγραφές για τις χρησιμοποιούμενες τεχνολογίες.

Στη φάση της υλοποίησης, γίνεται εστίαση στην υλοποίηση της λύσης και στην επέκταση των λειτουργιών. Αναφέρονται προκλήσεις προγραμματισμού και μηχανισμοί πιστοποίησης και

ΕΙΚΌΝΑ 1 : N-TIER ARCHITECTURE

εξουσιοδότησης, καθώς και η ενσωμάτωση με υπηρεσίες SaaS για την επεξεργασία δεδομένων και τη διαχείριση ειδοποιήσεων.

Κατά το στάδιο του testing, επικεντρώνεται στη δοκιμή της εφαρμογής με διάφορα σενάρια και τύπους δοκιμών, καθώς και στην αυτοματοποίηση αυτών των διαδικασιών.Στις προτάσεις για περαιτέρω βελτίωση, παρουσιάζονται ιδέες για μελλοντικές επεκτάσεις και προτείνονται τρόποι υλοποίησής τους, καθώς και σημειώσεις υλοποίησης που αφορούν στην κλιμάκωση της εφαρμογής.

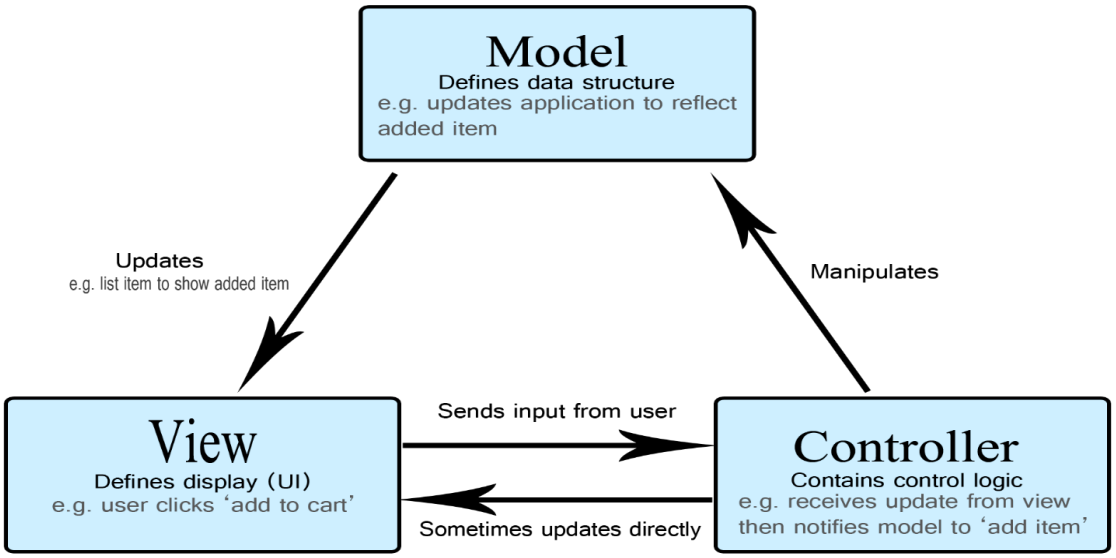
Η αρχιτεκτονική v-επιπέδων είναι μια από τις πιο δημοφιλείς προσεγγίσεις στον σχεδιασμό ενός web app. Χρησιμοποιείται ευρέως σε πολλές εφαρμογές παγκοσμίως και εξακολουθεί να παραμένει σημαντική, ακόμα και με την ανάπτυξη Micro-oriented αρχιτεκτονικών για μεγαλύτερες απαιτήσεις. Αυτή η προσέγγιση, επιτρέπει τη διαχείριση διαφορετικών λειτουργικών μονάδων σε διαφορετικά επίπεδα ή tiers της εφαρμογής.

Με τη διάκριση των λειτουργικών μονάδων σε διαφορετικά tiers, επιτυγχάνουμε προστασία των πόρων του κάθε επιπέδου και επιτρέπουμε την ανεξάρτητη ανάπτυξη και αναβάθμιση του κάθε τμήματος της εφαρμογής, χωρίς να επηρεάζονται τα υπόλοιπα. Επιπλέον, η διαχείριση τυχόν προβλημάτων είναι πιο εύκολη καθώς μπορούμε να εστιάσουμε σε ένα συγκεκριμένο tier χωρίς να επηρεαστεί η λειτουργία των υπολοίπων.

Επιπλέον, η δυνατότητα ανεξάρτητης ανάπτυξης και αναβάθμισης κάθε tier είναι κρίσιμη. Αυτό επιτρέπει την ευελιξία και τη δυνατότητα επέκτασης της εφαρμογής χωρίς να αντιμετωπίζουμε τα ίδια προβλήματα που προκύπτουν σε μονολιθικές αρχιτεκτονικές.

Σε πολλές περιπτώσεις, συνδυάζουμε την αρχιτεκτονική v-επιπέδων με άλλες προσεγγίσεις όπως το Model View Controller (MVC). Αυτή η συνδυαστική προσέγγιση μας επιτρέπει να απολαμβάνουμε τα οφέλη και των δύο μοντέλων, με το κάθε ένα να συμπληρώνει τα πλεονεκτήματα του άλλου. Στην περίπτωσή μας, το Presentation layer υιοθετεί το πρότυπο MVC, ενώ τα υπόλοιπα tiers ακολουθούν την αρχιτεκτονική v-επιπέδων,

παρέχοντας ένα ισχυρό και ευέλικτο πλαίσιο για την ανάπτυξη και τη διαχείριση της εφαρμογής μας.



ΕΙΚΟΝΑ 2: N-TIER ARCHITECTURE

Τα οφέλη που προκύπτουν από τη χρήση μιας αρχιτεκτονικής n-tier είναι πολλά και ποικίλα. Κάποια από τα βασικά οφέλη περιλαμβάνουν:

- **Αυξημένες δυνατότητες διαχείρισης**:

Κάθε επίπεδο μπορεί να διαχειριστεί τις λειτουργίες του ανεξάρτητα, επιτρέποντας την εύκολη προσθήκη λειτουργικούτητας χωρίς να επηρεαστούν τα υπόλοιπα tiers.

- **Ευελιξία**:

Η αρχιτεκτονική n-tier επιτρέπει την εύκολη προσαρμογή και επέκταση της εφαρμογής σε νέες απαιτήσεις και λειτουργικότητες.

- **Επεκτασιμότητα:**

Η δομή των tiers επιτρέπει την προσθήκη νέων στρωμάτων χωρίς να απαιτείται η αναδιάταξη ή η ανασχεδίαση της εφαρμογής.

- **Αυξημένα επίπεδα ασφάλειας:**

Κάθε επίπεδο μπορεί να υιοθετήσει διαφορετικές μεθόδους ασφάλειας, ενισχύοντας την προστασία των δεδομένων και της εφαρμογής συνολικά.

Μέσω της αρχιτεκτονικής n-tier, είναι δυνατή η ολοκλήρωση νέων τεχνολογιών και η προσθήκη νέων στοιχείων στην εφαρμογή χωρίς την ανάγκη για επανασχεδίαση της εφαρμογής ή αναδιάταξη της δομής της. Αυτό καθιστά ευκολότερο το κλιμάκωμα και τη συντήρηση της εφαρμογής.

Επιπλέον, η ασφάλεια ενισχύεται με τη δυνατότητα απομάκρυνσης εμπιστευτικών πληροφοριών από το επίπεδο παρουσίασης.
Άλλα οφέλη περιλαμβάνουν:

- **Πιο αποτελεσματική υλοποίηση:**

Διαφορετικές ομάδες μπορούν να εργαστούν σε διαφορετικά τμήματα της εφαρμογής, επιτρέποντας την πιο αποτελεσματική ανάπτυξη.

- **Εύκολη πρόσθεση νέας λειτουργικότητας:**

Η προσθήκη νέας λειτουργικότητας γίνεται με τη δέσμευση μόνο του αντίστοιχου tier χωρίς την ανάγκη για αναδιάταξη της εφαρμογής.

- **Καλή δυνατότητα επαναχρησιμοποίησης:**

Τα tiers μπορούν να επαναχρησιμοποιηθούν για την υποστήριξη άλλων λύσεων.
Ενώ η παραλλαγή 3-tier είναι η πιο διαδεδομένη, υπάρχουν εφαρμογές με περισσότερα από τρία tiers, όπως υπηρεσίες, επιχειρηματικός τομέας και tier παρουσίασης. Κάθε επίπεδο έχει

τον δικό του ρόλο και συμβάλλει στην ολοκληρωμένη λειτουργία της εφαρμογής. Εν κατακλείδι, η χρήση της αρχιτεκτονικής n-tier είναι οφέλιμη, αλλά πρέπει να λαμβάνεται υπόψη η προσοχή στον αριθμό των tiers για να μην υπερβαίνει τις ανάγκες της εφαρμογής και να μην προκαλεί αδικαιολόγητη πολυπλοκότητα και κόστος.

Οι σχεσιακές βάσεις δεδομένων αποτελούν ένα βασικό συστατικό στην πληροφορική και την επιστήμη των υπολογιστών. Αυτές οργανώνουν την πληροφορία σε πίνακες με σειρές και στήλες. Η αξία τους προέρχεται από την ικανότητά τους να διαχειρίζονται μεγάλο όγκο δεδομένων, να διατηρούν τη συνοχή και την ακεραιότητα των δεδομένων, καθώς και να παρέχουν εύκολη πρόσβαση και ανάκτηση δεδομένων.

Κάθε σχεσιακή βάση δεδομένων αποτελείται από πίνακες, οι οποίοι αποτελούνται από σειρές και στήλες. Κάθε στήλη αντιπροσωπεύει έναν τύπο δεδομένων και κάθε σειρά αντιπροσωπεύει μια εγγραφή. Το πρωτεύον κλειδί αναγνωρίζει μοναδικά κάθε εγγραφή σε έναν πίνακα και διευκολύνει την αναζήτηση και την αναφορά δεδομένων.

Για τη διαχείριση και την αλληλεπίδραση με τα δεδομένα, οι σχεσιακές βάσεις δεδομένων χρησιμοποιούν τη γλώσσα SQL. Η SQL παρέχει εντολές για τη δημιουργία, την τροποποίηση και τη διαγραφή δεδομένων, καθώς και για την επιλογή και την ενημέρωση δεδομένων. Η κανονικοποίηση είναι μια διαδικασία σχεδιασμού βάσης δεδομένων που στοχεύει στη μείωση της επανάληψης δεδομένων και στη διατήρηση της συνέπειας των δεδομένων.

Οι σχεσιακές βάσεις δεδομένων υποστηρίζουν τη διατήρηση ακεραιότητας δεδομένων μέσω διαφόρων περιορισμών, όπως η επιβολή συνθηκών στην ενημέρωση και η δημιουργία συσχετίσεων μεταξύ των πινάκων. Επιπλέον, παρέχουν μηχανισμούς ασφάλειας όπως η διαχείριση δικαιωμάτων πρόσβασης και η κρυπτογράφηση των δεδομένων.

Για την εκτέλεση των ερωτημάτων SQL, οι σχεσιακές βάσεις δεδομένων χρησιμοποιούν διάφορες τεχνικές βελτιστοποίησης της απόδοσης, όπως η χρήση δεικτών και η επιλογή κατάλληλων σχεδιαστικών προτύπων. Η ανάκτηση δεδομένων μπορεί να γίνει με διάφορους τρόπους, συμπεριλαμβανομένης της χρήσης ερωτημάτων SQL και αποθηκευμένων διαδικασιών.
Τέλος, οι σχεσιακές βάσεις δεδομένων συνεργάζονται με άλλα συστήματα πληροφορικής όπως εφαρμογές λογισμικού και διαδικτυακές εφαρμογές, επιτρέποντας την ανταλλαγή δεδομένων και την ολοκλήρωση των λειτουργιών, προωθώντας την ομαλή λειτουργία και την αποτελεσματικότητα των οργανισμών.

Η επικοινωνία μεταξύ εφαρμογών μέσω των διεπαφών εφαρμογών (API) είναι ουσιώδης στη σημερινή πληροφορική. Τα API επιτρέπουν σε εφαρμογές να ανταλλάσσουν δεδομένα με αξιοπιστία και αποτελεσματικότητα. Ένα από τα πιο δημοφιλή είδη API είναι αυτά που συνδέονται με κοινωνικά δίκτυα, όπως το Facebook, το Twitter και το Instagram, επιτρέποντας την πρόσβαση σε δεδομένα χρηστών και τη δημιουργία εφαρμογών που αλληλεπιδρούν με αυτά. Αναδεικνύεται επίσης η αρχιτεκτονική REST, που βασίζεται σε αρχές απλότητας και επεκτασιμότητας, διευκολύνοντας την αλληλεπίδραση μεταξύ εφαρμογών. Αυτό επιτρέπει στις εφαρμογές να επικοινωνούν αποτελεσματικά μεταξύ τους.

Η χρήση του JSON (JavaScript Object Notation) είναι ευρέως διαδεδομένη για την ανταλλαγή δεδομένων μεταξύ εφαρμογών. Αποτελεί έναν ελαφρύ, ανθεκτικό και ευανάγνωστο τρόπο ανταλλαγής δεδομένων μεταξύ διαφορετικών εφαρμογών και συστημάτων. Χρησιμοποιείται ευρέως σε ιστοσελίδες, κινητά τηλέφωνα, διακομιστές και υπηρεσίες cloud, διευκολύνοντας την ανταλλαγή δεδομένων σε πολυδιάστατες εφαρμογές. Τα JSON Web Tokens (JWTs) προσφέρουν έναν ασφαλή τρόπο τεκμηρίωσης για την αυθεντικοποίηση και την εξουσιοδότηση των χρηστών. Αυτά τα tokens περιλαμβάνουν υπογεγραμμένες πληροφορίες που είναι ασφαλείς για την ανταλλαγή μεταξύ διαφορετικών συστημάτων, επιτρέποντας την ασφαλή μεταφορά πληροφοριών μεταξύ διαφορετικών εφαρμογών και συστημάτων.

Η ανάπτυξη προσαρμοσμένων API επιτρέπει τη δημιουργία εφαρμογών που ικανοποιούν συγκεκριμένες ανάγκες, ενώ η ενσωμάτωση των API με άλλες τεχνολογίες είναι ζωτικής σημασίας για τη συνολική λειτουργία του συστήματος. Η προστασία των δεδομένων χρηστών και η συμμόρφωση με κανονιστικά πλαίσια ασφαλείας απαιτούν προσεκτικό σχεδιασμό και υλοποίηση των API.

Περίληψη

Within the framework of this dissertation, we delve into a comprehensive simulation outlining the standardized protocol for crafting novel software solutions tailored for expansive corporate settings. Our approach entails meticulous observation and documentation of business requisites, followed by the meticulous selection of appropriate software architectures. Subsequently, we transition into the high-level design phase, delineating the solution's blueprint, and then descend into the granular details of designing both the solution and its constituent components. Implementation and rigorous testing suites round out our methodology.

Our overarching goal has been the development of an application primed to serve as a Health Assistant/Planner, poised to provide an array of functionalities catering to health monitoring, personalized suggestions, and cost management strategies.

The software architecture we've conceived boasts contemporary features geared towards streamlined testing and deployment processes, thereby enhancing its adaptability across diverse environments, particularly facilitating deployments within cloud infrastructures. Leveraging containerization and OS virtualization technologies, we've engineered a solution that effectively simulates requisite environments. Central to our design philosophy is the emphasis on portability and seamless deployment, enabling its utilization as a backend for both conventional and software environments alike.

To bolster the comprehensive suite of functionalities our application offers, we leverage a myriad of Software-As-A-Service solutions, strategically incorporating them to underpin core functionalities such as notifications, SMS, email services, as well as specialized health calculations and database management essential for delivering the full spectrum of functionality.

Στα πλαίσια της διατριβής, εξερευνούμε μια εκτενή προσομοίωση που περιγράφει το πρωτόκολλο για τη δημιουργία νέων λογισμικών λύσεων προσαρμοσμένων σε εταιρικές ανάγκες. Η προσέγγισή μας περιλαμβάνει την προσεκτική παρατήρηση και την τεκμηρίωση των επιχειρησιακών αναγκών, ακολουθούμενη από την προσεκτική επιλογή κατάλληλων λογισμικών αρχιτεκτονικών. Στη συνέχεια, μεταβαίνουμε στη φάση σχεδιασμού υψηλού επιπέδου, περιγράφοντας την καταγραφή της λύσης, και στη συνέχεια αναλύουμε στις λεπτομέρειες του σχεδιασμού τόσο της λύσης όσο και των συστατικών της. Η υλοποίηση και οι αυστηρές δοκιμές συμπληρώνουν τη μεθοδολογία μας.

Ο κυρίαρχος στόχος μας ήταν η ανάπτυξη μιας εφαρμογής που θα λειτουργεί ως Υγειονομικός Βοηθός/Σχεδιαστής, έτοιμη να προσφέρει μια σειρά λειτουργιών που εξυπηρετούν την παρακολούθηση της υγείας, τις εξατομικευμένες προτάσεις και τις στρατηγικές διαχείρισης κόστους.

Η λογική αρχιτεκτονική που έχουμε σχεδιάσει διαθέτει σύγχρονα χαρακτηριστικά που στοχεύουν στην απλοποίηση των διαδικασιών δοκιμής και αναπτύξεως, βελτιώνοντας έτσι την προσαρμοστικότητά της σε διάφορα περιβάλλοντα, ιδίως επιτρέποντας αναπτύξεις σε cloud. Εκμεταλλευόμενοι τεχνολογίες όπως το containerization και η απεικόνιση λειτουργικού συστήματος, έχουμε δημιουργήσει μια λύση που προσομοιάζει αποτελεσματικά τα απαιτούμενα περιβάλλοντα. Κεντρικό στη φιλοσοφία σχεδιασμού μας είναι η έμφαση στη φορητότητα και την απρόσκοπτη ανάπτυξη, επιτρέποντας τη χρήση της ως πίσω μέρος τόσο για συμβατικά όσο και για λογισμικά περιβάλλοντα.

Το containerization επιτρέπει στους προγραμματιστές να δημιουργούν και να αναπτύσσουν εφαρμογές πιο γρήγορα και με μεγαλύτερη ασφάλεια. Με τις παραδοσιακές μεθόδους, ο κώδικας αναπτύσσεται σε ένα συγκεκριμένο περιβάλλον υπολογισμού, το οποίο, όταν μεταφερθεί σε ένα νέο μέρος, συχνά οδηγεί σε σφάλματα και λάθη.

Για να ενισχύσουμε την πλήρη συλλογή λειτουργιών που προσφέρει η εφαρμογή μας, χρησιμοποιούμε μια ποικιλία λύσεων Λογισμικού-Ως-Υπηρεσίας, ενσωματώνοντάς τις στρατηγικά για να υποστηρίξουν βασικές λειτουργίες όπως οι ειδοποιήσεις, τα SMS, οι υπηρεσίες email, καθώς και εξειδικευμένους υπολογισμούς υγείας και τη διαχείριση βάσεων δεδομένων που είναι απαραίτητοι για την παροχή του πλήρους φάσματος λειτουργικότητας.

Γιατί vue.js, υπέρ και κατά

Τι είναι το Vue.js;

Vue (προφέρεται /vju:/, όπως στο view) είναι ένα προοδευτικό framework JavaScript για τη δημιουργία χρήστη (UI). Το Vue έχει σχεδιαστεί από τη βάση για να είναι εύκολο στην εκμάθηση και την ενσωμάτωση, ενώ επίσης είναι ευέλικτο για να αναπτύσσετε μεγάλες εφαρμογές εάν απαιτείται. Είναι επίσης πολύ ελαφρύ και ευέλικτο να ενσωματωθεί σε οποιοδήποτε είδος εφαρμογής ή περιβάλλον ανάπτυξης.

Πλεονεκτήματα

Η Vue.js είναι μία πρωτοποριακή βιβλιοθήκη JavaScript για την ανάπτυξη του Front-End με ευκολία, αποτελεσματικότητα και ευελιξία. Υπάρχουν πολλοί λόγοι που καθιστούν τη Vue.js μια σωστή επιλογή για προγραμματιστές Front-End. Ας δούμε τα πλεονεκτήματα και τα μειονεκτήματα της:

Πρώτον, ένα από τα μεγαλύτερα πλεονεκτήματα της Vue.js είναι η ευκολία χρήσης της. Η σύνταξη της Vue.js είναι ευανάγνωστη και κατανοητή, επιτρέποντας στους προγραμματιστές να εργάζονται γρήγορα και αποτελεσματικά. Δεύτερον, η Vue.js προσφέρει μια ευέλικτη αρχιτεκτονική που επιτρέπει την οργάνωση του κώδικα σε συστατικά, κάνοντας τη διαχείριση μεγάλων εφαρμογών πιο εύκολη. Τρίτον, η Vue.js προσφέρει ένα εντυπωσιακό οικοσύστημα εργαλείων και βιβλιοθηκών που βοηθούν στην ανάπτυξη και τη συντήρηση των εφαρμογών.

Ένα ακόμη πλεονέκτημα της Vue.js είναι η κοινότητά της. Έχει μια μεγάλη και ενεργή κοινότητα προγραμματιστών που παρέχει υποστήριξη, συμβουλές και πόρους για την επίλυση προβλημάτων και τη βελτίωση των δεξιοτήτων.

ΕΙΚΟΝΑ 3: VUEJS

Μειονεκτήματα

Ωστόσο, υπάρχουν και μερικά μειονεκτήματα που πρέπει να ληφθούν υπόψη. Ένα από αυτά είναι η σχετική νεοφιλικότητα της Vue.js σε σχέση με άλλα πλαίσια εργασίας όπως το React ή το Angular. Αυτό μπορεί να σημαίνει ότι ενδέχεται να υπάρχουν λιγότεροι πόροι και βοήθεια διαθέσιμη σε σύγκριση με άλλες επιλογές.

