Εθνικο Μετσοβιο Πολυτεχνειο
Σχολη Ηλεκτρολογων Μηχανικων και Μηχανικων Υπολογιστων
Τομεας Μαθηματικων, Σχολη ΕΜΦΕ

Σχεδιαση και Αναπτυξη Διαδυκτιακου Βοηθηματος Λογοθεραπειας

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

της

Αθηνάς Ι. Μαυρομμάτη

Επιβλέπων: Στεφανέας Πέτρος
Αναπληρωτής Καθηγητής Ε.Μ.Π

Αθήνα, Οκτώβριος 2025

Εθνικο Μετσοβιο Πολυτεχνειο
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ
ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΜΑΘΗΜΑΤΙΚΩΝ, ΣΧΟΛΗ ΕΜΦΕ

ΣΧΕΔΙΑΣΗ ΚΑΙ ΑΝΑΠΤΥΞΗ ΔΙΑΔΥΚΤΙΑΚΟΥ ΒΟΗΘΗΜΑΤΟΣ ΛΟΓΟΘΕΡΑΠΕΙΑΣ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

της

Αθηνάς Ι. Μαυρομμάτη

Επιβλέπων: Στεφανέας Πέτρος
Αναπληρωτής Καθηγητής Ε.Μ.Π

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 16η Οκτωβρίου 2025.

.....	.....	.....
Πέτρος Στεφανέας Αναπληρωτής Καθηγητής Ε.Μ.Π	Αντώνιος Συμβώνης Καθηγητής Ε.Μ.Π	Δημήτριος Φωτάκης Καθηγητής Ε.Μ.Π

Αθήνα, Οκτώβριος 2025.

.....

Μαυρομμάτη Αθηνά
Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Copyright © Μαυρομμάτη Αθηνά, 2025

Με επιφύλαξη παντός δικαιώματος. All rights reserved.
Απαγορεύεται η αντιγραφή, αποθήκευση και διανομή της παρούσας εργασίας, εξ ολοκλήρου ή τμήματος αυτής, για εμπορικό σκοπό. Επιτρέπεται η ανατύπωση, αποθήκευση και διανομή για σκοπό μη κερδοσκοπικό, εκπαιδευτικής ή ερευνητικής φύσης, υπό την προϋπόθεση να αναφέρεται η πηγή προέλευσης και να διατηρείται το παρόν μήνυμα. Ερωτήματα που αφορούν τη χρήση της εργασίας για κερδοσκοπικό σκοπό πρέπει να απευθύνονται προς τον συγγραφέα.

Οι απόψεις και τα συμπεράσματα που περιέχονται σε αυτό το έγγραφο εκφράζουν τον συγγραφέα και δεν πρέπει να ερμηνευθεί ότι αντιπροσωπεύουν τις επίσημες θέσεις του Εθνικού Μετσόβιου Πολυτεχνείου.

Περίληψη

Η παρούσα διπλωματική εργασία πραγματεύεται τον σχεδιασμό και την ανάπτυξη μιας διαδικτυακής εφαρμογής λογοθεραπείας, με στόχο την υποστήριξη της εξ αποστάσεως καθοδήγησης και εξάσκησης ασθενών. Η αφορμή προήλθε από το διαπιστωμένο κενό στην ελληνική αγορά για ψηφιακά εργαλεία που προσφέρουν τη δυνατότητα στους λογοθεραπευτές να αναθέτουν εξατομικευμένα ασκησιολόγια στους ασθενείς τους, με τη χρήση πλούσιου οπτικοακουστικού υλικού.

Στο πλαίσιο της εργασίας, αξιοποιήθηκε το υλικό μιας προηγούμενης πρωτοβουλίας (ηχογραφήσεις ασκήσεων, εικόνες και περιγραφές), ώστε να υλοποιηθεί μια πλήρως λειτουργική web εφαρμογή με αρχιτεκτονική REST API. Η εφαρμογή επιτρέπει στον κλινικό να συνδέεται μέσω προσωπικού λογαριασμού, να διαχειρίζεται ασθενείς και να δημιουργεί/αναθέτει προσαρμοσμένα σετ ασκήσεων (bundles), ενώ ο ασθενής έχει τη δυνατότητα να εκτελεί τις ασκήσεις με σαφείς ηχογραφημένες οδηγίες και συνοδευτικό πολυμεσικό υλικό. Η αρχιτεκτονική της πλατφόρμας σχεδιάστηκε βάσει σύγχρονων προτύπων, χρησιμοποιώντας UML διαγράμματα , ενώ το σύστημα αναπτύχθηκε χρησιμοποιώντας σύγχρονα εργαλεία και τεχνολογίες για όλα τα επιμέρους τμήματα. Επιπλέον, διασφαλίστηκε η ανάπτυξη και η διάθεση της εφαρμογής σε παραγωγικό περιβάλλον μέσω Docker containers.