Επιπλέον, η Vue.js μπορεί να μην είναι η καλύτερη επιλογή για μεγάλες εφαρμογές με πολύπλοκη λογική ή υψηλές απαιτήσεις στην απόδοση. Σε τέτοιες περιπτώσεις, πλαίσια

όπως το Angular με τη δυνατότητά τους για αυτοματοποιημένη βελτιστοποίηση και τη δυνατότητα τους για ευέλικτη διαχείριση κώδικα μπορεί να είναι προτιμότερα.

Συνοψίζοντας, η Vue.js είναι μια εξαιρετική επιλογή για προγραμματιστές Front-End λόγω της ευκολίας χρήσης, της ευελιξίας και της ενεργής κοινότητάς της. Ωστόσο, πρέπει να ληφθούν υπόψη τα μειονεκτήματά της, όπως η σχετική νεοφιλικότητα και οι περιορισμοί σε μεγάλες εφαρμογές.

Η "νεοφιλικότητα" αναφέρεται στο γεγονός ότι η Vue.js είναι σχετικά νέα στον χώρο της ανάπτυξης λογισμικού σε σύγκριση με άλλα πλαίσια εργασίας όπως το Angular ή το React. Συνεπώς, υπάρχει λιγότερη εμπειρία και λιγότερα εργαλεία διαθέσιμα για την ανάπτυξη και τη συντήρηση εφαρμογών βασισμένων σε αυτήν την τεχνολογία.

Nice to know

1. **Progressive Framework**: Vue.js is often referred to as a "progressive framework" because it can be incrementally adopted into existing projects without the need for a complete rewrite.
2. **Lightweight**: Vue.js is very lightweight compared to other front-end frameworks, making it quick to download and easy to integrate into projects.
3. **Reactive Data Binding**: Vue.js uses a reactive data binding system which makes it easy to manage and synchronize changes between the model and the view.
4. **Virtual DOM**: Similar to React, Vue.js uses a virtual DOM which improves performance by only updating the parts of the DOM that have changed.
5. **Component-Based Architecture**: Vue.js encourages the use of components, allowing developers to build large-scale applications by breaking them down into smaller, reusable pieces.
6. **Versatile Templating**: Vue.js provides a simple and intuitive templating syntax that allows developers to write HTML-like code with additional features like directives and binding expressions.
7. **Official CLI Tool**: Vue.js comes with an official command-line interface (CLI) tool that streamlines project setup, development, and deployment processes.
8. **Growing Community**: Vue.js has a rapidly growing community of developers who contribute to its ecosystem by creating plugins, libraries, and tools, as well as providing support and sharing knowledge through forums, meetups, and conferences.
9. **Official Documentation**: Vue.js boasts comprehensive and well-organized official documentation, making it easy for developers to learn and reference while building applications.
10. **Adopted by Big Names**: Vue.js is used by many well-known companies and organizations, including Alibaba, Xiaomi, GitLab, Nintendo, and Adobe. Its popularity continues to rise in both the startup and enterprise sectors.

Docker & docker_compose

Εισαγωγή στο Docker:

Το Docker είναι μια πλατφόρμα λογισμικού που επιτρέπει την εύκολη δημιουργία, την παράδοση και την εκτέλεση εφαρμογών χρησιμοποιώντας τεχνολογίες ελαφρού virtualization. Με το Docker, μπορείτε να εκτελέσετε εφαρμογές σε οποιοδήποτε περιβάλλον χωρίς να χρειάζεται να ανησυχείτε για τις διαφορές στη ρύθμιση ή τις εξαρτήσεις.

Πλεονεκτήματα του Docker:

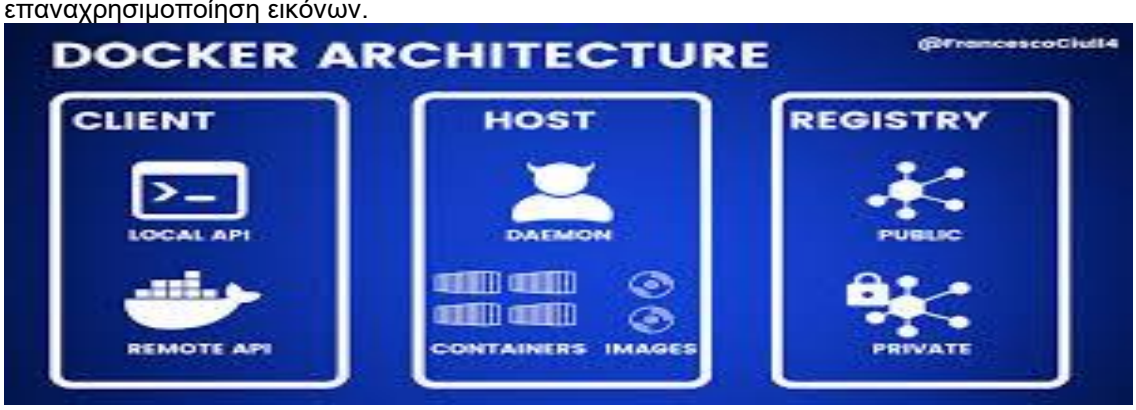
Ένα από τα κύρια πλεονεκτήματα του Docker είναι η απομόνωση των εφαρμογών σε ελαφριές εικόνες, οι οποίες είναι φορητές και ανεξάρτητες από το περιβάλλον εκτέλεσης.

Διαδικασία Εργασίας με το Docker:

Η διαδικασία εργασίας με το Docker περιλαμβάνει τη δημιουργία των Dockerfile, των αρχείων που περιγράφουν το περιβάλλον εκτέλεσης της εφαρμογής, και την κατασκευή των εικόνων Docker.

Διαχείριση των Docker images:

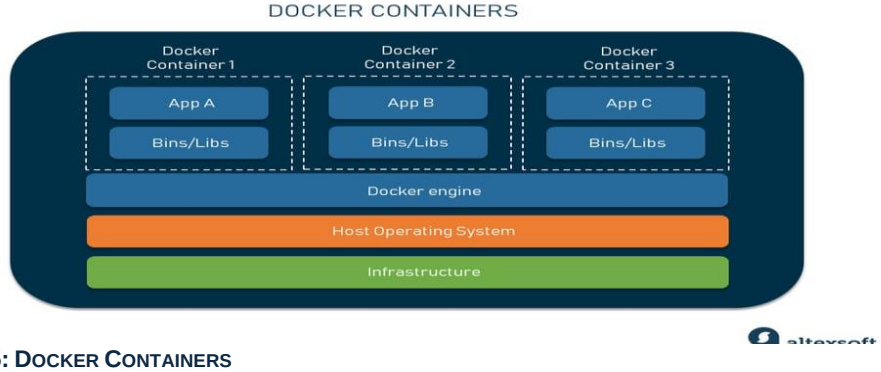
Οι εικόνες Docker μπορούν να αναπτυχθούν τοπικά ή να αποθηκευτούν σε κοινόχρηστα αποθετήρια, όπως το Docker Hub. Αυτό επιτρέπει την εύκολη ανταλλαγή και την επαναχρησιμοποίηση εικόνων.



Εικόνα 4: DOCKER ARCHITECTURE

Εισαγωγή στο Docker Compose:

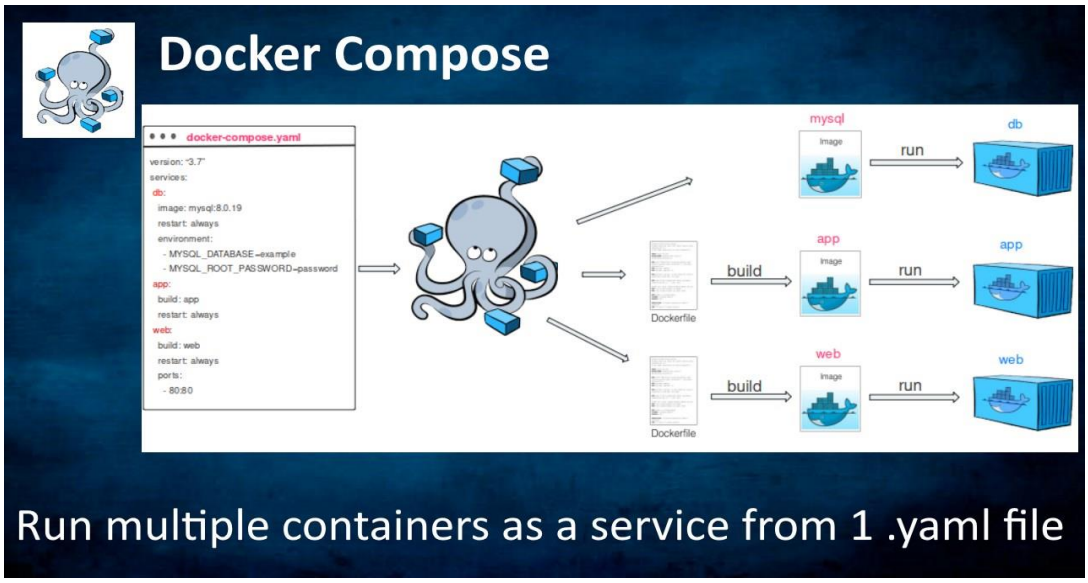
Το Docker Compose είναι ένα εργαλείο που επιτρέπει τη διαχείριση πολλαπλών εφαρμογών με χρήση ενός αρχείου διαμόρφωσης YAML. Με το Docker Compose, μπορείτε να ορίσετε τις ρυθμίσεις του δικτύου, τους όγκους δεδομένων και άλλες παραμέτρους για τις εφαρμογές σας.



Εικόνα 5: Docker Containers

Χρήση του Docker Compose:

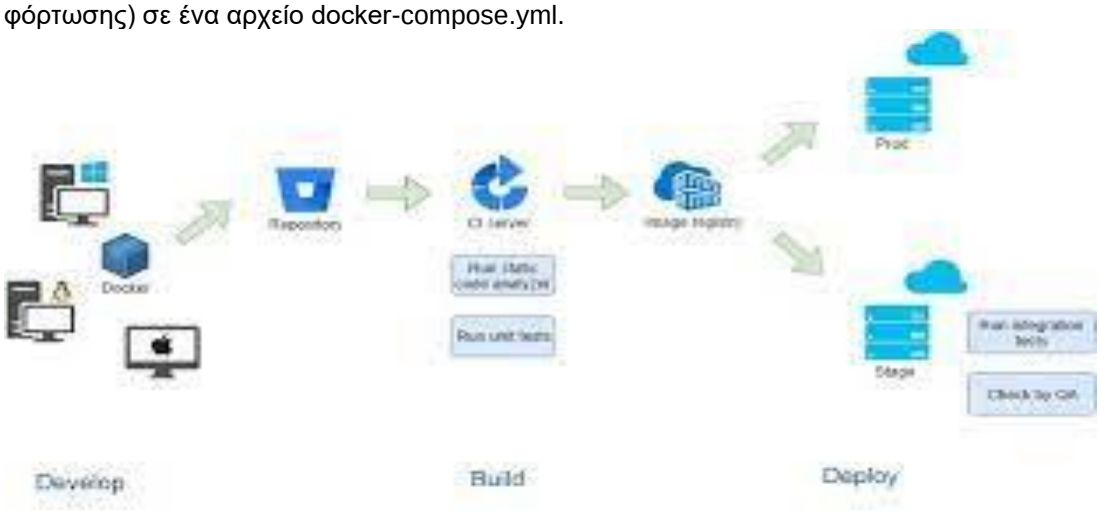
Η χρήση του Docker Compose απλοποιεί τη διαχείριση πολλαπλών εφαρμογών, καθώς μπορείτε να ξεκινήσετε, να σταματήσετε και να διαμορφώσετε ολόκληρο το περιβάλλον ανάπτυξής σας με ένα απλό αρχείο διαμόρφωσης.



Εικονα 6: Docker Compose

Διαχείριση Πολυεπίπεδων Εφαρμογών:

Ένα από τα βασικά πλεονεκτήματα του Docker Compose είναι η δυνατότητα διαχείρισης πολυεπίπεδων εφαρμογών. Μπορείτε να ορίσετε διάφορα τμήματα της εφαρμογής σας (όπως οι υπηρεσίες βάσεων δεδομένων, οι εξυπηρετητές εφαρμογών και οι διακομιστές φόρτωσης) σε ένα αρχείο docker-compose.yml.



Εικονα 7: Develop-build-deploy

Επεκτασιμότητα του Περιβάλλοντος Ανάπτυξης:

Ένα από τα κύρια χαρακτηριστικά του Docker Compose είναι η επεκτασιμότητά του.

Μπορείτε να προσθέσετε νέες υπηρεσίες και να προσαρμόσετε το περιβάλλον ανάπτυξής

σας με ελάχιστη προσπάθεια, επιτρέποντάς σας να ανταπεξέλθετε στις ανάγκες της εφαρμογής σας χωρίς να χρειάζεται να αναδιαμορφώσετε ολόκληρο το περιβάλλον.

Αυτοματοποίηση Διαδικασιών:

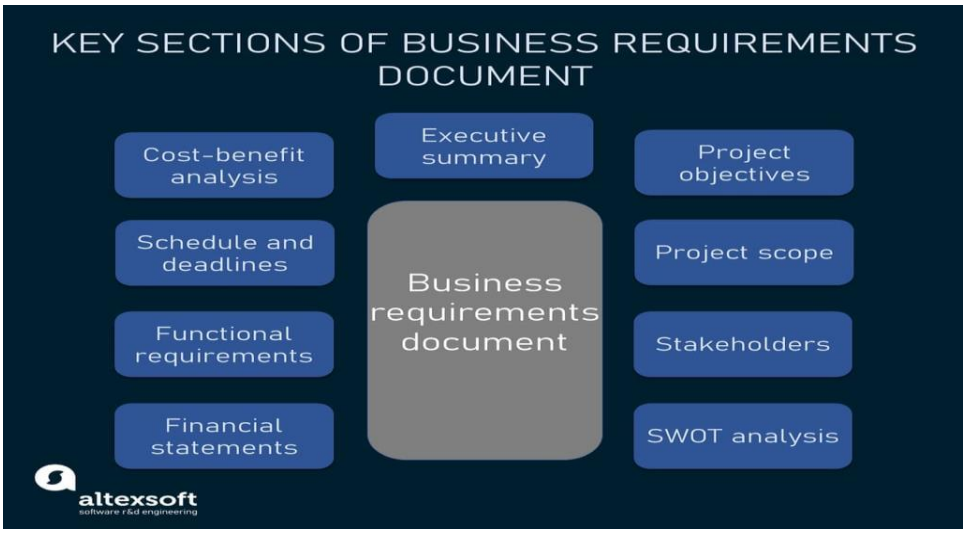
Οι δυνατότητες αυτοματοποίησης του Docker Compose επιτρέπουν τη δημιουργία αυτόματων διαδικασιών για τη διαχείριση και την ανάπτυξη των εφαρμογών σας. Μπορείτε να ορίσετε σενάρια εκτέλεσης που ανταποκρίνονται στις ανάγκες του κάθε περιβάλλοντος ανάπτυξης σας, εξοικονομώντας έτσι χρόνο και πόρους.

Συνεργασία και Ανάπτυξη Ομάδας:

Ο Docker Compose διευκολύνει τη συνεργασία και την ανάπτυξη ομάδας, καθώς επιτρέπει στους προγραμματιστές να ορίζουν και να μοιράζονται το περιβάλλον ανάπτυξής τους με άλλα μέλη της ομάδας. Αυτό διευκολύνει την ανταλλαγή γνώσεων και επιτρέπει την ομαδική ανάπτυξη εφαρμογών με αποτελεσματικό τρόπο.

Ανάπτυξη του έργου

Η εμπορική ανάγκη ενός έργου αποτυπώνεται στο εγγραφό BRD (Business Requirement Document), που περιλαμβάνει πλήρη περιγραφή της ανάγκης, των παραδοτέων και των κριτηρίων αποδοχής, καθώς και των KPIs για την αξιολόγησή τους. Αποτελεί το βασικό έγγραφο εκκίνησης για κάθε σοβαρό έργο πληροφορικής και αποτελεί τη βάση για τους αναλυτές επιχειρήσεων και τους αρχιτέκτονες λύσεων για τη μετατροπή της εμπορικής ανάγκης σε τεχνικές προδιαγραφές.



Για το εν λόγω έργο, ο στόχος ήταν:

ΕΙΚΌΝΑ 8: BUSINESS REQUIREMENTS DOCUMENT

- Η δημιουργία ενός σύγχρονου πλαισίου εργασίας για έναν ιστότοπο που εστιάζει στην υγεία.

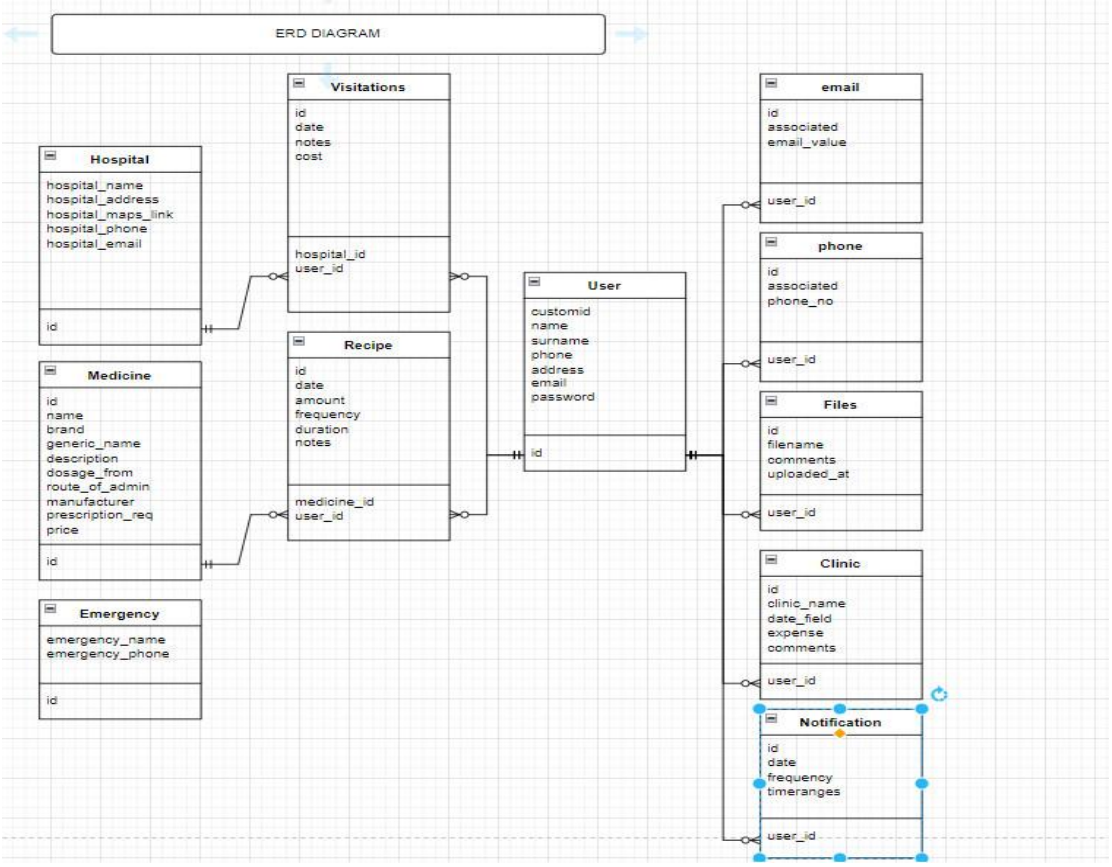
- Η προσθήκη λειτουργιών για την υποστήριξη μεταφόρτωσης/λήψης ιατρικών εγγράφων.
- Η προσθήκη λειτουργιών για την καταγραφή ιατρικών επισκέψεων σε νοσοκομεία, κλινικές και προσωπικούς ιατρούς.
- Η προσθήκη λειτουργιών για την παρακολούθηση συνταγών από τους χρήστες.
- Η δημιουργία λειτουργιών για αυτόματες ειδοποιήσεις που σχετίζονται με θέματα υγείας.
- Η παροχή λειτουργιών διαχείρισης λογαριασμών και σχετικών λειτουργιών.
- Η προσφορά λειτουργιών για τον υπολογισμό και την πρόταση διατροφής βάσει θερμιδικής αξίας και επιπέδου άσκησης.
- Η προσφορά λειτουργιών για τον υπολογισμό του δείκτη μάζας σώματος βάσει των παραμέτρων του χρήστη.
- Η ανάπτυξη ενός συστήματος διαχείρισης λογαριασμών και σχετικών λειτουργιών.
- Όσον αφορά τις τεχνικές απαιτήσεις, αποφασίστηκε:
- Η υλοποίηση της λύσης σε αρχιτεκτονική N-tier με ξεχωριστά επίπεδα backend, frontend και logic.
- Η δημιουργία μιας διεπαφής με υπηρεσία SaaS για την αποστολή ειδοποιήσεων (sms/email).
- Η χρήση ενός περιβάλλοντος περιέκτη (container-driven pipeline) με χρήση Docker και Docker Compose.
- Η υλοποίηση κάθε συστατικού της εφαρμογής ως ξεχωριστού Docker container για ανεξάρτητο deploy.
- Η υποστήριξη διαχείρισης χρηστών, καταγραφής δραστηριοτήτων, και καταγραφής συμβάντων (logging), καθώς και η χρήση μοντέρνου τρόπου πιστοποίησης (jwt token).
- Η εφαρμογή ελέγχθηκε στο Docker Desktop (UX/Windows).

Επιπλέον, υπάρχουν τεχνικές απαιτήσεις σχετικά με τη δομή και την υλοποίηση της εφαρμογής, καθώς και το deploy pipeline.
Αυτές περιλαμβάνουν:

I. Τη διάταξη των επιπέδων της εφαρμογής σε ξεχωριστά Docker files που συντονίζονται από το Docker Compose.
II. Τη δυνατότητα ενσωμάτωσης του Docker Compose σε ένα Kubernetes cluster για scalable deployment (horizontal/vertical scaling).
III. Τη δυνατότητα ενσωμάτωσης του Docker Compose σε ένα Kubernetes cluster με σχήμα υψηλής διαθεσιμότητας ή την ενσωμάτωση των Docker files σε διακομιστές VM με Load Balancer.
IV. Τη χρήση της Vue/Vite για το προσωπικό λογισμικό (front-end) και τη χρήση της Python/Flask για το λογισμικό λογικής/αποθήκευσης (back-end), σε συνδυασμό με μια βάση δεδομένων SQLite για ευελιξία στην επέκταση στα δεδομένα για κινητές συσκευές.

Στο βασικό μας σχήμα βάσης, ο πίνακας των χρηστών λειτουργεί ως κεντρικός πυλώνας οργάνωσης, ο οποίος συνδέεται με άλλες οντότητες μέσω σχέσεων. Συνήθως, η σύνδεση γίνεται μέσω εξωτερικών κλειδιών (foreign key), αλλά υπάρχουν και πίνακες πολλαπλών συσχετίσεων που απαιτούν join με πολλαπλά κλειδιά. Η επιλογή της σχεσιακής βάσης δεδομένων έγινε λόγω της ανάγκης για γρήγορη πρόσβαση και υψηλή απόδοση στις συγκεκριμένες συνδέσεις. Οι σχεσιακές βάσεις έχουν εξαιρετική απόδοση στη συνένωση δεδομένων, και με την κατάλληλη ευρετηρίαση των πινάκων μπορούν να παρέχουν εξαιρετικές επιδόσεις. Επιπλέον, επιβάλλουν ένα σύνολο κανόνων που εξασφαλίζουν την ακεραιότητα των δεδομένων μας και προστατεύουν από διπλοεγγραφές, κακή χρήση τύπων δεδομένων και σφάλματα καταχώρισης δεδομένων.

Database ERD diagram



ΕΙΚΟΝΑ 9: ER-DIAGRAM

Στο διάγραμμα ERD φαίνεται ότι έχουν περιγραφεί διάφορες σχέσεις όπως παρακάτω:
Χρήστες -> Ειδοποιήσεις (Ένα προς Πολλά) – Σύνδεση με FK(user_id)
Χρήστες -> Κλινικές (Ένα προς Πολλά) – Σύνδεση με FK(user_id)
Χρήστες -> Αρχεία (Ένα προς Πολλά) – Σύνδεση με FK(user_id)
Χρήστες -> Τηλέφωνα (Ένα προς Πολλά) – Σύνδεση με FK(user_id)
Χρήστες -> Email (Ένα προς Πολλά) – Σύνδεση με FK(user_id)
Χρήστες -> Επισκέψεις (Ένα προς Πολλά) – Σύνδεση με FK(user_id)
Νοσοκομείο -> Επισκέψεις (Ένα προς Πολλά) – Σύνδεση με FK(hospital_id)
Χρήστες -> Νοσοκομεία (Πολλαπλή προς Πολλαπλή) – Σύνδεση με FK(hospital_id, user_id)
Χρήστες -> Συνταγές (Ένα προς Πολλά) – Σύνδεση με FK(user_id)
Φάρμακο -> Συνταγές (Ένα προς Πολλά) – Σύνδεση με FK(medicine_id)
Συνταγές -> Χρήστες (Πολλαπλή προς Πολλαπλή) – Σύνδεση με FK(medicine_id, user_id)

Βάσεις δεδομένων – SQLite

Οι σχεσιακές βάσεις δεδομένων αποτελούν ένα θεμέλιο στον κόσμο της πληροφορικής και των εφαρμογών λογισμικού. Αυτές οι βάσεις δεδομένων οργανώνουν τα δεδομένα σε πίνακες με στήλες και γραμμές, επιτρέποντας την εύκολη ανάκτηση, ενημέρωση και διαγραφή των δεδομένων.

Η δομή των σχεσιακών βάσεων δεδομένων είναι προσεκτικά σχεδιασμένη, με κάθε πίνακα να έχει ένα κύριο κλειδί που το καθιστά μοναδικά αναγνωρίσιμο. Αυτό επιτρέπει τη διασφάλιση της ακεραιότητας των δεδομένων και την αποφυγή της αντιφάσεων.

Η χρήση των σχεσιακών βάσεων δεδομένων επιτρέπει τη διευκόλυνση της ανάπτυξης εφαρμογών, καθώς παρέχουν ένα σταθερό και οργανωμένο περιβάλλον για την αποθήκευση και την επεξεργασία των δεδομένων.

Ωστόσο, ορισμένες φορές η σχεδίαση και η διαχείριση των σχεσιακών βάσεων δεδομένων μπορεί να είναι περίπλοκη και χρονοβόρα. Η απαίτηση για προσεκτική σχεδίαση των πινάκων και των σχέσεών τους μπορεί να απαιτεί εξειδικευμένες γνώσεις και προσοχή στη λεπτομέρεια.

Παρ' όλα αυτά, οι σχεσιακές βάσεις δεδομένων παραμένουν μια αξιόπιστη και αποτελεσματική λύση για την οργάνωση και τη διαχείριση μεγάλων όγκων δεδομένων σε ποικίλες εφαρμογές, από επιχειρήσεις έως και ιστοσελίδες κοινωνικής δικτύωσης.

Οι σχεσιακές βάσεις δεδομένων προσφέρουν πλειάδα πλεονεκτημάτων στον κόσμο της πληροφορικής και των εφαρμογών λογισμικού. Ένα από τα κύρια πλεονεκτήματα είναι η αξιόπιστη διαχείριση των δεδομένων, καθώς η δομή τους επιτρέπει την αποτελεσματική οργάνωση και ανάκτησή τους.

Μια άλλη σημαντική πλευρά των σχεσιακών βάσεων δεδομένων είναι η ασφάλεια. Οι περισσότερες σύγχρονες σχεσιακές βάσεις δεδομένων παρέχουν προηγμένα μέσα προστασίας, συμπεριλαμβανομένων της κρυπτογράφησης και του συστήματος ελέγχου πρόσβασης.

Ωστόσο, ορισμένα μειονεκτήματα συνδέονται επίσης με τις σχεσιακές βάσεις δεδομένων. Ένα από αυτά είναι η πολυπλοκότητα στη σχεδίαση και τη διαχείριση των σχέσεων μεταξύ των πινάκων, ιδιαίτερα σε πολύπλοκα συστήματα.

Ένα άλλο μειονέκτημα είναι η απόδοση σε κάποιες περιπτώσεις. Καθώς οι σχεσιακές βάσεις δεδομένων αυξάνουν σε μέγεθος, μπορεί να παρουσιαστούν προβλήματα απόδοσης λόγω του μεγάλου όγκου δεδομένων και των συνεχών λειτουργιών ανάκτησης και ενημέρωσης.

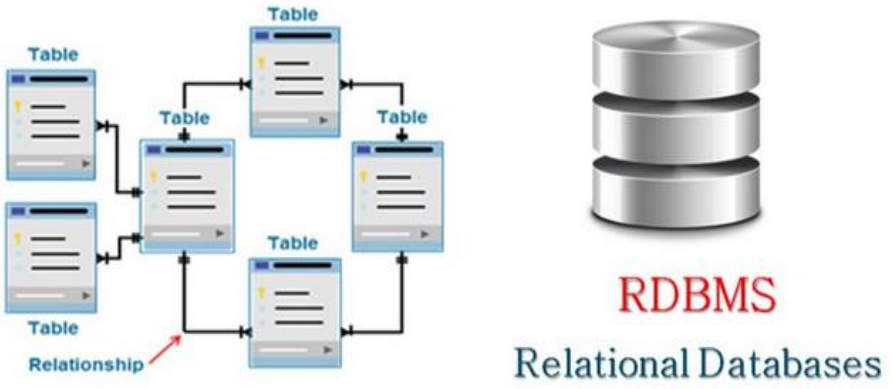
Οι σχεσιακές βάσεις δεδομένων αντιπροσωπεύουν ένα θεμελιώδες μέρος της πληροφορικής και των υπολογιστών. Αυτές οι βάσεις οργανώνουν τα δεδομένα σε πίνακες που περιλαμβάνουν γραμμές και στήλες. Η σημαντικότητα τους πηγάζει από τη δυνατότητα να διαχειρίζονται με επιτυχία μεγάλους όγκους πληροφοριών, να διασφαλίζουν την ακεραιότητα και συνέπεια των δεδομένων, και να προσφέρουν εύκολη πρόσβαση και ανάκτηση των πληροφοριών.

ΕΙΚΌΝΑ 10: DATABASE IMAGE

Κάθε σχεσιακή βάση δεδομένων αποτελείται από ένα σύνολο πινάκων, ο καθένας αποτελούμενος από σειρές και στήλες. Σε κάθε στήλη αντιστοιχεί ένας τύπος δεδομένων, ενώ κάθε σειρά αντιπροσωπεύει ένα εγγραφή. Το πρωτεύον κλειδί αναγνωρίζει μοναδικά μια εγγραφή σε έναν πίνακα, επιτρέποντας την εύρεση και αναφορά της.

Οι σχεσιακές βάσεις δεδομένων χρησιμοποιούν την SQL (Structured Query Language) για τη διαχείριση και την επικοινωνία με τα δεδομένα. Η SQL παρέχει εντολές για τη δημιουργία, την τροποποίηση, και τη διαγραφή δεδομένων, καθώς επίσης και για την επιλογή και την ανανέωση τους.

Η διαδικασία της κανονικοποίησης στοχεύει στη μείωση της επανάληψης δεδομένων και στη διατήρηση της συνέπειας των δεδομένων. Αυτό βοηθά στην αποφυγή προβλημάτων που προκύπτουν από την αντίφαση δεδομένων και βελτιστοποιεί την απόδοση της βάσης δεδομένων. Επιπλέον, οι σχεσιακές βάσεις δεδομένων υποστηρίζουν τη διατήρηση ακεραιότητας δεδομένων μέσω των περιορισμών που μπορούν να οριστούν στους πίνακες, όπως η περιορισμένη εισαγωγή τιμών, οι συνθήκες ενημέρωσης και η δημιουργία συσχετίσεων μεταξύ πινάκων.



ΕΙΚΟΝΑ 11:RELATIONAL DATABASES

Είναι αναγκαίο να διατηρείται η ακεραιότητα και η εμπιστοσύνη στη βάση δεδομένων. Οι σχεσιακές βάσεις δεδομένων παρέχουν μηχανισμούς ασφαλείας, όπως η διαχείριση των δικαιωμάτων πρόσβασης, ο ορισμός ρόλων χρηστών και η κρυπτογράφηση των δεδομένων.

Για τη βελτιστοποίηση της απόδοσης κατά την εκτέλεση των ερωτημάτων SQL, οι σχεσιακές βάσεις δεδομένων χρησιμοποιούν διάφορες τεχνικές. Αυτές περιλαμβάνουν τη χρήση δεικτών για την επιτάχυνση της αναζήτησης, την επιλογή κατάλληλων σχεδιαστικών προτύπων και τη βελτιστοποίηση των ερωτημάτων SQL μέσω εργαλείων εξαγωγής εκτέλεσης και στατιστικών.

Η ανάκτηση δεδομένων από μια σχεσιακή βάση δεδομένων μπορεί να γίνει με διάφορους τρόπους, όπως μέσω ερωτημάτων SQL, αποθηκευμένων διαδικασιών, ή μέσω εφαρμογών που χρησιμοποιούν το API της βάσης δεδομένων.

SQLite

SQLite είναι μια ελαφριά, αυτόνομη, αυτό-επαρκής βάση δεδομένων που αποθηκεύεται σε ένα μόνο αρχείο. Αυτό την καθιστά ιδανική για ενσωμάτωση σε εφαρμογές που χρειάζονται μια απλή, αξιόπιστη βάση δεδομένων χωρίς την ανάγκη ενός ξεχωριστού διακομιστή βάσης δεδομένων.

Ένα από τα μεγάλα πλεονεκτήματα της SQLite είναι η φορητότητα. Το αρχείο βάσης δεδομένων SQLite μπορεί να μεταφερθεί εύκολα από μια πλατφόρμα σε μια άλλη χωρίς προβλήματα συμβατότητας.

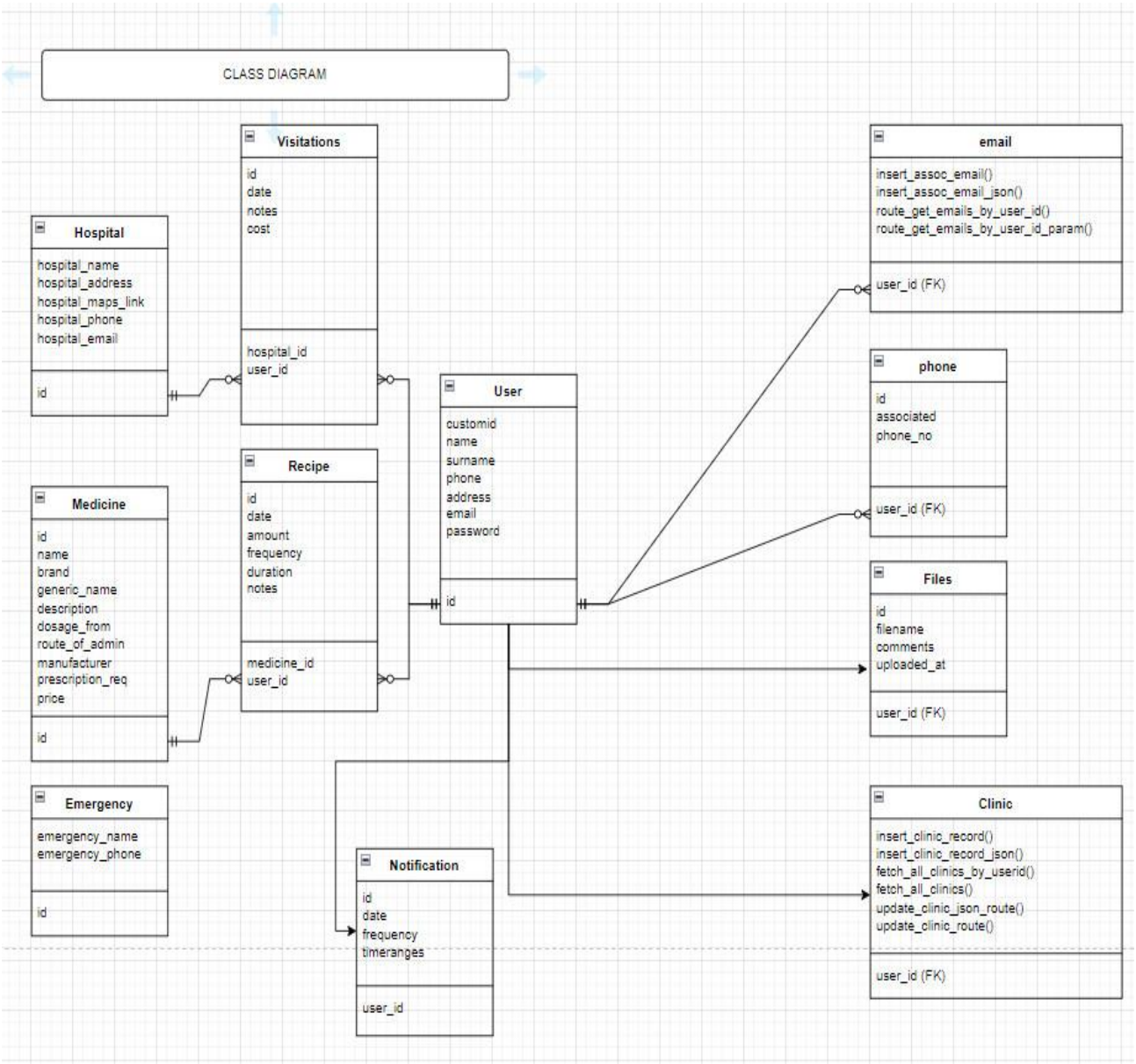
Επίσης, η SQLite είναι αξιόπιστη και εύκολη στη χρήση. Η βάση δεδομένων SQLite απαιτεί λίγη ή καθόλου συντήρηση και διαχείριση, καθιστώντας την ιδανική για μικρές ή μεσαίου μεγέθους εφαρμογές.

ΕΙΚΟΝΑ 12:SQLITE

Ένα άλλο πλεονέκτημα είναι η υψηλή απόδοση. Η SQLite είναι γρήγορη στην ανάκτηση και ενημέρωση δεδομένων, ειδικά σε μικρές έως μέτριες εφαρμογές. Επιπλέον, η SQLite είναι δωρεάν και ανοικτού κώδικα, διαθέσιμη υπό την άδεια διανομής του κοινού και υποστηρίζεται από μια ενεργή κοινότητα προγραμματιστών.

Τέλος, η SQLite είναι ευέλικτη και υποστηρίζει πολλές γλώσσες προγραμματισμού, συμπεριλαμβανομένων των C/C++, Python, Java και πολλών άλλων. Αυτό την καθιστά ιδανική για μια ευρεία γκάμα εφαρμογών και περιβαλλόντων ανάπτυξης.

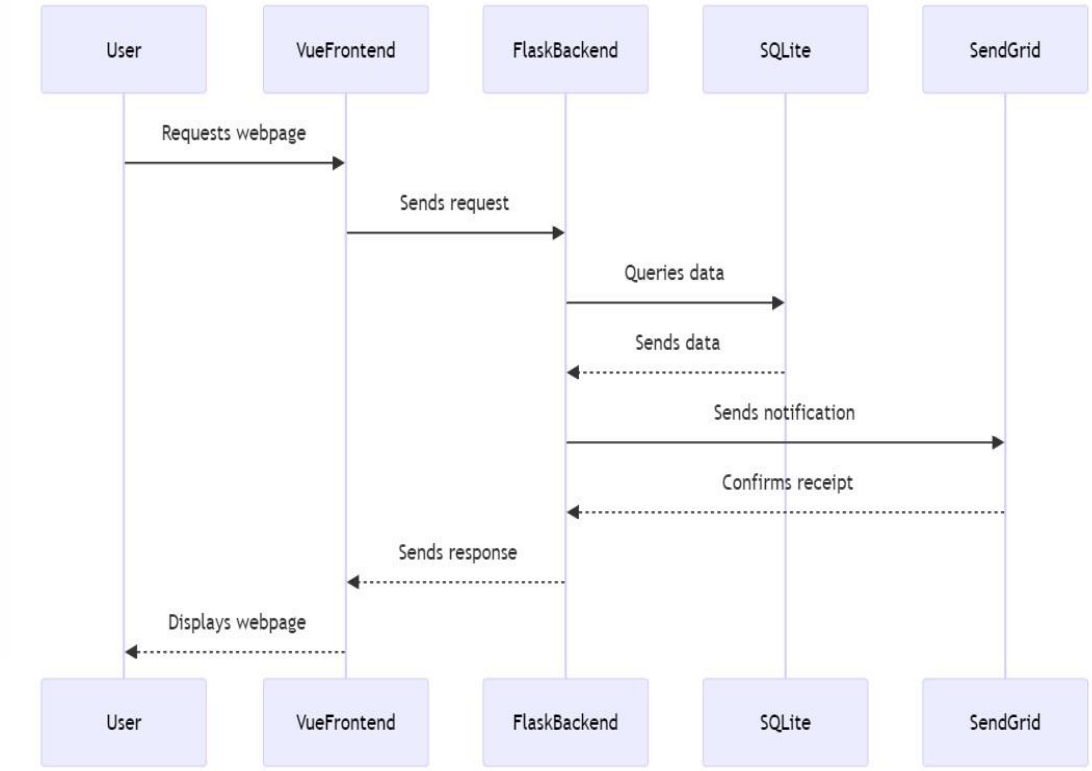
Database Class Diagram



ΕΙΚΟΝΑ 13: CLASS DIAGRAM

Στο παραπάνω διάγραμμα παρουσιάζονται επιγραμματικά μερικές από τις μεθόδους που διατίθενται από κάθε οντότητα της βάσης δεδομένων μας στο επίπεδο του μοντέλου κώδικα. Οι περιορισμένες εμφανίσεις οφείλονται στον περιορισμένο χώρο που διαθέτουμε στο διάγραμμα.

Data Flow Sequence Diagram



ΕΙΚΟΝΑ 14: DATA FLOW SEQUENCE DIAGRAM

Στο παραπάνω διάγραμμα ακολουθείται ένα ενδεικτικό μοτίβο λειτουργίας της εφαρμογής μας, με πλήρη ανάλυση των αντίστοιχων διεπαφών:

- Αρχικά, ένας χρήστης αιτείται πρόσβαση στην ιστοσελίδα της εφαρμογής.
- Το αίτημα διαχειρίζεται από το Vue Front-end, το οποίο το μεταβιβάζει στον αντίστοιχο ελεγκτή του front-end.
- Ο ελεγκτής του front-end μεταβιβάζει το αίτημα στο επόμενο επίπεδο, το Flask Backend.
- Το Flask Backend αναλαμβάνει το αίτημα και επιλέγει τον κατάλληλο ελεγκτή με βάση τον ρόλο του, στη συνέχεια το προωθεί στο επίπεδο δεδομένων.
- Το επίπεδο δεδομένων επιλέγει το κατάλληλο ερώτημα βάσης δεδομένων για την ανάκτηση των σχετικών πληροφοριών και τη μορφοποίησή τους για το επόμενο επίπεδο.
- Σε περίπτωση ανάγκης για ειδοποιήσεις, το επίπεδο επιχειρησιακής λογικής αποστέλλει αίτημα προς την πλατφόρμα SaaS του Sendgrid, με κατάλληλο πρότυπο μηνύματος email.
- Η απάντηση προωθείται στο Flask Backend από το επίπεδο δεδομένων και στη συνέχεια στο Vue Front-end, το οποίο είναι υπεύθυνο για την απεικόνιση των πληροφοριών στο front-end.