Λέξεις Κλειδιά

Λογοθεραπεία, Διαδικτυακή εφαρμογή, Εξατομικευμένο ασκησιολόγιο, REST API, HTTP, Frontend, Οπτικοακουστικό υλικό, UML διαγράμματα, Ανάθεση ασκήσεων, Backend, Βάση δεδομένων, Docker, κλινικός, ασθενής, διαχειριστής

Abstract

This thesis presents the design and development of a web-based application for speech therapy, aiming to support remote guidance and practice for patients. The motivation originated from the observed gap in the Greek market regarding digital tools that allow speech therapists to assign personalized exercise sets to their patients, utilizing rich audiovisual material and interactive functionalities.

Within the scope of the project, material from a previous initiative (exercise recordings, images, and descriptions) was used to implement a fully functional web application with a REST API architecture. The platform enables clinicians to log in with a personal account, manage patients, and create/assign customized exercise sets (bundles), while patients are able to perform the assigned exercises with clear audio instructions and accompanying multimedia content. The architecture of the platform was designed according to modern standards, utilizing UML diagrams, and the system was developed using up-to-date tools and technologies for all subsystems. Furthermore, deployment and delivery in a production environment was ensured through the use of Docker containers.

Keywords

Speech Therapy, Web Application, Personalized Exercise Sets, REST API, HTTP, Frontend, Audiovisual Material, UML Diagrams, Exercise Assignment, Backend, Database, Docker, Clinician, Patient, Administrator

Ευχαριστίες

Θα ήθελα να εκφράσω τις θερμές μου ευχαριστίες στον επιβλέποντα καθηγητή μου για την υποστήριξη και εμπιστοσύνη κατά τη διάρκεια της εκπόνησης της παρούσας διπλωματικής εργασίας.

Ευχαριστώ ιδιαίτερα την οικογένειά μου, που με στήριξε έμπρακτα σε κάθε μου βήμα, δίνοντάς μου δύναμη, στήριξη και κουράγιο σε όλες τις δυσκολίες.

Θα ήθελα επίσης να ευχαριστήσω τους compañeros συμφοιτητές μου για τη συντροφιά, τη συνεργασία και τη συμβολή τους στη φοιτητική μου εξέλιξη.

Ιδιαίτερες ευχαριστίες απευθύνω στους «iron lobby» φίλους μου για την ιδέα της διπλωματικής εργασίας, τον εφοδιασμό με συμβουλές, ιδέες και οπτικοακουστικό υλικό, καθώς και για τη φιλία και τη στήριξή τους όλα αυτά τα δέκα χρόνια.

Τέλος, ευχαριστώ όλους τους φίλους μου που στάθηκαν αληθινοί φίλοι και ήταν δίπλα μου σε κάθε στιγμή, καλή και δύσκολη, όλα αυτά τα χρόνια.

Περιεχόμενα

Κατάλογος Σχημάτων

Κατάλογος Πινάκων

Κεφάλαιο 1

Εισαγωγή

1.1 Η Σπουδαιότητα της Λογοθεραπείας

Αν ρωτούσαμε τον γενικό πληθυσμό ποια λειτουργία του σώματος θεωρεί πιο σημαντική για την καθημερινή ζωή, πιθανότατα η φωνή δεν θα ήταν η πρώτη απάντηση. Παρ' όλα αυτά, αποτελεί το πιο άμεσο και αυθόρμητο μέσο επικοινωνίας: με αυτήν εκφράζουμε ανάγκες, συναισθήματα, ιδέες, αλλά και την προσωπική μας ταυτότητα. Η καθημερινή πίεση, οι θορυβώδεις συνθήκες εργασίας και η συχνά επιπόλαιη χρήση της φωνής (π.χ. το συνεχές φώναγμα για να ακουστούμε) δημιουργούν μικροτραυματισμούς που, αν δεν αντιμετωπιστούν, οδηγούν σε σοβαρότερες διαταραχές.