Git Repository

Ένα Git repository είναι ένας τόπος όπου αποθηκεύετε τον κώδικά σας και διαχειρίζεστε την ανάπτυξή του. Ας δούμε πώς λειτουργεί: Όταν αποθηκεύετε τον κώδικά σας σε ένα Git, ο κώδικας αυτός παρακολουθείται και καταγράφονται οι αλλαγές που κάνετε. Αυτό σημαίνει ότι μπορείτε να πάρετε μια εικόνα ιστορικού κάθε φορά που αλλάζετε τον κώδικά σας.

Τώρα, γιατί θα έπρεπε να χρησιμοποιήσετε το Git; Υπάρχουν αρκετοί λόγοι. Καταρχάς, σας επιτρέπει να δουλεύετε εύκολα με άλλους ανθρώπους σε έναν κοινό κώδικα. Μπορείτε να συνεργαστείτε πάνω σε ένα έργο και να βλέπετε τις αλλαγές που κάνουν οι άλλοι και να ενσωματώνετε τις δικές σας αλλαγές με ασφάλεια.

Επιπλέον, το Git σας επιτρέπει να δημιουργήσετε διακλαδώσεις (branches) του κώδικα σας. Αυτό σας επιτρέπει να εργαστείτε σε διαφορετικά χαρακτηριστικά ή διορθώσεις χωρίς να επηρεάζετε τον κώδικα στο κύριο κλαδί (main branch). Όταν είστε έτοιμοι, μπορείτε να συγχωνεύσετε (merge) τις αλλαγές σας με το κύριο κλαδί.

Τέλος, το Git προσφέρει έναν τρόπο εύκολου αναίρεσης αλλαγών. Αν κάνετε κάτι λάθος, μπορείτε εύκολα να επιστρέψετε σε προηγούμενες εκδόσεις του κώδικά σας.

Συνοψίζοντας, το Git είναι ένα ισχυρό εργαλείο για τη διαχείριση του κώδικά σας, επιτρέποντάς σας να συνεργαστείτε εύκολα με άλλους, να δημιουργείτε διακλαδώσεις για την ανάπτυξη χαρακτηριστικών και να διαχειρίζεστε αποτελεσματικά τις αλλαγές σας.

Διαδικασία ανάπτυξης

Το Docker αποτελεί μια πλατφόρμα ανοιχτού κώδικα που προσφέρει μια τεχνολογία λογισμικού για τη δημιουργία, ανάπτυξη και εκτέλεση εφαρμογών μέσα σε περιβάλλοντα που ονομάζονται "containers" ή "images". Αυτή η προσέγγιση επιτρέπει την εκτέλεση εφαρμογών σε απομονωμένα περιβάλλοντα, χωρίς να επηρεάζεται το σύνολο του συστήματος.

Χρηση του DOCKER

Το Docker επιτρέπει στους αρχιτέκτονες και τους προγραμματιστές να δημιουργούν, να τροποποιούν και να διαμοιράζονται εφαρμογές χρησιμοποιώντας την τεχνολογία containerization. Αυτή η προσέγγιση παρέχει μια σειρά από πλεονεκτήματα για τους χρήστες.

Πλεονεκτήματα του DOCKER

• **Απομόνωση**: Τα containers προσφέρουν απομόνωση για τις εφαρμογές, επιτρέποντάς τους να λειτουργούν ανεξάρτητα χωρίς να επηρεάζουν άλλες εφαρμογές ή το σύνολο του συστήματος.
• **Ελαφρότητα**: Οι containers είναι ελαφριές και μπορούν να ξεκινήσουν γρήγορα, χρησιμοποιώντας λιγότερους πόρους σε σύγκριση με τις παραδοσιακές μεθόδους ανάπτυξης εφαρμογών.
• **Ευελιξία**: Το Docker παρέχει ευελιξία στην ανάπτυξη και την εκτέλεση εφαρμογών, καθώς οι containers είναι φορητοί και εύκολο να μεταφερθούν από ένα περιβάλλον σε ένα άλλο, ανεξάρτητα από το λειτουργικό σύστημα ή την υποδομή.
• **Αυτάρκεια**: Οι containers μπορούν να δημιουργηθούν, να ενσωματωθούν και να καταστραφούν αυτόματα, διευκολύνοντας τη διαδικασία της ανάπτυξης και της παράδοσης του λογισμικού.

Παραδείγματα χρήσης

Για να κατανοήσουμε καλύτερα τη χρήση του Docker, ας εξετάσουμε μερικά παραδείγματα εφαρμογών:
• **Ανάπτυξη Εφαρμογών**: Ένας προγραμματιστής μπορεί να χρησιμοποιήσει το Docker για να δημιουργήσει ένα container που περιλαμβάνει όλα τα εργαλεία και τα dependencies που απαιτούνται για την ανάπτυξη της εφαρμογής του, εξαλείφοντας την ανάγκη για περιβάλλον ανάπτυξης σε κάθε μηχάνημα ξεχωριστά.
• **Δοκιμές Εφαρμογών**: Μια ομάδα δοκιμών μπορεί να χρησιμοποιήσει το Docker για να δημιουργήσει ένα περιβάλλον container που περιλαμβάνει όλα τα απαραίτητα στοιχεία για την εκτέλεση των δοκιμών, εξαλείφοντας τυχόν προβλήματα με τα dependencies του λογισμικού.
• **Παραγωγικό Περιβάλλον**: Ένας διαχειριστής συστήματος μπορεί να χρησιμοποιήσει το Docker για να δημιουργήσει containers που περιλαμβάνουν εφαρμογές ή υπηρεσίες και να τα εκτελέσει στο παραγωγικό περιβάλλον, εξασφαλίζοντας συνέπεια και αξιοπιστία.
• **Κατανεμημένα Συστήματα**: Μια εφαρμογή που απαιτεί κατανεμημένη αρχιτεκτονική μπορεί να εκμεταλλευτεί το Docker για τη δημιουργία διαφορετικών υπηρεσιών σε containers, τα οποία μπορούν να επικοινωνήσουν μεταξύ τους μέσω δικτύου.

ΕΙΚΌΝΑ 15:DOCKER

Συμπεράσματα

Συνολικά, το Docker προσφέρει ένα ισχυρό εργαλείο για τη δημιουργία, ανάπτυξη και εκτέλεση εφαρμογών. Με την απομόνωση, την ελαφρότητα και την ευελιξία που παρέχει, το Docker καθιστά την ανάπτυξη λογισμικού πιο αποτελεσματική και τη διαχείριση των εφαρμογών πιο αποτελεσματική. Με τη συνεχή ανάπτυξη και την υποστήριξη της κοινότητας, αναμένεται να δούμε ακόμα περισσότερες καινοτομίες και βελτιώσεις στο μέλλον, κάνοντας το Docker ακόμα πιο ανεκτίμητο για την ανάπτυξη λογισμικού και τη διαχείριση των υποδομών.

Αρχείο DOCKER-COMPOSE.YAML

Το αρχείο docker-compose.yaml αποτελεί ένα αρχείο ρύθμισης που χρησιμοποιείται για τον καθορισμό και τη διαχείριση πολλαπλών υπηρεσιών σε ένα περιβάλλον Docker. Αυτό περιλαμβάνει τις παραμέτρους και τις ρυθμίσεις που απαιτούνται για τη δημιουργία και την εκτέλεση αυτών των υπηρεσιών, όπως οι πόρτες που θα χρησιμοποιηθούν, τα volumes που θα χρησιμοποιηθούν και άλλες ρυθμίσεις περιβάλλοντος.

```yaml
version: '3'
services:
  backend:
    build: ./backend
    ports:
      - '5000:5000'
    volumes:
      - './backend:/app'

  frontend:
    image: node:21-alpine
    volumes:
      - './frontend:/app'
    ports:
      - "5173:5173"
    build: ./frontend
    environment:
      - CHOKIDAR_USEPOLLING=true
```

Στην περίπτωση μας ορίζουμε δύο υπηρεσίες:
backend: Ορίζεται με

- build: ./backend, προσδιορίζοντας ότι το image της υπηρεσίας θα δημιουργηθεί από το Dockerfile που βρίσκεται στον κατάλογο ./backend.
- Οι πόρτες ports: - '80:80' καθορίζουν ότι η υπηρεσία είναι προσβάσιμη από τη θύρα 80 του συστήματος και τα
- volumes: - './backend:/app' καθορίζουν ότι ο τρέχων φάκελος ./backend του συστήματος είναι διαθέσιμος ως /app μέσα στο container.

- frontend: Ορίζεται με το image: node:21-alpine, σημαίνοντας ότι χρησιμοποιείται το image node:21-alpine ως βάση για το container.
- Τα volumes: - './frontend:/app' δηλώνουν ότι ο τρέχων φάκελος ./frontend του συστήματος είναι διαθέσιμος ως /app μέσα στο container.
- Οι πόρτες ports: - "5173:5173" καθορίζουν ότι η υπηρεσία είναι προσβάσιμη από τη θύρα 5173 του συστήματος.
- Τέλος, το environment: - CHOKIDAR_USEPOLLING=true καθορίζει μια μεταβλητή περιβάλλοντος που θέτει την επιλογή CHOKIDAR_USEPOLLING σε true.

Dockerfile frontend

Το αρχείο **Dockerfile frontend** περιέχει οδηγίες για το πώς θα δημιουργηθεί το image ενός container στο Docker. Αυτό καθορίζει την εκκίνηση του image από μια συγκεκριμένη βάση, τον φάκελο εργασίας του container, τις εντολές αντιγραφής αρχείων στον container, τις εντολές για τη δημιουργία του image, και την εντολή εκκίνησης του container.

```
FROM node:21-alpine
WORKDIR /app
COPY package*.json .
RUN npm install
CMD ["npm","run","dev"]
```

Αναλυτικά:

- Από node:21-alpine: Αυτή η γραμμή ορίζει τη βάση για το image του container, χρησιμοποιώντας την επίσημη εικόνα του Node.js σε έκδοση 21, η οποία βασίζεται στην Alpine Linux για ελαφρύτερη εκδοχή.
- Ορισμός του WORKDIR /app: Αυτή η εντολή ορίζει τον φάκελο εργασίας του container στον οποίο θα εκτελείται η εφαρμογή.
- Αντιγραφή των αρχείων package.json και package-lock.json στον container.
- Εγκατάσταση των dependencies της εφαρμογής χρησιμοποιώντας το npm κατά τη δημιουργία του image.
- Ορισμός της προεπιλεγμένης εντολής που θα εκτελείται όταν ξεκινά ο container, πιθανότατα ξεκινώντας τον ανάπτυξης του server της εφαρμογής.

Dockerfile backend

Το αρχείο Dockerfile backend παρέχει οδηγίες για το πώς θα δημιουργηθεί το image ενός container στο Docker, καθορίζοντας τη βάση του image, τον φάκελο εργασίας του container, τις εντολές αντιγραφής αρχείων, την εγκατάσταση dependencies και την εντολή εκκίνησης του container.

```
FROM python:3.11-alpine

WORKDIR /app
COPY requirements.txt /app
RUN pip install -r requirements.txt
CMD ["flask","run","--port","5000","--host","0.0.0.0"]

EXPOSE 80
```

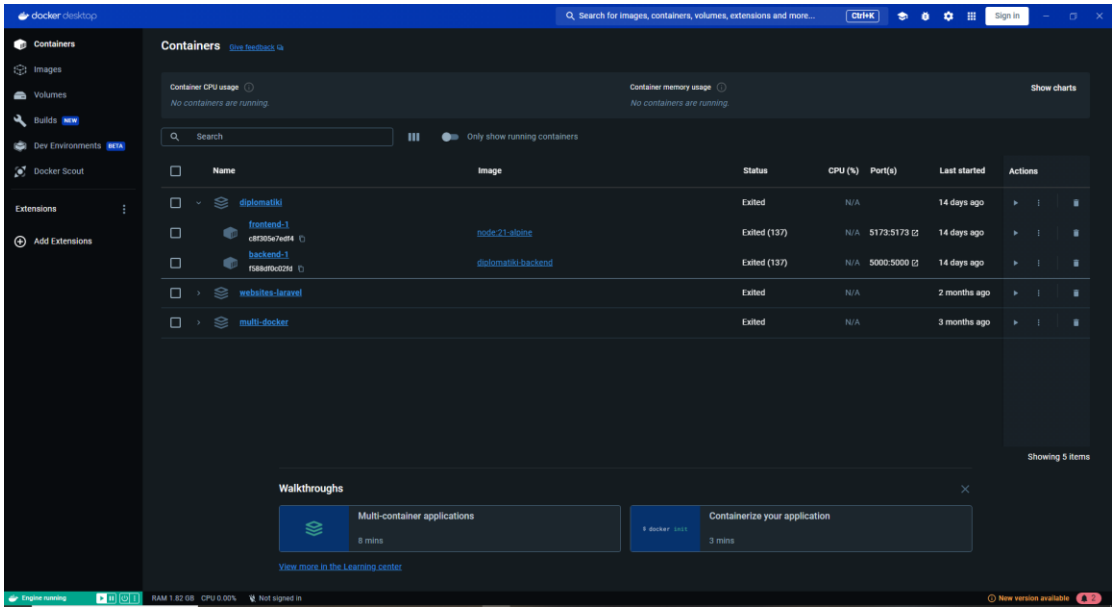
Αναλυτικά:

- Από python:3.11-alpine: Αυτή η γραμμή ορίζει το βασικό image για το container, χρησιμοποιώντας την εικόνα Python 3.11-alpine.

- Ορισμός του WORKDIR /app: Αυτή η εντολή ορίζει τον κατάλογο εργασίας μέσα στο container.
- Αντιγραφή του αρχείου requirements.txt στον directory /app μέσα στο container.
- Εγκατάσταση των απαιτούμενων dependencies της Python που αναφέρονται στο αρχείο requirements.txt χρησιμοποιώντας το pip.
- Ορισμός της προεπιλεγμένης εντολής που θα εκτελείται όταν ξεκινά ο container, εκκινώντας τον Flask development server.

Docker desktop

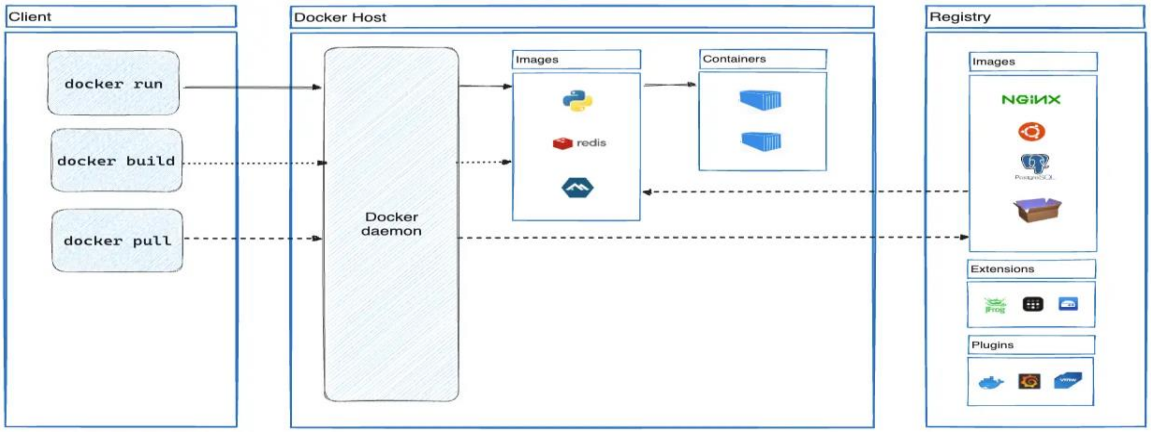
Για να γίνει το deploy εύκολα και με τα πλεονεκτήματα που αναφέρθηκαν παραπάνω, έχουμε χρησιμοποιήσει το εργαλείο docker desktop καθώς το λειτουργικό μας περιβάλλον είναι windows .
Για να «σηκωθεί» ένα container χρησιμοποιούμε την εντολή docker compose up στο root directory που βρίσκεται το έργο μας.
Προκαταβολικά έχουμε ανοίξει το docker desktop .
Η παρακάτω εικόνα δείχνει το πως εμφανίζεται το έργο μας.



Εικονα 16:DOCKER DESKTOP

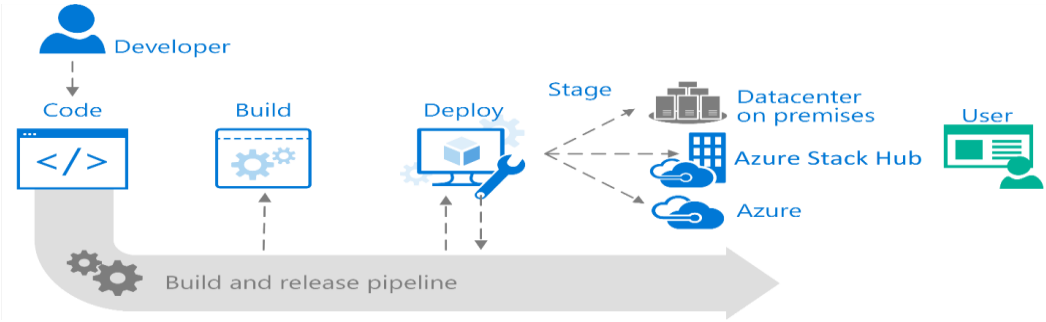
Έχει δημιουργηθεί ένα container που περιέχει ένα frontend και ένα backend image.
Έτσι πλέον ο server μας έχει ενεργοποιηθεί και μπορούμε πλέον να δούμε το UI μας στην πόρτα 5173.

Solution Docker Compose Schema



ΕΙΚΌΝΑ 17: DOCKER COMPOSE SCHEMA

Solution Deploy Pipeline



ΕΙΚΟΝΑ 18: DEPLOY PIPELINE

Frontend

Για την υλοποίηση του έργου μας, επιλέξαμε να χρησιμοποιήσουμε το framework Vue.js της JavaScript. Η απόφαση αυτή βασίστηκε σε προσεκτική ανάλυση και εκτίμηση των αναγκών του έργου, καθώς και στην επίγνωση των πλεονεκτημάτων που προσφέρει το Vue.js στην ανάπτυξη του frontend.

Το Vue.js επιλέχθηκε για αρκετούς λόγους, μεταξύ των οποίων συγκαταλέγονται:

1. **Ευκολία Χρήσης:** Το Vue.js προσφέρει μια εύκολη και ευέλικτη συντακτική δομή που επιτρέπει στους προγραμματιστές να αναπτύσσουν γρήγορα και αποτελεσματικά τις εφαρμογές τους. Η ομαλή ενσωμάτωση του Vue.js σε υπάρχοντα έργα είναι επίσης μια αξιοσημείωτη προσθήκη.
2. **Αποδοτικότητα:** Ο συνδυασμός της απλότητας του Vue.js με τη δυνατότητα δημιουργίας πολύπλοκων εφαρμογών καθιστά το framework ιδανικό για έργα με διαφορετικά επίπεδα πολυπλοκότητας.
3. **Απόδοση:** Το Vue.js διαθέτει αποδοτικούς μηχανισμούς ανανέωσης, οι οποίοι βελτιώνουν την απόδοση της εφαρμογής μας, επιτρέποντάς μας να παρέχουμε μια ομαλή εμπειρία χρήστη.

Σχεδιασμός

Για κάθε μία από τις σελίδες που θέλαμε να εμφανίζουμε στο frontend μας, δημιουργήσαμε το αντίστοιχο component. Κεντρική σελίδα είναι ή

- App.vue η οποία καθοδηγεί τον χρήστη σε ποια σελίδα βρίσκεται.

```
<template>
  <div v-if="user.isLoggedIn">
    <SideBar />
  </div>
  <div v-if="!user.isLoggedIn && !user.isRegistering">
    <LoginPage />
  </div>
  <div v-if="user.isRegistering">
    <RegisterForm />
  </div>
</template>
```

ΕΙΚΌΝΑ 19: APP. VUE

- Στην συνέχεια έχουμε τις σελίδες login.vue και register.vue.
- Για την αποφυγή πολλών redirect και την συντομότερη σε χρόνο ανανέωση της σελίδας δημιουργήσαμε το SideBar.vue που περιλαμβάνει όλες τις επιμέρους σελίδες μας.

- Clinics.vue
- Emergencies.vue
- Hospitals.vue
- InsertNewClinic.vue
- InsertNewEmail.vue
- InsertNewFile.vue
- InsertNewMedicine.vue
- InsertNewPhone.vue
- InsertNewRecipe.vue
- InsertNewVisitation.vue
- Medicine.vue
- MyBodyfat.vue
- MyDiet.vue
- MyFiles.vue
- MyProfile.vue
- PhoneEmail.vue
- Randevous.vue
- Recipes.vue
- UpdateRecipe.vue
- Visitations.vue

Side bar

Στο side bar χτίζονται δυναμικά όλες οι επιμέρους σελίδες, με αυτόν τον τρόπο αποφεύγουμε τα redirect σε σελίδες με αποτέλεσμα να φορτώνει η σελίδα σε λίγα δευτερόλεπτα.
Με την δημιουργία των components ανάλογα με το ποιο στοιχείο έχει επιλεχτεί εμφανίζεται η σελίδα όπως φαίνεται παρακάτω.

```
 <section class="home-section ">
    <i :class="currentActiveItem().icon"></i>
    <span class="text">{{ currentActiveItem().name }}</span>
    <div class="home-content">
      <component :is="selectedOption" v-if="selectedOption !== 'My
Profile'" />
      <Hospitals v-if="currentActiveItem().name === 'Hospitals'" />
      <Recipes v-else-if="currentActiveItem().name === 'Recipes'" />
      <PhoneEmail v-else-if="currentActiveItem().name === 'Associated
emails / phones'" />
      <Clinics v-else-if="currentActiveItem().name === 'Clinics'" />
      <Medicine v-else-if="currentActiveItem().name === 'Medicine'" />
      <Visitations v-else-if="currentActiveItem().name === 'Visitations'"
/>
      <MyProfile v-else-if="currentActiveItem().name === 'My Profile' "/>
      <Emergencies v-else-if="currentActiveItem().name === 'Emergencies'"
/>
      <MyFiles v-else-if="currentActiveItem().name === 'My Files'" />
      <MyDiet v-else-if="currentActiveItem().name === 'My Diet'"/>
      <MyBodyfat v-else-if="currentActiveItem().name === 'My Bodyfat'"/>
    </div>
  </section>
```

Η σελίδα MY PROFILE αποτελεί την πρώτη σελίδα που μεταφερόμαστε μετά την επιτυχή είσοδο μας. Όταν επιλέγεται άλλη σελίδα τότε η συγκεκριμένη παίρνει την κλάση currentActiveItem αι αλλάζει το χρώμα στα αριστερά.

My Profile

```
<div v-if="userInfo" class="border p-3 mb-2 bg-white text-dark">
  <h5 class="border-bottom text-center">Προσωπικές Πληροφορίες</h5>
  <div class="row mb-3">
    <div class="col-6">
      <label for="name" class="form-label">Όνομα:</label>
    </div>
    <div class="col-6">
      {{ userInfo.user_name }}
    </div>
  </div>
  <div class="row mb-3">
    <div class="col-6">
      <label for="surname" class="form-label">Επώνυμο:</label>
    </div>
    <div class="col-6">
      {{ userInfo.user_surname }}
    </div>
  </div>
  <div class="row mb-3">
    <div class="col-6">
      <label for="phone" class="form-label">Τηλέφωνο:</label>
    </div>
    <div class="col-6" style="font-size: 14px;">
      <ul>
        <li v-for="phone in userInfo.associated_phones" :key="phone">{{
phone }}</li>
      </ul>
    </div>
  </div>
  <div class="row mb-3">
    <div class="col-6">
      <label for="email" class="form-label">Email:</label>
    </div>
    <div class="col-6" style="font-size: 14px;">
      <ul>
        <li v-for="email in userInfo.associated_emails" :key="email">{{
email }}</li>
      </ul>
    </div>
   </div>
  </div>
</div>
```

Hospitals

Στην συνέχεια της περιήγησης επισκεπτόμαστε στη σελίδα Hospitals. Σε αυτή την επιλογή εμφανίζονται όλα τα διαθέσιμα νοσοκομεία.

```
<script>
  import { hospital } from "@/store/hospitals.js";
export default {
  name: "Hospital",
  async mounted() {
    this.hospitalInfos = await hospital.getHospitalInfo();
  },
  data() {return { hospitalInfos: [] }; },
  openNewWindow(url) {// Open the new window with the provided URL
      window.open(url, '_blank');} };
</script>
<template>
<div id="app" class="container mt-5">
    <div v-if="hospitalInfos">
      <div class="row">
        <div class="col-6" v-for="hospitalInfo in this.hospitalInfos" >
            <div class="card">
            <div class="row">
              <div class="col-4">
                Όνομα Νοσοκομείου:
              </div>
              {{ hospitalInfo.hospital_name }}
            </div>
            <div class="row">
              <div class="col-4"> Διεύθυνση: </div>
                  {{ hospitalInfo.hospital_address }}
            </div>
            <div class="row">
              <div class="col-4"> Χάρτες: </div> <div class="col-8">
<a :href="hospitalInfo.hospital_maps_link" target="_blank">
{{hospitalInfo.hospital_maps_link }}</a></div>
              </div>
            <div class="row">
              <div class="col-4"> Τηλέφωνο: </div> {{
hospitalInfo.hospital_phone }}
              </div>
             <div class="row">
              <div class="col-4"> Email: </div> {{
hospitalInfo.hospital_email }}
              </div>
            </div>
          </div>
        </div>
    </div></div>
</template>
```

MyFiles

Ενδεικτικά στην σελίδα my files εμφανίζεται ένας πίνακας με όλα τα αρχεία που έχουν ανέβει. Αυτά τα αρχεία μπορεί να είναι συνταγές γιατρών αποτελέσματα εξετάσεων, ακτινογραφίες ή οτιδήποτε θέλει ο χρήστης να ανεβάσει προκειμένου να έχει ένα αρχείο συγκεντρωτικά αποθηκευμένο σε ένα σημείο.

```
  <div id="app" class="container mt-5">
    <button type="button" class="btn btn-primary" data-bs-
toggle="modal" data-bs-target="#uploadFile">
      Προσθήκη νέου αρχείου
    </button>
  <div class="card">
    <div>
      <table class="table table-bordered">
        <thead>
        <tr>
          <th>Id</th>
          <th>Όνομα Αρχείου</th>
          <th>Ενέργεια</th>
        </tr>
        </thead>
        <tbody>
        <tr v-for="files in filesInfo">
          <td>
            {{ files.id }}
          </td>
          <td>
            {{ files.filename }}
          </td>
          <td>
        <a class="btn btn-primary"
:href="'http://localhost:5000/download_file/'+
files.id+'/'+user.userId">Αποθήκευση αρχείου</a>
          </td>
        </tr>
        </tbody>
      </table>
      <div id="uploadFile" aria-hidden="true" aria-
labelledby="exampleModalLabel" class="modal fade" tabindex="-1">
        <div class="modal-dialog">
          <div class="modal-content">
            <div class="modal-header">
              <h5 id="exampleModalLabel" class="modal-title">Προσθήκη
νέου αρχείου</h5>
              <button aria-label="Close" class="btn-close" data-bs-
dismiss="modal" type="button"></button>
            </div>
            <InsertNewFile/>
          </div></div></div></div></div></div>
```

Επικοινωνία με backend

Για την επίτευξη της επικοινωνίας μεταξύ του back-end και του front-end δημιουργούνται αρχεία με κατάληξη ".js". Σε αυτά τα αρχεία περιέχονται οι εντολές που καλούνται για να ληφθούν οι πληροφορίες που επιλέγονται κάθε φορά στο front-end. Επιπλέον, σε αυτά τα συγκεκριμένα αρχεία έχουμε τη δυνατότητα να ενσωματώσουμε τη δική μας λογική.

- clinics.js
- emergencies.js
- files.js
- hospitals.js
- medicine.js
- phone_email.js
- recipes.js
- requests.js
- user.js
- visitations.js

Όπως φαίνεται στην παρακάτω εικόνα.

```
import {reactive} from "vue";       You, 2 months ago • insert phone/emails/visitations/medicine/clinic…
import axios from "axios";

export const phone_email = reactive({
    phone_data:  [],
    email_data:[],
    phone_info: {},
    email_info:{},

    async getPhoneInfo(userId) {
        this.phone_info = (await axios.get(`http://localhost:5000/get_phones_by_user_id/${userId}`)).data.phone_data
        return this.phone_info
    },
    async getEmailInfo(userId) {
        this.email_info = (await axios.get(`http://localhost:5000/get_emails_by_user_id/${userId}`)).data.email_data
        return this.email_info
    },

})
```

ΕΙΚΟΝΑ 20:BACKEND CONNECTION PHONE-EMAIL

Για την ανάπτυξη χρησιμοποιήθηκε το αρχείο vite.config.js με χρήση ccs και της βιβλιοθήκης bootstrap.

Ποιο αναλυτικά

Δηλώσεις Εισαγωγής:

```
import {reactive} from "vue";
import axios from "axios";
```

Αυτός ο κώδικας εισάγει τη συνάρτηση **reactive** από τη βιβλιοθήκη Vue.js και τη βιβλιοθήκη **axios**, η οποία είναι μια δημοφιλής βιβλιοθήκη JavaScript για την κατασκευή αιτημάτων HTTP.

Clinics.js

Δήλωση Εξαγωγής:

```
export const clinic = reactive({
    clinic_data:[],
    clinic_info: {},
    async getClinicInfo(userId) {
        this.clinic_info = (await
axios.get(`http://localhost:5000/fetch_all_clinics_by_user_id?use
r_id=${userId}`)).data.clinics
        console.log(this.clinic_info)
        return this.clinic_info
    },
})
```

- Το **clinic** είναι ένα εξαγόμενο σταθερό αντικείμενο.
- Το αντικείμενο **clinic** γίνεται αντιδραστικό χρησιμοποιώντας τη συνάρτηση **reactive** από το Vue.js. Αυτό σημαίνει ότι οποιεσδήποτε αλλαγές στις ιδιότητές του θα προκαλέσουν ενημερώσεις αντίδρασης σε συστατικά Vue.js που τις αναφέρουν.
- Έχει δύο ιδιότητες:
    - **clinic_data**: Ένα κενό πίνακα.
    - **clinic_info**: Ένα κενό αντικείμενο.
- Περιλαμβάνει έναν μέθοδο **getClinicInfo(userId)**:
    - Αυτή η μέθοδος είναι ασύγχρονη (με τη λέξη-κλειδί **async**).
    - Παίρνει ένα παράμετρο **userId**.
    - Κάνει ένα αίτημα HTTP GET στο **http://localhost:5000/fetch_all_clinics_by_user_id** με το **userId** ως παράμετρο ερωτήματος.
    - Περιμένει την απάντηση από τον διακομιστή.
    - Αναθέτει την ιδιότητα **clinic_info** του αντικειμένου **clinic** στα δεδομένα **clinics** που λαμβάνονται από την απάντηση του διακομιστή.
    - Καταγράφει το **clinic_info** στην κονσόλα.
    - Τέλος, επιστρέφει το **clinic_info**.

Emergencies.js

Δήλωση Εξαγωγής:

```
export const emergency = reactive({
    emergency_info: {},
    async getEmergencyInfo() {
        if (this.emergency_info.length > 0) {
            return this.emergency_info;
        }
        this.emergency_info = (await
axios.get(`http://localhost:5000/fetch_all_emergencies`)).data.emergenc
ies
        console.log(this.emergency_info);
        return this.emergency_info
    },
})
```

- Το **emergency** είναι ένα εξαγόμενο σταθερό αντικείμενο.
- Το αντικείμενο **emergency** γίνεται αντιδραστικό χρησιμοποιώντας τη συνάρτηση **reactive** από το Vue.js. Αυτό σημαίνει ότι αλλαγές στις ιδιότητές του θα προκαλέσουν ενημερώσεις αντίδρασης σε συστατικά Vue.js που τις αναφέρουν.
- Έχει μια ιδιότητα **emergency_info**, η οποία αρχικά είναι ένα κενό αντικείμενο.
- Περιλαμβάνει μια μέθοδο **getEmergencyInfo()**:
    - Αυτή η μέθοδος είναι ασύγχρονη.
    - Ελέγχει αν η **emergency_info** έχει μήκος μεγαλύτερο από 0. Αν ναι, επιστρέφει αμέσως τα δεδομένα **emergency_info**.
    - Αν όχι, κάνει ένα αίτημα HTTP GET στη διεύθυνση **http://localhost:5000/fetch_all_emergencies**.
    - Περιμένει την απάντηση από τον διακομιστή.
    - Αναθέτει τα δεδομένα **emergencies** από την απάντηση στην **emergency_info**.
    - Καταγράφει τα **emergency_info** στην κονσόλα.
    - Τέλος, επιστρέφει τα δεδομένα **emergency_info**

Files.js

Δήλωση Εξαγωγής:

```
export const files = reactive({
    file: '',user_id : 0,file_id: 0,comments:'',files_info: {},
    async getUserFiles()
    {
        this.files_info = (await
axios.get(`http://localhost:5000/get_all_filenames/${this.user_id}`)).d
ata.filenames
        return this.files_info
    },

    async donwloadFile() {
        return await
axios.get(`http://localhost:5000/download_file/${this.file_id}/${this.u
ser_id}`);
    },
    async uploadFile() {
        return await
axios.post(`http://localhost:5000/upload_new_file`, {
            'user_id': this.user_id, 'comments': this.comments, 'file':
this.file
        });
    }
})
```

- Το **files** είναι ένα εξαγόμενο σταθερό αντικείμενο.
- Το αντικείμενο **files** γίνεται αντιδραστικό χρησιμοποιώντας τη συνάρτηση **reactive** από το Vue.js.
- Έχει πολλές ιδιότητες, συμπεριλαμβανομένων:
    - **file**: Το όνομα του αρχείου που θα ανέβει.
    - **user_id**: Το αναγνωριστικό του χρήστη.
    - **file_id**: Το αναγνωριστικό του αρχείου που θα ληφθεί.
    - **comments**: Σχόλια σχετικά με το αρχείο που θα ανέβει.
    - **files_info**: Πληροφορίες σχετικά με τα αρχεία του χρήστη.
- Περιλαμβάνει τρεις ασύγχρονες μεθόδους:
    - **getUserFiles()**: Ανακτά όλα τα ονόματα αρχείων για έναν συγκεκριμένο χρήστη από τον διακομιστή.
    - **donwloadFile()**: Κατεβάζει ένα αρχείο από τον διακομιστή χρησιμοποιώντας το αναγνωριστικό αρχείου και το αναγνωριστικό χρήστη.
    - **uploadFile()**: Ανεβάζει ένα νέο αρχείο στον διακομιστή, περνώντας το αναγνωριστικό χρήστη, τα σχόλια και το αρχείο ως δεδομένα προς τον διακομιστή.

Hospitals.js

Δήλωση Εξαγωγής:

```
export const hospital = reactive({
    hospital_info: {},
    async getHospitalInfo() {
        this.hospital_info = (await
axios.get(`http://localhost:5000/fetch_all_hospitals`)).data.hospitals
        return this.hospital_info
    },
})
```

- Το **hospital** είναι ένα εξαγόμενο σταθερό αντικείμενο.
- Το αντικείμενο **hospital** γίνεται αντιδραστικό χρησιμοποιώντας τη συνάρτηση **reactive** από το Vue.js.
- Έχει μια ιδιότητα **hospital_info**, η οποία αρχικά είναι ένα κενό αντικείμενο.
- Περιλαμβάνει τη μέθοδο **getHospitalInfo()**:
    - Αυτή η μέθοδος είναι ασύγχρονη.
    - Κάνει ένα αίτημα HTTP GET στη διεύθυνση **http://localhost:5000/fetch_all_hospitals**.
    - Περιμένει την απάντηση από τον διακομιστή.
    - Αναθέτει τα δεδομένα **hospitals** από την απάντηση στην **hospital_info**.
    - Τέλος, επιστρέφει τα δεδομένα **hospital_info**.

Medicine.js

Δήλωση Εξαγωγής:

```
export const medicine = reactive({
    medicine_data:[],
    medicine_info: {},
    async getMedicineInfo() {
        this.medicine_info = (await
axios.get(`http://localhost:5000/get_all_medicines`)).data.medicines
        return this.medicine_info
    },
})
```

- Το **medicine** είναι ένα εξαγόμενο σταθερό αντικείμενο.
- Το αντικείμενο **medicine** γίνεται αντιδραστικό χρησιμοποιώντας τη συνάρτηση **reactive** από το Vue.js.
- Έχει δύο ιδιότητες:
    - **medicine_data**: Ένας κενός πίνακας.
    - **medicine_info**: Ένα κενό αντικείμενο.
- Περιλαμβάνει τη μέθοδο **getMedicineInfo()**:
    - Αυτή η μέθοδος είναι ασύγχρονη.
    - Κάνει ένα αίτημα HTTP GET στη διεύθυνση **http://localhost:5000/get_all_medicines**.
    - Περιμένει την απάντηση από τον διακομιστή.
    - Αναθέτει τα δεδομένα **medicines** από την απάντηση στην **medicine_info**.
    - Τέλος, επιστρέφει τα δεδομένα **medicine_info**.

Phone_email.js

Δήλωση Εξαγωγής:

```
export const phone_email = reactive({
    phone_data:  [],
    email_data:[],
    phone_info: {},
    email_info:{},
    async getPhoneInfo(userId) {
        this.phone_info = (await
axios.get(`http://localhost:5000/get_phones_by_user_id/${userId}`)).dat
a.phone_data
        return this.phone_info
    },
    async getEmailInfo(userId) {
        this.email_info = (await
axios.get(`http://localhost:5000/get_emails_by_user_id/${userId}`)).dat
a.email_data
        return this.email_info
    },

})
```

- Το **phone_email** είναι ένα εξαγόμενο σταθερό αντικείμενο.
- Το αντικείμενο **phone_email** γίνεται αντιδραστικό χρησιμοποιώντας τη συνάρτηση **reactive** από το Vue.js.
- Έχει τέσσερις ιδιότητες:
    - **phone_data**: Ένας κενός πίνακας για τα δεδομένα τηλεφώνου.
    - **email_data**: Ένας κενός πίνακας για τα δεδομένα email.
    - **phone_info**: Ένα κενό αντικείμενο για τις πληροφορίες τηλεφώνου.
    - **email_info**: Ένα κενό αντικείμενο για τις πληροφορίες email.
- Περιλαμβάνει δύο μεθόδους:
    - **getPhoneInfo(userId)**: Ανακτά τις πληροφορίες τηλεφώνου για έναν συγκεκριμένο χρήστη από τον διακομιστή.
    - **getEmailInfo(userId)**: Ανακτά τις πληροφορίες email για έναν συγκεκριμένο χρήστη από τον διακομιστή.

Recipes.js

Δήλωση Εξαγωγής:

```
export const recipe = reactive({
    recipe_data:[],
    recipe_info: {},
    async getRecipeInfo(userId) {
        this.recipe_info = (await
axios.get(`http://localhost:5000/fetch_recipes_by_user_id/${userId}`)).
data.recipes
        return this.recipe_info
    },
    async deleteRecipe(recipeId) {
        await axios.post('http://localhost:5000/delete_recipe',{
            'recipe_id': recipeId
        })
    }
})
```

- Το **recipe** είναι ένα εξαγόμενο σταθερό αντικείμενο.
- Το αντικείμενο **recipe** γίνεται αντιδραστικό χρησιμοποιώντας τη συνάρτηση **reactive** από το Vue.js.
- Έχει δύο ιδιότητες:
    - **recipe_data**: Ένας κενός πίνακας για τα δεδομένα συνταγών.
    - **recipe_info**: Ένα κενό αντικείμενο για τις πληροφορίες συνταγών.
- Περιλαμβάνει δύο μεθόδους:
    - **getRecipeInfo(userId)**: Ανακτά τις πληροφορίες συνταγών για έναν συγκεκριμένο χρήστη από τον διακομιστή.
    - **deleteRecipe(recipeId)**: Διαγράφει μια συνταγή με βάση το ID της συνταγής μέσω ενός αιτήματος POST προς τον διακομιστή.

User.js

Δήλωση Εξαγωγής:

```
export const user = reactive({
    userId: 0,
    username: '',
    isLoggedIn: false,
    isRegistering: false,
    email: '',
    password: '',
    userRegister: {
        name: '',
        surname: '',
        phone: '',
        address: '',
        email: '',
        password: '',
        verify_password: ''
      },
    user_info: {},
    userRegistering() {
        this.isRegistering = true
    },
    async getUserInfo() {
        if (this.user_info.length > 0) {
            return this.user_info;
        }
        this.user_info = (await
axios.get(`http://localhost:5000/get_user_info_by_id/${this.userId}`)).
data
        console.log(this.user_info);
        return this.user_info
    },
    async userLoggedIn() {
        const result = await axios.post("http://localhost:5000/login",
{
            email: this.email, password: this.password
        });
        if(result.data.status === "1") {
            //login successfully
            this.userId = result.data.user_id
            this.isLoggedIn = true;
            alert(result.data.message)
        } else {
            alert(result.data.message)
        }
    },
    async registerUser() {
```

```
        try {
          const response = await
axios.post('http://localhost:5000/register', {
            name: this.userRegister.name,
            surname: this.userRegister.surname,
            phone: this.userRegister.phone,
            address: this.userRegister.address,
            email: this.userRegister.email,
            password: this.userRegister.password
          });

          alert('Success');
          this.isRegistering = false;
          // You can redirect or show a success message here
        } catch (error) {
          console.error('Registration failed:', error);
          // Handle error response, show an error message to the user
        }
      }
})
```

- Το **user** είναι ένα εξαγόμενο σταθερό αντικείμενο.
- Το αντικείμενο **user** γίνεται αντιδραστικό χρησιμοποιώντας τη συνάρτηση **reactive** από το Vue.js.
- Έχει πολλές μεταβλητές και μεθόδους για τη διαχείριση του χρήστη και την αυθεντικοποίηση.
- Περιλαμβάνει μια μεθόδο **userRegistering()** για την ενεργοποίηση της διαδικασίας εγγραφής χρήστη.
- Περιλαμβάνει μια μέθοδο **getUserInfo()** για τη λήψη πληροφοριών του χρήστη από τον διακομιστή.
- Περιλαμβάνει μια μέθοδο **userLoggedIn()** για την εκτέλεση της διαδικασίας σύνδεσης του χρήστη.
- Περιλαμβάνει μια μέθοδο **registerUser()** για την εκτέλεση της διαδικασίας εγγραφής του χρήστη.