Η λογοθεραπεία, ως επιστήμη της υγείας, έχει κεντρικό ρόλο στην αξιολόγηση, πρόληψη και αποκατάσταση των διαταραχών αυτών. Δεν περιορίζεται μόνο στη θεραπευτική αντιμετώπιση, αλλά στοχεύει και στη διαπαιδαγώγηση του ατόμου, ώστε να αναπτύξει υγιέστερες φωνητικές συνήθειες. Η φωνή αποτελεί θεμελιώδες μέσο επικοινωνίας για κάθε άνθρωπο, ενώ σε ορισμένα επαγγέλματα, όπως η διδασκαλία, η υποκριτική ή η τηλεφωνική εξυπηρέτηση, η σημασία της είναι ακόμη μεγαλύτερη. Η απώλεια ή η αλλοίωσή της επηρεάζει σε κάθε περίπτωση όχι μόνο την επαγγελματική πορεία, αλλά και την κοινωνική συμμετοχή του ατόμου.

Τα δεδομένα είναι ανησυχητικά: επιδημιολογικές μελέτες καταγράφουν ότι περίπου το 50% της πλειονότητας εμφανίζει κάποια μορφή φωνητικής διαταραχής κάθε χρόνο. Το φαινόμενο είναι πιο συχνό στις γυναίκες, ενώ σε επαγγέλματα με αυξημένη φωνητική καταπόνηση τα ποσοστά είναι ακόμη υψηλότερα. Ο συνδυασμός απαιτητικής εργασίας και έλλειψης εκπαίδευσης στη σωστή χρήση της φωνής αυξάνει σημαντικά τον κίνδυνο.

Οι συνέπειες δεν είναι μόνο βιολογικές· έχουν σαφές κοινωνικό, επαγγελματικό και ψυχολογικό αντίκτυπο. Το άτομο με διαταραχή φωνής μπορεί να αισθάνεται μειονεκτικά, να

περιορίζει την επικοινωνία του και να βλέπει την αποδοτικότητά του να φθίνει. Η συσσώρευση αυτών των παραγόντων οδηγεί σε μείωση ποιότητας ζωής, με συνέπεια πολλές φορές την απομόνωση.

Σε αυτό το πλαίσιο, η λογοθεραπεία λειτουργεί ως ουσιαστικός μοχλός αποκατάστασης και ενδυνάμωσης. Μέσα από στοχευμένα προγράμματα ασκήσεων, ο ασθενής μαθαίνει να προστατεύει τη φωνή του, να αναγνωρίζει τα λάθη στη χρήση της και να εφαρμόζει τεχνικές αυτοφροντίδας που μειώνουν τον κίνδυνο υποτροπής. Έτσι, η θεραπευτική διαδικασία δεν εξαντλείται στη διόρθωση μιας υφιστάμενης βλάβης, αλλά εξοπλίζει το άτομο με γνώσεις και δεξιότητες που το συνοδεύουν μακροπρόθεσμα.

1.2 Παρουσίαση του Προβλήματος

Παρά τη διαχρονική αξία της λογοθεραπείας ως βασική λύση για τις διαταραχές φωνής, η πρακτική άσκηση των ασκήσεων στο σπίτι συχνά αντιμετωπίζει σοβαρές δυσκολίες. Ο ασθενής λαμβάνει από τον λογοθεραπευτή ένα σετ οδηγιών και καλείται να επαναλαμβάνει τις ασκήσεις χωρίς καμία επιπλέον καθοδήγηση ή ανατροφοδότηση. Αυτό το «κενό επίβλεψης» μπορεί να οδηγήσει σε ασυνεπή εξάσκηση, λανθασμένη εκτέλεση ή ακόμη και σε σταδιακή εγκατάλειψη του προγράμματος.

Σήμερα, υπάρχουν αρκετές ξενόγλωσσες εφαρμογές που παρέχουν ενδεικτικές εικόνες ή βίντεο οδηγιών για ασκήσεις φωνής, αλλά καμία δεν προσφέρει τη δυνατότητα στον κλινικό να οργανώσει και να αναθέσει εξατομικευμένο ασκησιολόγιο στον κάθε ασθενή στα ελληνικά. Δεν υπάρχει, δηλαδή, ελληνική εφαρμογή σε μορφή ασκησιολογίου όπου ο λογοθεραπευτής μπορεί να καταρτίζει, να διαχειρίζεται και να προσαρμόζει συγκεκριμένες ασκήσεις για τον ασθενή του, μέσα από ένα ενιαίο περιβάλλον.