Visitations.js

Δήλωση Εξαγωγής:

```
export const visitation = reactive({
    visitation_data:[],
    visitation_info: {},
    async getVisitationInfo(userId) {
        this.visitation_info = (await
axios.get(`http://localhost:5000/get_visitations_by_user/${userId}`)).d
ata.visitations
        return this.visitation_info
    },
})
```

- Το **visitation** είναι ένα εξαγόμενο σταθερό αντικείμενο.
- Το αντικείμενο **visitation** γίνεται αντιδραστικό χρησιμοποιώντας τη συνάρτηση **reactive** από το Vue.js.
- Έχει δύο ιδιότητες:
    - **visitation_data**: Ένας κενός πίνακας για τα δεδομένα επισκέψεων.
    - **visitation_info**: Ένα κενό αντικείμενο για τις πληροφορίες επισκέψεων.
- Περιλαμβάνει τη μέθοδο **getVisitationInfo(userId)**:
    - Αυτή η μέθοδος είναι ασύγχρονη.
    - Κάνει ένα αίτημα HTTP GET στη διεύθυνση **http://localhost:5000/get_visitations_by_user/${userId}** για τις επισκέψεις του συγκεκριμένου χρήστη.
    - Περιμένει την απάντηση από τον διακομιστή.
    - Αναθέτει τα δεδομένα **visitations** από την απάντηση στην **visitation_info**.
    - Τέλος, επιστρέφει τα δεδομένα **visitation_info**.

My profile

Για την επικοινωνία του backend με το front στην σελίδα myProfile για παράδειγμα έχουμε το αρχείο που γίνεται import στην σελίδα του vue.

```
export const user = reactive({
    userId: 0,
    username: '',
    isLoggedIn: false,
    isRegistering: false,
    email: '',
    password: '',
    userRegister: {
        name: '',
        surname: '',
        phone: '',
        address: '',
        email: '',
        password: '',
        verify_password: ''
      },
    user_info: {},
    userRegistering() {
        this.isRegistering = true
    },
    async getUserInfo() {
        if (this.user_info.length > 0) {
            return this.user_info;
        }
        this.user_info = (await axios.get(`http://localhost:5000/get_user_info_by_id/${this.userId}`)).data
        console.log(this.user_info);
        return this.user_info
    },

    async userLoggedIn() {

        const result = await axios.post("http://localhost:5000/login", {
            email: this.email, password: this.password
        });
        if(result.data.status === "1") {
            //login successfully
            this.userId = result.data.user_id
            this.isLoggedIn = true;
            alert(result.data.message)
        } else {
            alert(result.data.message)
        }
    },
    async registerUser() {
        try {
          const response = await axios.post('http://localhost:5000/register', {
            name: this.userRegister.name,
            surname: this.userRegister.surname,
            phone: this.userRegister.phone,
            address: this.userRegister.address,
            email: this.userRegister.email,
            password: this.userRegister.password
          });

          alert('Success');
          this.isRegistering = false;
          // You can redirect or show a success message here
        } catch (error) {
          console.error('Registration failed:', error);
          // Handle error response, show an error message to the user
        }
      }
})
```

ΕΙΚΟΝΑ 21: BACKEND CONNECTION USER.JS

Σε αυτό το σημείο γίνεται η κλήση στο backend μέσω του localhost . Χρησιμοποιούμε το localhost επειδή το backend είναι ανεβασμένο σε τοπικό περιβάλλον docker. Για παράδειγμα αν το backend μας ήταν ανεβασμένο σε έναν server με ένα διαφορετικό όνομα τοτε αντι για localhost θα χρησιμοποιούσαμε το url του server.
Αντίστοιχα με το localhost, αν κάναμε μια κλήση σε εξωτερικό api τότε θα χρησιμοποιούσαμε το url και την πόρτα που μας επιστέφει το αποτέλεσμα που θέλουμε.

MyFiles

Άλλο ένα παράδειγμα επικοινωνίας είναι το backend είναι το My files όπως φαίνεται παρακάτω.

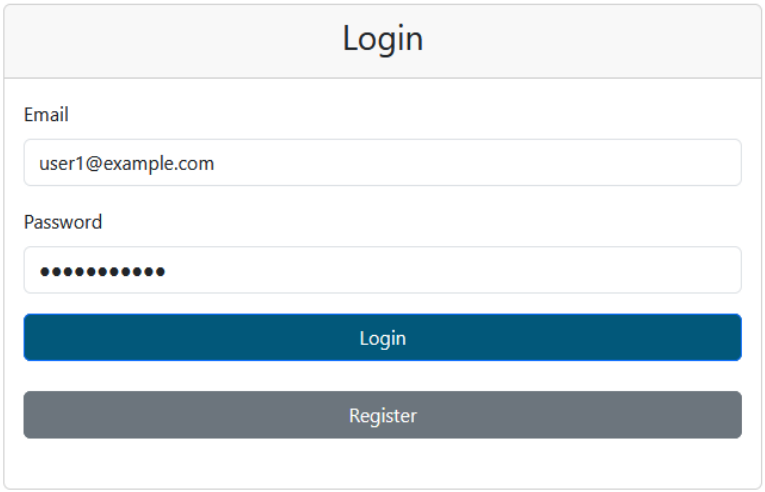
```
import {reactive} from "vue";
import axios from "axios";

export const files = reactive({

    file: '',user_id : 0,file_id: 0,comments:'',files_info: {},
    async getUserFiles()
    {
        this.files_info = (await
axios.get(`http://localhost:5000/get_all_filenames/${this.user_id}`)).data.filena
mes
        return this.files_info
    },

    async donwloadFile() {
        return await
axios.get(`http://localhost:5000/download_file/${this.file_id}/${this.user_id}`);
    },

    async uploadFile() {
        return await axios.post(`http://localhost:5000/upload_new_file`, {
            'user_id': this.user_id, 'comments': this.comments, 'file': this.file
        });
    }
})
```

Στο συγκεκριμένο παράδειγμα έχουμε wild cards όπως το user id και το file id.

Το τελικό αποτέλεσμα

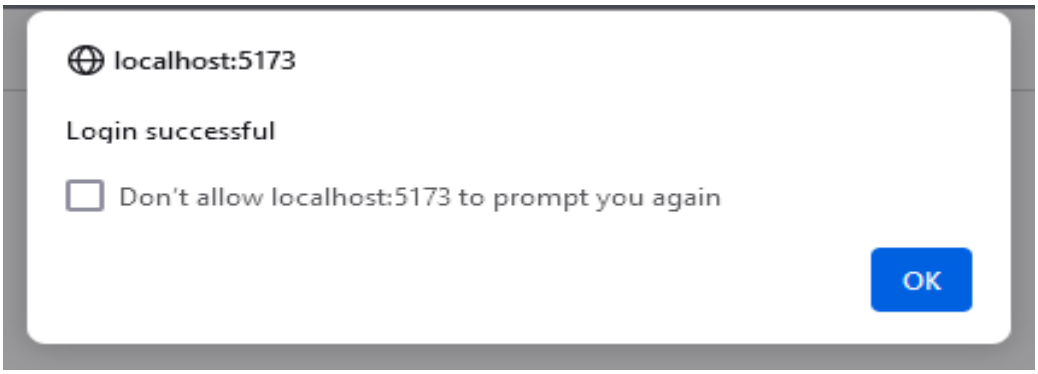
Login

Στην σελίδα αυτή ο χρήστης καλείται να εισάγει το email και τον κωδικό του. Σε περίπτωση που κάποιο από τα παραπάνω είναι λανθασμένο τότε επιστρέφεται σε αυτόν μήνυμα λάθους και καλείται να δοκιμάσει εκ νέου. Σε περίπτωση που ο χρήστης καταχωρήσει σωστά τα στοιχεία εισαγωγής τότε εμφανίζεται ένα μήνυμα επιτυχούς σύνδεσης. Πατώντας το οκ ο χρήστης ανακατευθύνεται στην αρχική σελίδα του έργου.



ΕΙΚΌΝΑ 22:LOGIN PAGE

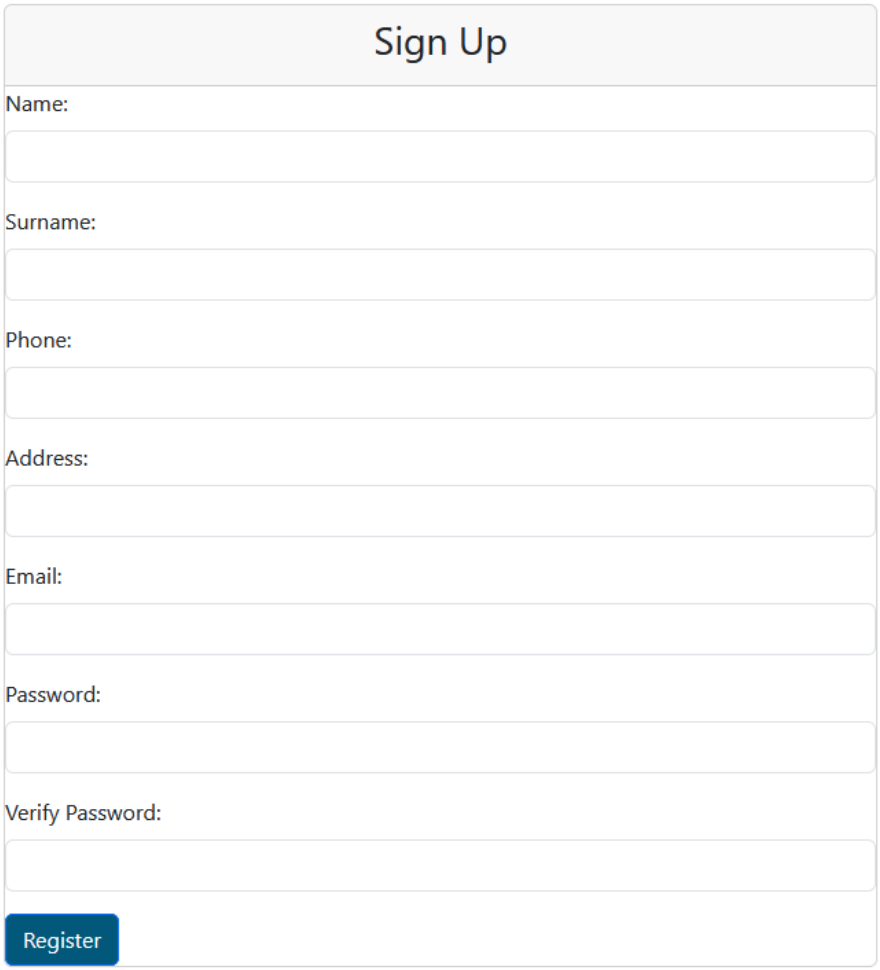
ΕΙΚΌΝΑ 23:UNSUCCESSFUL LOGIN



ΕΙΚΟΝΑ 24:SUCCESSFUL LOGIN

Signup

Αν κάποιος χρήστης επιθυμεί να κάνει εγγραφή στην συγκεκριμένη εφαρμογή τότε πρέπει να συμπληρώσει τα πεδία που φαίνονται στην παρακάτω εικόνα. Όλα τα πεδία είναι υποχρεωτικά. Το τηλέφωνο πρέπει να αποτελείται από δέκα ψηφία και το email να είναι της μορφής something@something_else.gr/com etc.

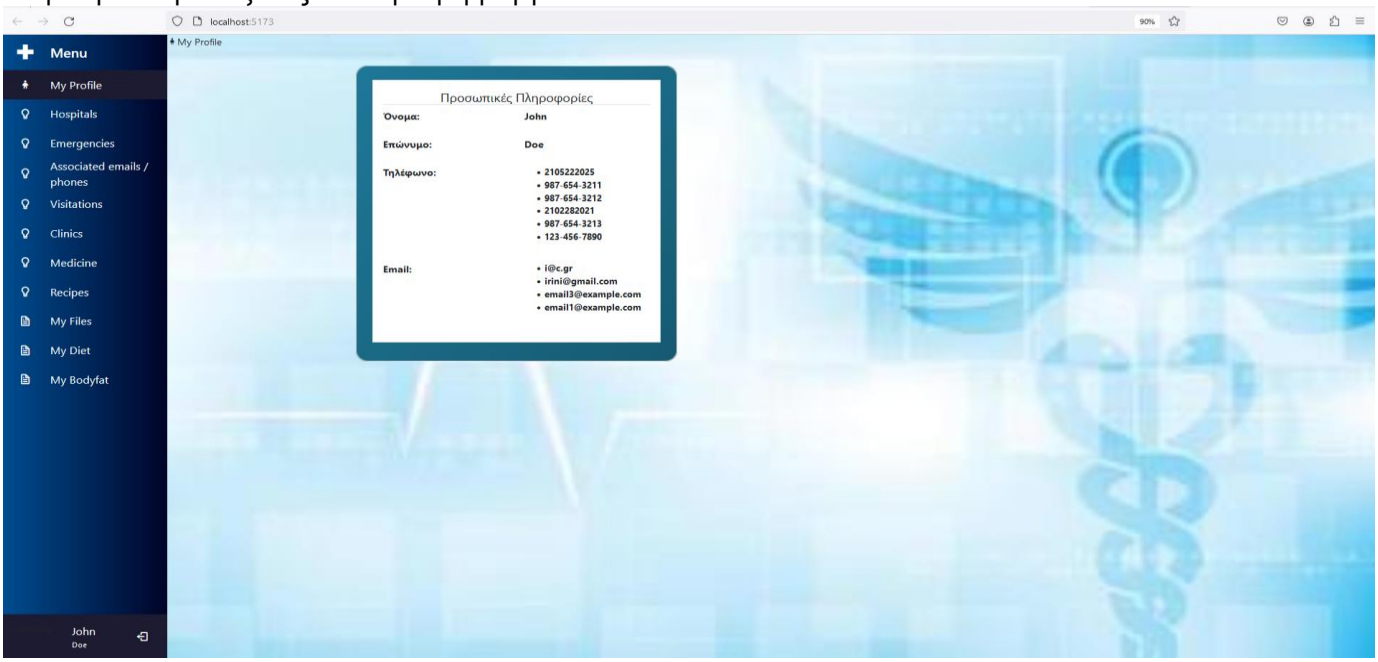


ΕΙΚΟΝΑ 25: SIGN UP

My profile

Μετά την επιτυχή είσοδο του χρήστη στην εφαρμογή η πρώτη σελίδα που εμφανίζεται είναι η σελίδα my profile. Σε αυτή θα βρει τα προσωπικά του στοιχεία, όπως το όνομα, το επώνυμο του καθώς και τα τηλέφωνα επικοινωνίας και τα email που αφορούν τις επαφές έκτακτης ανάγκης.

Επιπλέον στο κάτω αριστερό μέρος της σελίδας εμφανίζονται πάλι τα στοιχεία σύνδεσης και το κουμπί για να γίνει έξοδος από την εφαρμογή.



ΕΙΚΟΝΑ 26:MY PROFILE

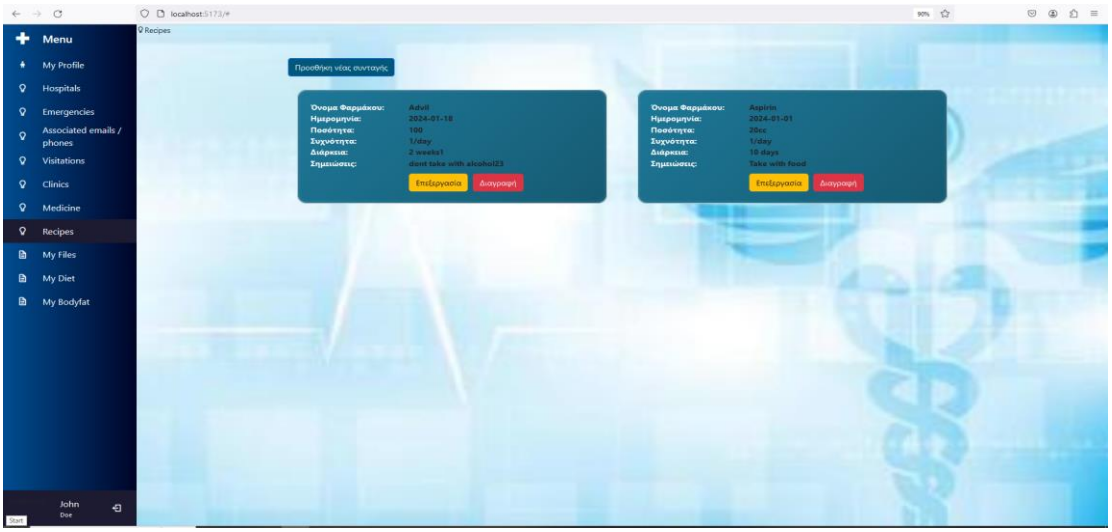
Recipes new-recipe edit-recipe

Σε αυτή την σελίδα ο χρηστης βλεπει σε κάρτες όλες τις συναγέςπου έχει λάβει. Οι πληροφορίες που γνωρίζει για αυτές είναι:

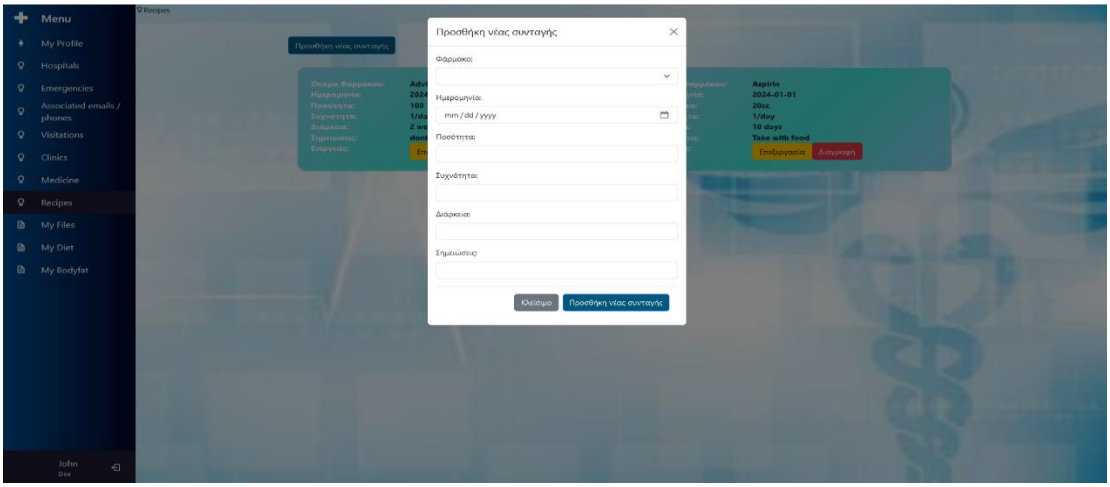
- Το ονομα του φαρμάκου
- Ημερομηνια ληψης
- Την ποσότητα πρόσληψης
- Την συχνοτητα προσληψης
- Την διάρκεια λήψης
- Και καποια σχόλια σχετικα με την λήψη
- Επιπλέον, εμφανίζεται το κουμπί «ΕΠΕΞΕΡΓΑΣΙΑ»
- Και το κουμπί «ΔΙΑΓΡΑΦΗ»

Στην σελίδα αυτή υπάρχει επίσης το κουμπί «ΠΡΟΣΘΗΚΗ ΝΕΑΣ ΣΥΝΤΑΓΗΣ». Στην Εικόνα 25 υπάρχει η καταχώρηση νέας συναγής με τα πεδία που ο χρήστης πρέπει να συμπληρώσει. Στο τέλος επιλέγει το κουμπί «Καταχώρηση νέας συναγής» και επιστρέφεται στην σελίδα με τις συναγές.

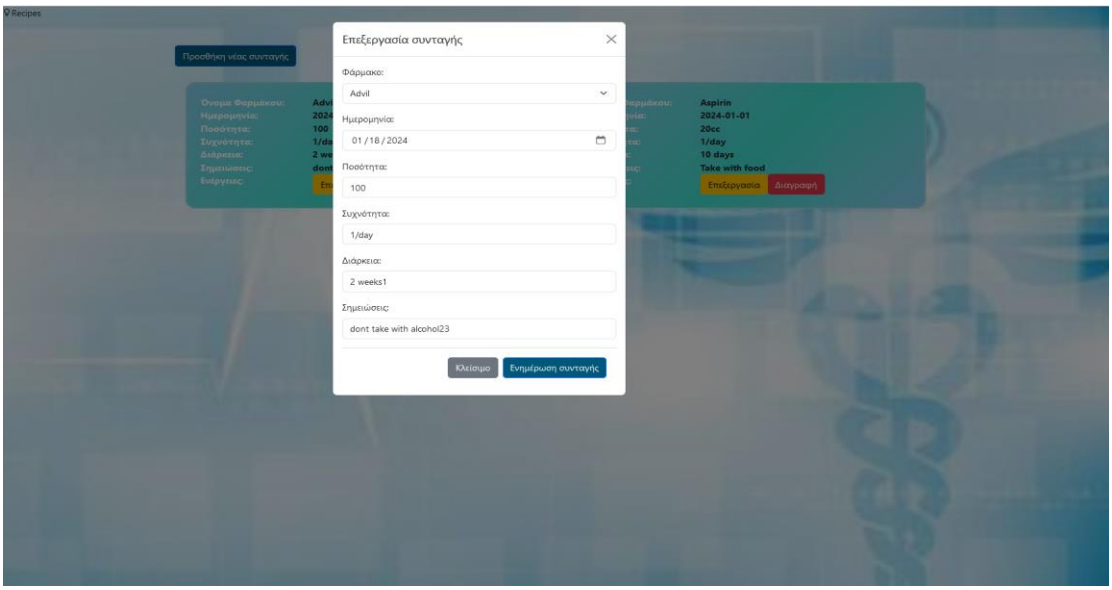
Αντιστοιχα την Εικόνα 27, έχει επιλεγεί μια συνταγη για επεξεργασία. Στην περίπτωση επεξεργασίας τα πεδία είναι συμπληρωμένα με βάση την προηγούμενη καταχώρηση. Ο χρήστης κάνει οποιαδήποτε αλλαγή επιθυμει και πατάει «Ενημέρωση συνταγής».



ΕΙΚΌΝΑ 27:RECIPES



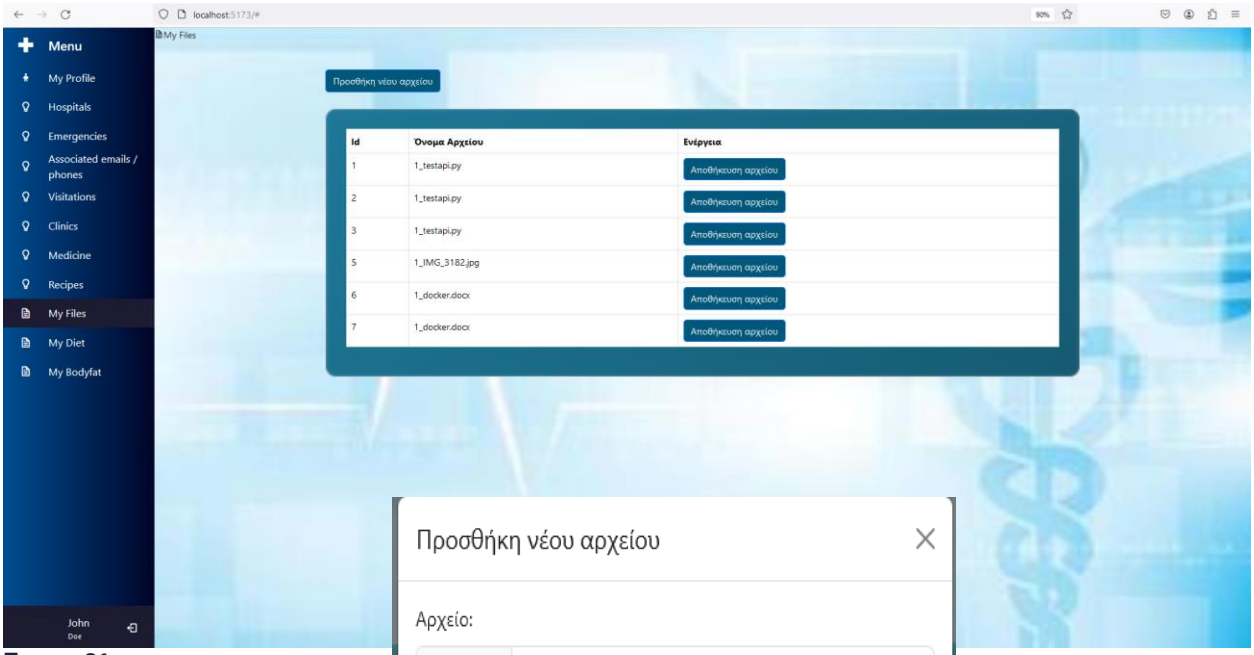
ΕΙΚΌΝΑ 28:ADD NEW RECIPE



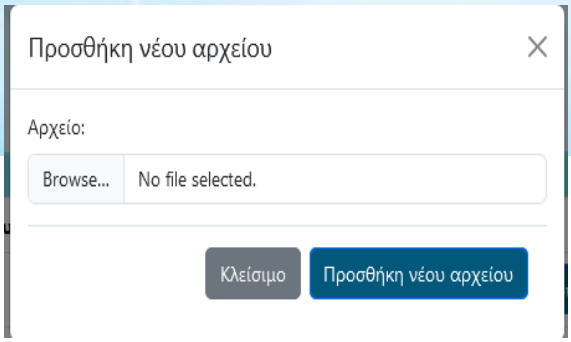
ΕΙΚΟΝΑ 29:EDIT RECIPE

Files

Στην επιλογή Files ο χρήστης έχει την επιλογή να κάνει upload ένα αρχείο παντωντας το κουμπί «προσθηκη νέου αρχειου». Αυτά τα άρχεια μπορει να είναι συνταγες γιατρου, πληροφορίες για ειδική διαιτα, αποτελέσματα εξετάσεων, ακτινογραφίες ή οτιδήποτε άλλο αρχειο που χρειάζεται να αρχειοθετησει για μελλοντική χρηση.



ΕΙΚΟΝΑ 31:FILES



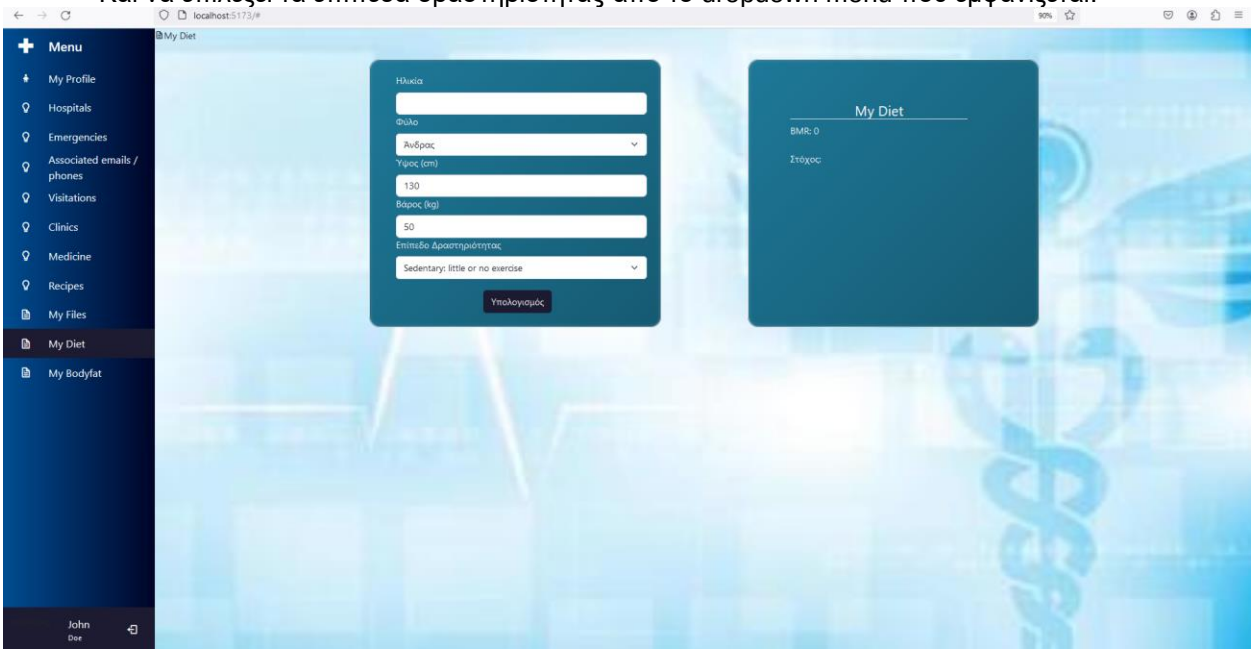
ΕΙΚΌΝΑ 30:ADD FILES

Επιπλέον ο χρηστης μπορεί ανα πασα στιγμη να κάνει download τα αρχεία που επιθυμει για να δει πληροφορίες σχετικές με την υγεία του.
Για την προσθηκη νέου αρχείου χρειάζεται να περιηγηθει στα τοπίκα του αρχεια. Τέλος, πατώντας το κουμπί «προσθηκη νέου αρχειου». Αν άλλαξε γνώμη μπορεί να επιλέξει το κουμπί «Κλείσιμο» ή το Χ που βρίσκεται στο πάνω δεξία μέρος του αναδυόμενου παραθύρου.

My diet

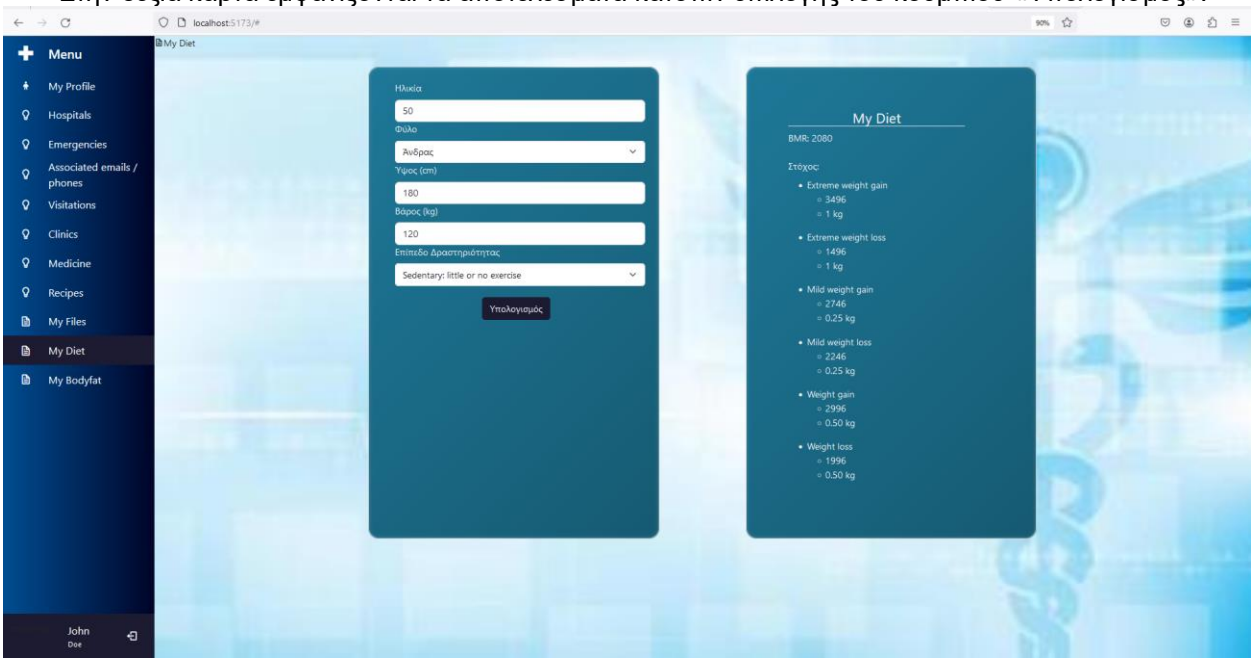
Στην επιλογή My diet εμφανίζονται δυο κάρτες. Η αριστερή κάρτα ζητά από τον χρήστη καλείται να συμπληρώσει τα πεδία

- Ηλικία
- Φύλο
- Ύψος
- Βάρος
- Και να επιλέξει τα επίπεδα δραστηριότητας από το dropdown menu που εμφανίζεται.



ΕΙΚΟΝΑ 32:MY DIET

Στην δεξιά κάρτα εμφανίζονται τα αποτελέσματα κατόπιν επιλογής του κουμπιού «Υπολογισμός».



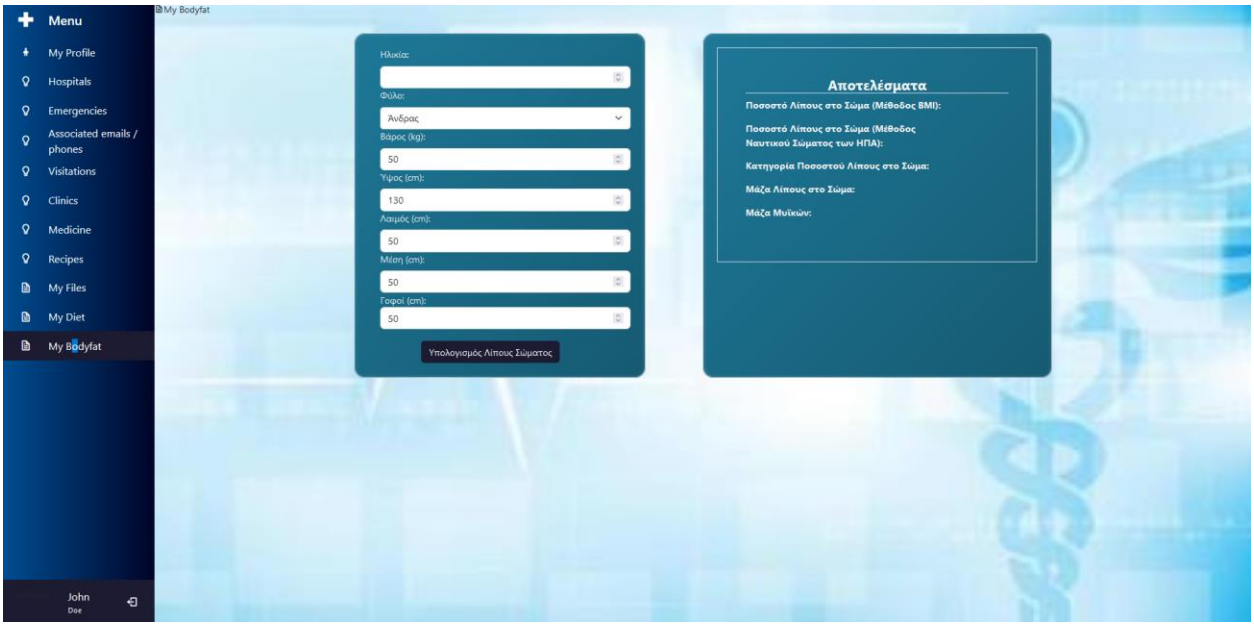
ΕΙΚΟΝΑ 33:MY DIET EXAMLE

Στην Εικόνα 30 εμφανίζονται τα αποτελέσματα για τις συγκεκριμένες μετρήσεις. Πιο αναλυτικά με βάση τον δείκτη μάζας σώματος αν ο χρήστης χρήζει ανάγκης έκτακτης μείωσης βάρους θα πρέπει να λαμβάνει 1496 θερμίδες ημερησίως. Για απλή απώλεια τότε η λήψη θερμίδων ανεβαίνει στις 1996 θερμίδες ημερησίως.

My Body Fat

Στην επιλογή My body fat εμφανίζονται δυο κάρτες.
Στην αριστερή κάρτα ο χρήστης καλείται να συμπληρώσει τα πεδία

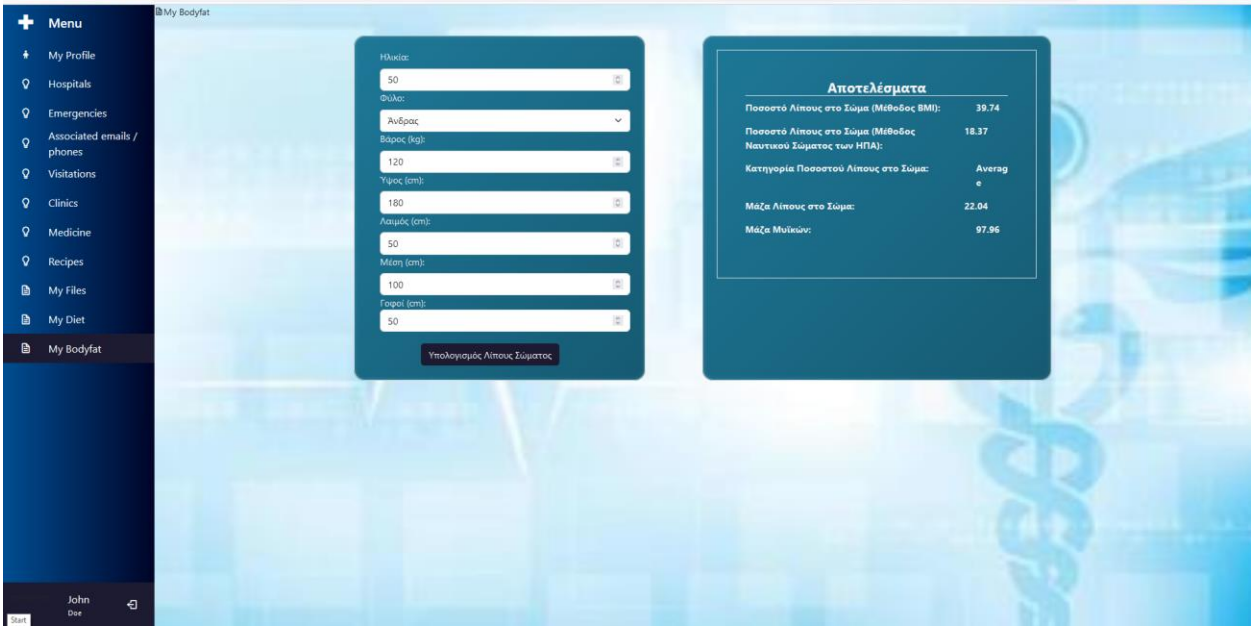
- Ηλικία
- Φύλο
- Ύψος
- Βάρος
- Λαιμός (cm)
- Μέση
- Γοφοί



ΕΙΚΟΝΑ 34:MY BODY FAT

Πατώντας το κουμπί «Υπολογισμός Λίπους Σώματος» εμφανίζονται τα αποτελέσματα στην αριστερή κάρτα όπως φαίνεται στην Εικόνα 32.
Πιο συγκριμένα ένας άνδρας, ηλικίας 50ετών, βάρους 120 κιλών, ύψους 180 εκατοστών, με λαιμό 50εκατοστων, μέση 100 εκατοστών και γοφούς 50 εκατοστών, λαμβάνει τις παρακάτω πληροφορίες.

- Το ποσοστό λίπους στο σώμα του (μέθοδος BMI) είναι 39,74%.
- Το ποσοστό λίπους στο σώμα του (Μέθοδος ναυτικού σώματος των ΗΠΑ) είναι 18,37%.
- Η κατηγορία ποσοστού λίπους στο σώμα είναι στον μέσο όρο.
- Η μάζα του λίπους στο σώμα του είναι 22,04 κιλά.
- Η μυϊκή μάζα του χρήστη είναι 97,96 κιλά.

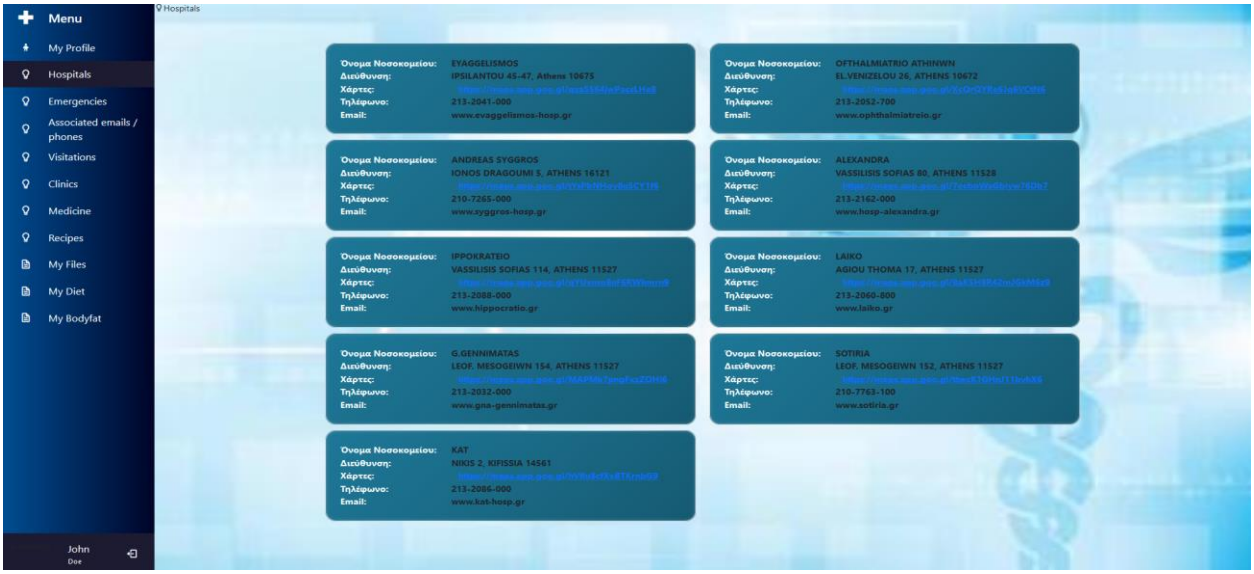


ΕΙΚΌΝΑ 35: MY BODY FAT EXAMPLE

Hospitals

Προκειμένου ο χρήστης να έχει άμεση πρόσβαση στα διαθέσιμα νοσοκομεία της Αττικής. Ο χρήστης βλέπει

- το όνομα του νοσοκομείου,
- την διεύθυνση αυτού,
- τον χάρτη, ο οποίος συνδέεται με το google maps για την εύκολη μετάβαση σε αυτό. Πατώντας το link του χάρτη ανοίγει μία νέα σελίδα με πινέζα το επιλεγμένο νοσοκομείο.
- Το τηλέφωνο του νοσοκομείου
- Και την ηλεκτρονική διεύθυνση του νοσοκομείου στην περίπτωση που ο χρήστης επιθυμεί την επικοινωνία.