Αυτό το κενό υπογραμμίζει την ανάγκη για ένα εύχρηστο ψηφιακό εργαλείο που θα γεφυρώνει το χάσμα μεταξύ συνεδρίας και αυτοτελούς εξάσκησης. Μια πλατφόρμα που θα επιτρέπει στον λογοθεραπευτή να αναθέτει προσεκτικά επιλεγμένες ασκήσεις, προσαρμοσμένες στις ανάγκες κάθε χρήστη, και ταυτόχρονα θα παρέχει στον ασθενή ξεκάθαρη παρουσίαση με οπτικοακουστικό υλικό, καθιστώντας την εξάσκηση πιο ευχάριστη και αποτελεσματική.

1.3 Προτεινόμενη Λύση του Παρελθόντος: Δημιουργία Διαδικτυακής Εφαρμογής Λογοθεραπείας

Στο πλαίσιο ερευνητικής πρωτοβουλίας φοιτητών του Τμήματος Λογοθεραπείας του Πανεπιστημίου Πατρών, αναπτύχθηκε ένα οργανωμένο σύνολο περίπου 40 ασκήσεων λογοθεραπείας. Κάθε άσκηση ηχογραφήθηκε με φυσική φωνή φοιτητή, ώστε ο ασθενής να ακούει προφορική καθοδήγηση και να εκτελεί με ακρίβεια τις οδηγίες. Η επιλογή της φυσικής ηχογράφησης, αντί

για συνθετική ανάγνωση μέσω συστημάτων τεχνητής νοημοσύνης, δεν είναι τυχαία: η λογοθεραπεία απαιτεί αυθεντική φωνητική επαφή με το άτομο, καθώς η ανθρώπινη χροιά μεταφέρει ρυθμό, έμφαση και εκφραστικότητα που είναι ουσιώδη για τη σωστή εκτέλεση των ασκήσεων. Με αυτόν τον τρόπο διασφαλίζεται ότι το περιβάλλον μάθησης προσεγγίζει περισσότερο τις πραγματικές συνθήκες επικοινωνίας.

Παράλληλα, ο Μαρίος Κατσίκης από το Τμήμα Μηχανικών Σχεδίασης Προϊόντων και Συστημάτων του Πανεπιστημίου Αιγαίου δημιούργησε εικονογραφικές απεικονίσεις για όλα τα βήματα των ασκήσεων, παρέχοντας οπτικό υλικό που διευκολύνει την κατανόηση και την εκτέλεση από τον χρήστη . Ο Ηλίας Μαγγίνας από το Τμήμα Πληροφορικής του Οικονομικού Πανεπιστημίου Αθηνών ανέπτυξε μια πιλοτική εφαρμογή ασκησιολογίου, μέσα στην οποία οι ασκήσεις παρουσιάζονταν οργανωμένα, χωρίς όμως να υποστηρίζεται σύστημα σύνδεσης, διαχείρισης χρηστών ή αποθήκευσης προόδου. Ήταν, δηλαδή, μια αρχική, μη ολοκληρωμένη εφαρμογή που στόχευε αποκλειστικά στην παρουσίαση των ασκήσεων, χωρίς λειτουργίες εξατομίκευσης ή παρακολούθησης από τον κλινικό .

Αν και απουσίαζαν κρίσιμα χαρακτηριστικά διαχείρισης χρηστών και εξατομίκευσης, η δοκιμαστική εφαρμογή απέδειξε την αξία ενός ψηφιακού εργαλείου ικανή να οργανώνει και να προβάλλει το πλήρες πρόγραμμα άσκησης του ασθενή, θέτοντας τα θεμέλια για τη μελλοντική υλοποίηση μιας πλήρως λειτουργικής πλατφόρμας.

1.4 Ανατροφοδότηση Λογοθεραπευτών

Η ενδεικτική εφαρμογή υποβλήθηκε σε αξιολόγηση από 8 επαγγελματίες λογοθεραπευτές, με πάνω από 10 χρόνια εμπειρίας, οι οποίοι κλήθηκαν να χρησιμοποιήσουν το δοσμένο ασκησιολόγιο και να καταγράψουν τις εντυπώσεις τους. Από αυτούς, το 75% δήλωσε ότι θα σύστηνε την εφαρμογή και σε άλλους συναδέλφους, επισημαίνοντας την πρακτικότητα της προσέγγισης και την ευκολία στην παρουσίαση των ασκήσεων. Επιπλέον, το 60% διαφόρων αξιολογητών, σχετικών και μη του κλάδου, θεώρησε τις ασκήσεις αποτελεσματικές σε ό,τι αφορά την ποικιλία, την κατανόηση των οδηγιών και την καταλληλότητα για καθημερινή εξάσκηση, επισημαίνοντας μόνο τη βελτίωση της αισθητικής και της διάταξης των εικόνων ως κύριο σημείο ανατροφοδότησης.

Οι παρατηρήσεις αυτές υπογραμμίζουν την ανάγκη για περαιτέρω ανάπτυξη λειτουργιών διαχείρισης χρήστη και εξατομίκευσης, αλλά επιβεβαιώνουν την αξία ενός εργαλείου που φέρνει την εξειδικευμένη καθοδήγηση του λογοθεραπευτή στο σπίτι του ασθενή, με σαφείς ηχογραφημένες οδηγίες και οπτικοακουστικό υλικό.

1.5 Σκοπός της Διπλωματικής Εργασίας

Σκοπός της παρούσας διπλωματικής εργασίας είναι να προχωρήσει ένα βήμα παραπέρα την προσπάθεια των φοιτητών που ανέπτυξαν την αρχική πρωτοτυπη εφαρμογή, αξιοποιώντας το σύνολο των δεδομένων τους —εικόνες, ηχογραφήσεις και περιγραφές ασκήσεων— για τη δημιουργία μιας πλήρως λειτουργικής web εφαρμογής με REST API. Στην εφαρμογή αυτή, ο κλινικός θα έχει τη δυνατότητα να συνδέεται μέσω προσωπικού λογαριασμού και να αναθέτει εξατομικευμένα ασκησιολόγια στους ασθενείς του. Ο ασθενής, με τη σειρά του, θα μπορεί να συνδέεται και να βλέπει με σαφήνεια τις ασκήσεις που του έχουν ανατεθεί, προκειμένου να τις εκτελεί με τη βοήθεια του οπτικοακουστικού υλικού που έχει ήδη δημιουργηθεί.

Στο πλαίσιο της διπλωματικής εργασίας θα σχεδιαστεί η αρχιτεκτονική της εφαρμογής μέσω κατάλληλων UML διαγραμμάτων (Use Case, Class, Sequence κ.ά.), θα επιλεγούν οι απαιτούμενες τεχνολογίες για το front-end, το back-end και τη βάση δεδομένων, και θα υλοποιηθεί η διάθεσή της σε παραγωγικό περιβάλλον χρησιμοποιώντας Docker containers. Με αυτόν τον τρόπο, επιδιώκεται η ανάπτυξη ενός ολοκληρωμένου ψηφιακού εργαλείου που θα υποστηρίζει τόσο τους λογοθεραπευτές όσο και τους ασθενείς τους σε πραγματικές συνθήκες κλινικής εφαρμογής.

1.6 Δομή Διπλωματικής Εργασίας

Η διπλωματική εργασία είναι οργανωμένη σε έξι κεφάλαια.

Τα Κεφάλαια 2 και 3 αποτελούν το θεωρητικό υπόβαθρο της εργασίας. Στο δεύτερο κεφάλαιο παρουσιάζεται το πρωτόκολλο HTTP, η λειτουργία του και οι βασικές μέθοδοι και μηχανισμοί του. Στο τρίτο κεφάλαιο εξετάζονται τα REST APIs, οι αρχές σχεδίασής τους και οι τρόποι αλληλεπίδρασης με εφαρμογές.

Στο Κεφάλαιο 4 παρουσιάζεται η ανάλυση των απαιτήσεων και η σχεδίαση του συστήματος. Περιγράφονται οι λειτουργικές και μη λειτουργικές απαιτήσεις, η αρχιτεκτονική δομή καθώς και οι βασικές επιλογές σχεδίασης που καθόρισαν την πορεία της υλοποίησης.

Το Κεφάλαιο 5 παρουσιάζει την εφαρμογή LogiCare μέσα από χαρακτηριστικά στιγμιότυπα οθόνης, αναδεικνύοντας τη λειτουργικότητα και την ευχρηστία της για ασθενείς, κλινικούς και διαχειριστές.