ΕΙΚΟΝΑ 36:HOSPITAL

Emergencies

Για την διευκόλυνση του χρήστη επίσης, δημιουργήθηκε η σελίδα με τα επείγοντα.
Στην Εικόνα 35 υπάρχει

- το όνομα κάθε επείγουσας υπηρεσίας καθώς και
- το τηλέφωνο έκτακτης αναγκης.
-

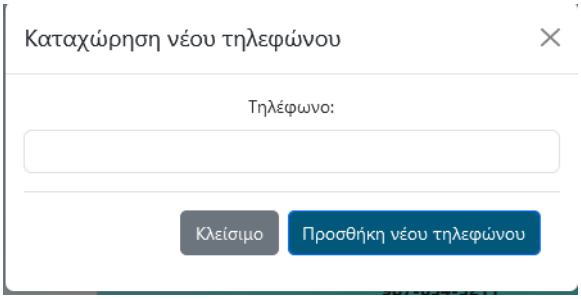
ΕΙΚΌΝΑ 37: EMERGENCIES

Phone-emails / new- phone new-email

Σε αυτή την επιλογή εμφανίζονται τα συσχετισμένα τηλέφωνα και έκτακτης ανάγκης.
Στο πάνω μέρος της οθόνης υπάρχουν τα κουμπιά «Καταχώρηση νέου email» και «Καταχώρηση νέου τηλεφώνου».

ΕΙΚΌΝΑ 38:EMAIL-PHONES

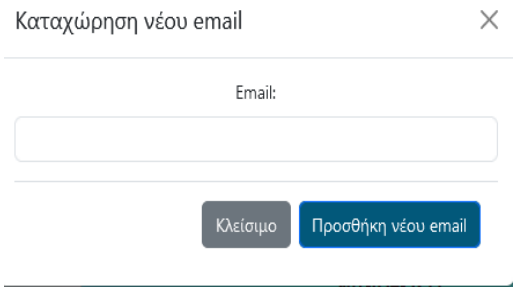
Στην καταχώρηση νέου τηλεφώνου έχουμε την Εικόνα 36. Πατώντας το κουμπί «Προσθήκη νέου



ΕΙΚΌΝΑ 39: ADD NEW PHONE

τηλεφώνου» καταχωρείται στην βάση δεδομένων το τηλέφωνο έκτακτης ανάγκης.

Στην καταχώρηση νέου email έχουμε την Εικόνα 37. Πατώντας το κουμπί «Προσθήκη νέου email»



ΕΙΚΌΝΑ 40: ADD NEW EMAIL

καταχωρείται στην βάση δεδομένων το email έκτακτης ανάγκης.

Visitations new-visitation

Στην επιλογή Visitations ο χρήστης βλέπει τις καταχωρημένες επισκέψεις που έχει πραγματοποιήσει η προκειται να πραγματοποιήσει. Εμφανίζονται οι πληροφορίες

- Νοσοκομειο
- Ημερομηνια
- Σημειώσεις/ σχόλια
- Κοστος

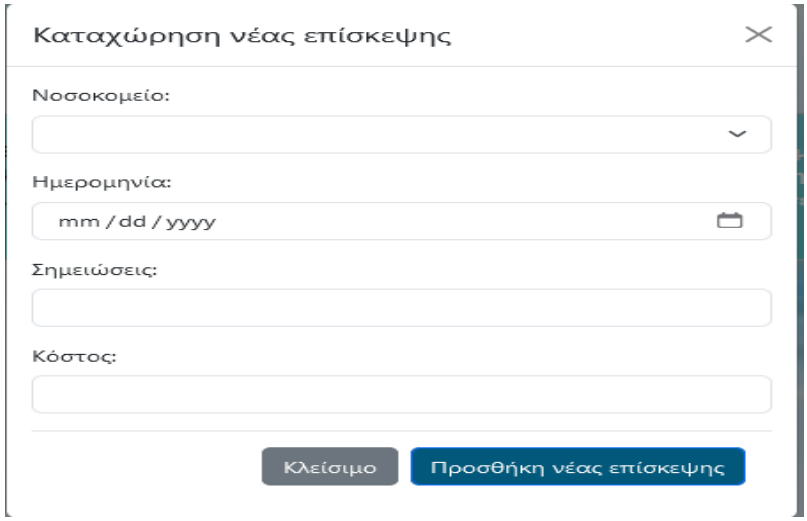
Στο πάνω μέρος της οθόνης υπάρχει το κουμπί «Καταχώρηση νέας επίσκεψης».

Στην Εικόνα 40 που αφορά την καταχώρηση νέας επίσκεψης εμφανίζονται τα πεδία

- Νοσοκομειο dropdown menu με τα ήδη καταχωρημένα νοσοκομεία που είδαμε στην επιλογή «Hospitals».

- Ημερομηνια υπάρχει ένα calendar ώστε να καταχωρηθεί σωστός τύπος ημερομηνίας.
- Σημειώσεις/ σχόλια
- Κοστος είναι αλφαριθμιτικό.

ΕΙΚΟΝΑ 41:VISITATIONS



ΕΙΚΌΝΑ 42: ADD VISITATION

Clinics new-clinic

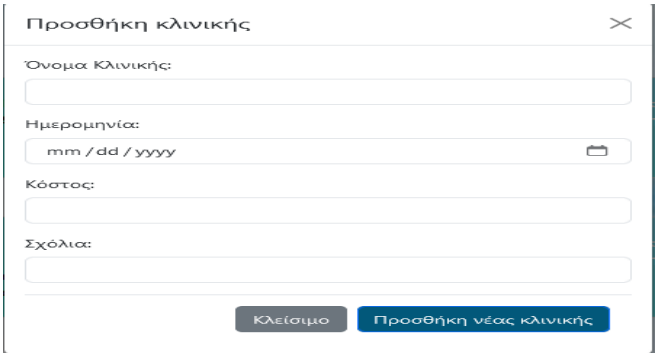
Στην επιλογή Clinics ο χρήστης βλέπει τις καταχωρημένες κλινικές που έχει επισκεφτεί και καταχωρήσει ο χρήστης. Στην συγκεκριμένη σελίδα εμφανίζονται πληροφορίες όπως:

- Όνομα της κλινικής
- Ημερομηνια
- Κοστος
- Σχόλια

ΕΙΚΟΝΑ 43:CLINICS

Στο πάνω μέρος της οθόνης υπάρχει το κουμπί «Προσθήκη κλινικής».

Στην Εικόνα 42 που αφορά την καταχώρηση νέας κλινικής εμφανίζονται τα πεδία



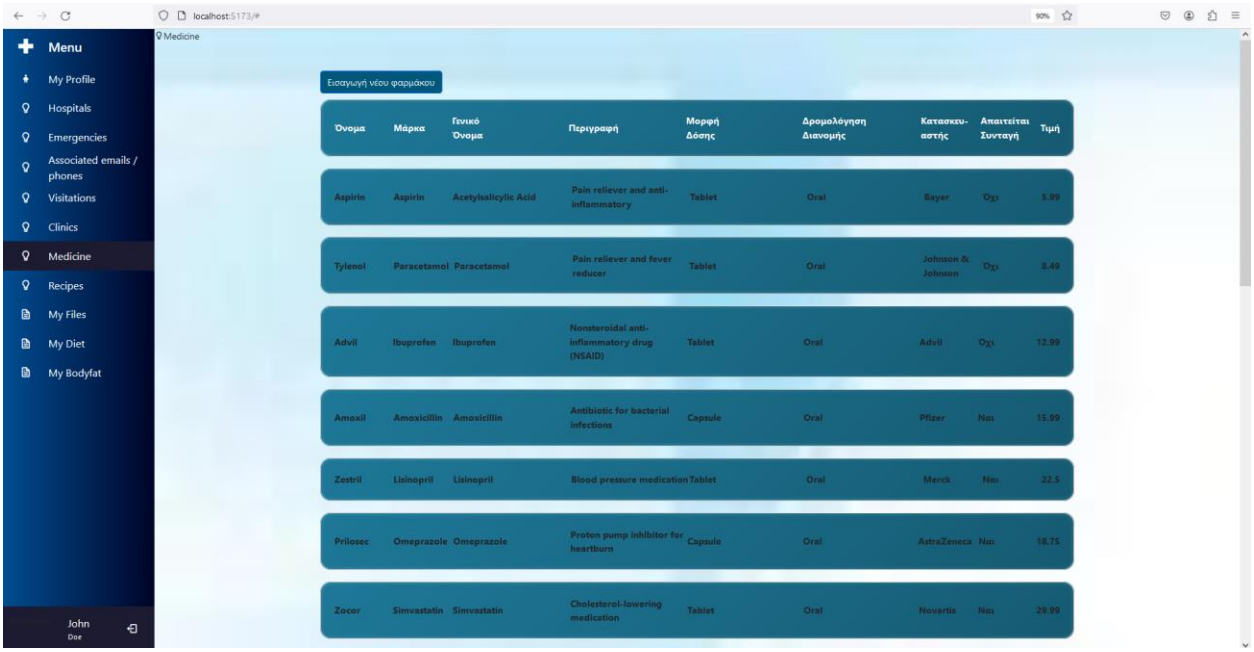
- Στο πεδίο Όνομα κλινικής ο χρηστης καταχωρεί είτε το όνομα του γιατρού, ειτε την διεύθυνση είτε οποιαδήποτε πληροφορία μπορει να ανακαλέσει σχετικά με την κληνική.
- Ημερομηνια υπάρχει ένα calendar ώστε να καταχωρηθεί σωστός τύπος ημερομηνίας.
- Κοστος είναι αλφαριθμιτικό.
- Σημειώσεις/ σχόλια

ΕΙΚΌΝΑ 44:ADD NEW CLINIC

Medicine new-medicine

Στην επιλογή Medicine ο χρήστης βλέπει τα καταχωρημένα φάρμακα που έχει λάβει στις συνταγές ή από το φαρμακείο . Στην συγκεκριμένη σελίδα εμφανίζονται πληροφορίες όπως:

- Όνομα
- Μάρκα
- Γενικό όνομα
- Περιγραφή
- Μορφή δόσης (ταμπλέτα, κάψουλα, υγρή μορφή)
- Δρομολόγιση διανομής (ή υπεύθυνος διανομής)
- Κατασκευαστής
- Αν απαιτείται συνταγή
- Τιμή

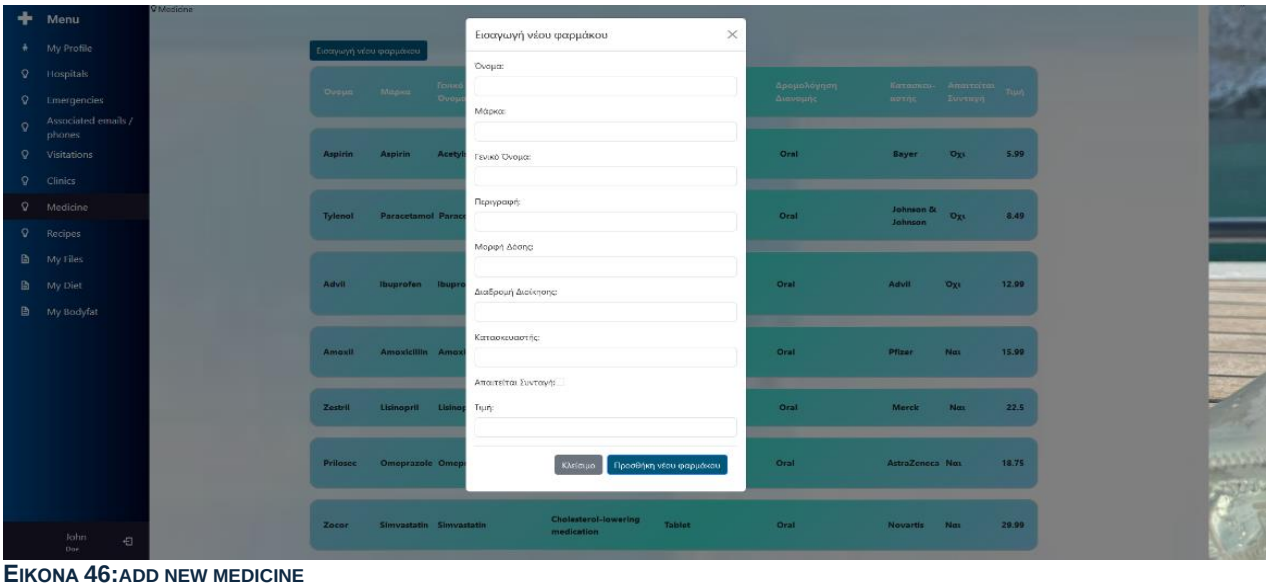


ΕΙΚΟΝΑ 45:MEDICINE

Στο πάνω μέρος της οθόνης υπάρχει το κουμπί «Καταχώρηση νέου φαρμάκου».

Στην Εικόνα 44 που αφορά την καταχώρηση νέου φαρμάκου εμφανίζονται τα πεδία

- Όνομα
- Μάρκα
- Γενικό όνομα
- Περιγραφή
- Μορφή δόσης (ταμπλέτα, κάψουλα, υγρή μορφή ή σε όποια άλλη μορφή μπορεί ο ασθενης να λάβει καποιο φάρμακο).
- Δρομολόγιση διανομής (ή υπεύθυνος διανομής)
- Κατασκευαστής
- Απαιτείται συνταγή checkbox
- Τιμή



ΕΙΚΟΝΑ 46:ADD NEW MEDICINE

Συμπεράσματα

Στις προηγούμενες ενότητες αναφέρθηκε η σύνθεση της τελικής υλοποίησης, η οποία εμπεριέχει την αλληλεπίδραση πολλών διαφορετικών τμημάτων που χρησιμοποιούν διαφορετικές τεχνολογίες. Αυτό απαιτεί εκτεταμένη εξοικείωση με διαφορετικές γλώσσες και συντακτικά. Παρά ταύτα, ο τρόπος με τον οποίο δομήθηκε η εφαρμογή παρέχει ουσιαστικά μιαν αρχιτεκτονική modular, η οποία είναι εύκολα φορητή και αναδιατάξιμη. Μπορεί να εξυπηρετήσει οποιοδήποτε front-end ή να λειτουργήσει ανεξάρτητα ως επίπεδο που παρέχει συγκεκριμένα APIs.

Επιπλέον, η εφαρμογή είναι εύκολα μεταφερόμενη στον τομέα των κινητών συσκευών, καθώς χρησιμοποιεί lightweight serverless βάση δεδομένων. Η μοναδική αλλαγή που θα απαιτούνταν θα ήταν στην εμφάνιση, ώστε να γίνει προσαρμοστική σε διάφορες οθόνες.

Επίσης, η εφαρμογή μας μπορεί εύκολα να αναπτυχθεί ως υπηρεσία Platform-as-a-Service σε cloud περιβάλλον, μέσω της χρήσης τεχνολογιών όπως το Docker ή το Docker Compose. Αυτό θα μας επιτρέψει να απολαύσουμε όλα τα οφέλη που προσφέρει ένας cloud deployment, όπως η ανθεκτικότητα, η υψηλή διαθεσιμότητα, η επεκτασιμότητα και η κατανομή του κόστους βάσει της χρήσης.

Βιβλιογραφία

- Alonistioti, Nancy, Evangelia Aikaterini Tsichrintzi, Konstantina Chrysafiadi, and Efthimios Alepis. 2023. “Requirements for Fuzzy Logic in Personalisation of Fire Emergency Alerts.” In 2023 14th International Conference on Information, Intelligence, Systems & Applications (IISA), 1–8. IEEE.

- Argyropoulos, Vasileios, Efthimios Alepis, and Constantinos Patsakis. 2022. “Semi-Decentralized File Sharing as a Service.” In 2022 13th International Conference on Information, Intelligence, Systems & Applications (IISA), 1–8. IEEE.

- Bilika, Domna, Nikoletta Michopoulou, Efthimios Alepis, and Constantinos Patsakis. “Hello Me, Meet the Real Me: Voice Synthesis Attacks on Voice Assistants.” Computers & Security 137: 103617.

- Douladiris, Anargyros, and Efthimios Alepis. 2023. “Covid-19 New Cases Correlation Analysis: Weather Conditions, Citizen Traffic and Vaccination Statistics Impact in NARX Estimated Regressions in Attica, Greece.” In 2023 14th International Conference on Information, Intelligence, Systems & Applications (IISA), 1–7. IEEE.

- Giannikis, Athanasios, Efthimios Alepis, and Maria Virvou. 2021. “Crowdsourcing Recognized Image Objects in Mobile Devices Through Machine Learning.” In 2021 IEEE 33rd International Conference on Tools with Artificial Intelligence (ICTAI), 560–67. IEEE.

- Kapetanios, Constantinos, Theodoros Polyzos, Efthimios Alepis, and Constantinos Patsakis. 2021. “This Is Just Metadata: From No Communication Content to User Profiling, Surveillance and Exploitation.” Advances in Core Computer Science-Based Technologies: Papers in Honor of Professor Nikolaos Alexandris, 277–302.

- Kontogianni, Aristea, and Efthimios Alepis. 2020. “Smart Tourism: State of the Art and Literature Review for the Last Six Years.” Array 6: 100020.

- ———. 2022. “AI, Blockchain & Cyber Tourism Joining the Smart Tourism Realm.” In 2022 13th International Conference on Information, Intelligence, Systems & Applications (IISA), 1–6. IEEE.

- ———. 2023. “Social Network Data Enabling Smart Tourism.” In 2023 14th International Conference on Information, Intelligence, Systems & Applications (IISA), 1–6. IEEE.

- Kontogianni, Aristea, Efthimios Alepis, and Constantinos Patsakis. 2022a. “Promoting Smart Tourism Personalised Services via a Combination of Deep Learning Techniques.” Expert Systems with Applications 187: 115964.

- ———. 2022b. “Smart Tourism and Artificial Intelligence: Paving the Way to the Post-Covid-19 Era.” Advances in Artificial Intelligence-Based Technologies: Selected Papers in Honour of Professor Nikolaos G. Bourbakis—Vol. 1, 93–109.

- Matzavela, Vasiliki, and Efthimios Alepis. 2021. “M-Learning in the COVID-19 Era: Physical Vs Digital Class.” Education and Information Technologies 26 (6): 7183–203.

- ———. 2023. “An Application of Self-Assessment of Students in Mathematics with Intelligent Decision Systems: Questionnaire, Design and Implementation at Digital Education.” Education and Information Technologies, 1–16.

- Michail, Tselepatiotis, and Efthimios Alepis. 2023. “Design of Real-Time Multiplayer Word Game for the Android Platform Using Firebase and Fuzzy Logic.” In 2023 14th International Conference

on Information, Intelligence, Systems & Applications (IISA), 1–8. IEEE.

- Patsakis, Constantinos, Eugenia Politou, Efthimios Alepis, and Julio Hernandez-Castro. 2023. “Cashing Out Crypto: State of Practice in Ransom Payments.” International Journal of Information Security, 1–14.

- Politou, Eugenia, Efthimios Alepis, Maria Virvou, and Constantinos Patsakis. 2022. “Privacy and Data Protection Challenges in the Distributed Era.” Springer.

- Politou, Eugenia, Efthimios Alepis, Maria Virvou, Constantinos Patsakis, Eugenia Politou, Efthimios Alepis, Maria Virvou, and Constantinos Patsakis. 2022a. “Open Questions and Future Directions.” Privacy and Data Protection Challenges in the Distributed Era, 175–80.

- ———. 2022b. “State-of-the-Art Technological Developments.” Privacy and Data Protection Challenges in the Distributed Era, 69–91.

- Sigala, Effrosyni, Efthimios Alepis, and Constantinos Patsakis. 2020. “Measuring the Quality of Street Surfaces in Smart Cities Through Smartphone Crowdsensing.” In 2020 11th International Conference on Information, Intelligence, Systems and Applications (IISA, 1–8. IEEE.

- Triantafyllou, Andreas M, George A Tsihrintzis, Maria Virvou, and Efthimios Alepis. 2021. “A Bimodal System for Emotion Recognition via Computer of Known or Unknown Persons in Normal or Fatigue Situations.” In Advances in Core Computer Science-Based Technologies, 9–35. Springer, Cham.

- Virvou, Maria, Efthimios Alepis, George A Tsihrintzis, and Lakhmi C Jain. 2020. “Machine Learning Paradigms: Advances in Learning Analytics.” Machine Learning Paradigms: Advances in Learning Analytics, 1–5.

- Docker. (n.d.). Docker Documentation. Retrieved from https://docs.docker.com/ This is the official documentation for Docker, providing comprehensive information on Docker containers, Docker Engine, Docker Compose, and related tools.

- Docker Compose. (n.d.). Docker Documentation. Retrieved from https://docs.docker.com/compose/ Official documentation for Docker Compose, offering guidance on how to define and run multi-container Docker applications.

- SQLite. (n.d.). SQLite Documentation. Retrieved from https://www.sqlite.org/docs.html. Official documentation for SQLite, including guides, references, and tutorials for utilizing SQLite in your project.

- Python Software Foundation. (n.d.). Python Documentation. Retrieved from https://docs.python.org/ Official documentation for Python programming language, providing guidance on Python syntax, libraries, and best practices.

- Vue.js. (n.d.). Vue.js Documentation. Retrieved from https://vuejs.org/v2/guide/ Official documentation for Vue.js, offering guides and API references for Vue.js framework, which you are using for frontend development.

- Mitchell, N., & Docker, J. (2019). Docker Deep Dive. Berkely, CA: Apress. This book provides an in-depth exploration of Docker concepts, architecture, and practical use cases, which can be helpful in understanding Docker for your project.

- Reitz, K., & Schlusser, T. (2018). The Docker Book: Containerization is the new virtualization. CreateSpace Independent Publishing Platform. This book offers practical insights and examples for using Docker in various scenarios, which can be beneficial for your Docker-related tasks.

- Grinberg, M. (2014). Flask Web Development: Developing Web Applications with Python. O'Reilly Media.