Στο Κεφάλαιο 6 διενεργείται μια σύντομη αναφορά στα συμπεράσματα και τις μελλοντικές επεκτάσεις αυτής της εργασίας.

Κεφάλαιο 2

Το Πρωτόκολλο HTTP

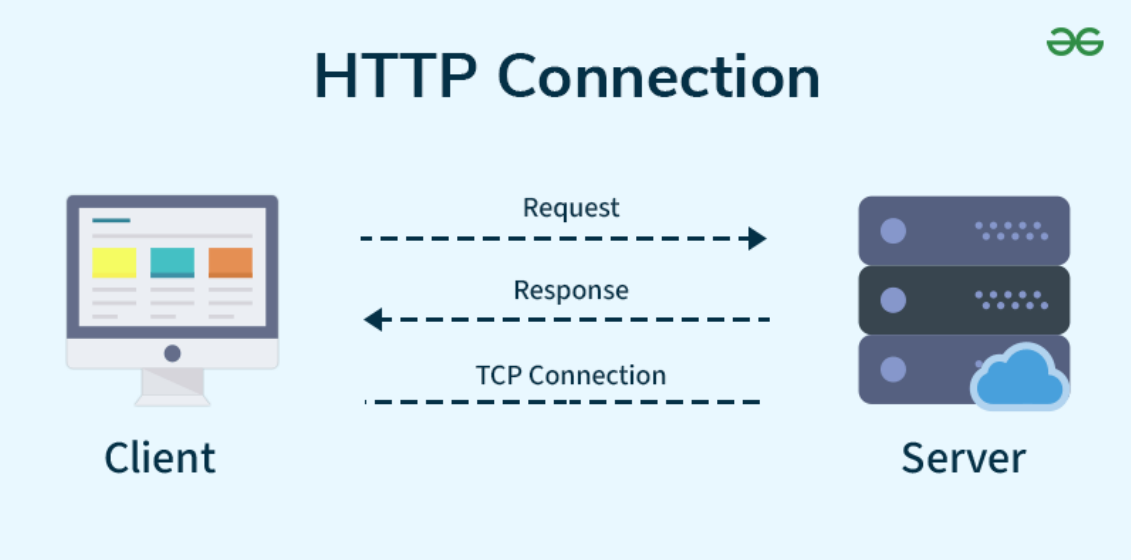
2.1 Ιστορική αναδρομή και ορισμός

Το Hypertext Transfer Protocol (HTTP) είναι το πρωτόκολλο που χρησιμοποιείται για την επικοινωνία στον Παγκόσμιο Ιστό. Εμφανίστηκε το 1991 από τον Tim Berners-Lee στο CERN, ως μέρος του αρχικού συστήματος του World Wide Web. Αποτελεί πρωτόκολλο αιτήσεων–απαντήσεων (request–response) και χρησιμοποιείται για τη μεταφορά όχι μόνο υπερκειμένου (HTML), αλλά και κάθε είδους δεδομένων όπως εικόνες, ήχο και δομημένα έγγραφα (JSON, XML κ.ά.). Το HTTP χαρακτηρίζεται ως *stateless*: κάθε αίτηση είναι ανεξάρτητη και ο εξυπηρετητής δεν θυμάται προηγούμενες αλληλεπιδράσεις.

2.2 Μοντέλο λειτουργίας

Το HTTP ακολουθεί το μοντέλο client–server. Ο πελάτης (π.χ. browser ή εφαρμογή) στέλνει μια αίτηση, και ο εξυπηρετητής την επεξεργάζεται και απαντά με κατάλληλο μήνυμα απόκρισης. Η ροή συνοψίζεται ως εξής:

1. Ο πελάτης ανοίγει σύνδεση (HTTPS, δηλαδή HTTP πάνω από TLS/TCP).
2. Στέλνει μήνυμα αίτησης (HTTP request).
3. Ο εξυπηρετητής το επεξεργάζεται (π.χ. προσπελαύνει βάση δεδομένων).
4. Επιστρέφει μήνυμα απόκρισης (HTTP response).



Σχήμα 2.1: Ροή αιτήσεων–απαντήσεων στο πρωτόκολλο HTTP.[1]

2.3 Μέθοδοι HTTP

Οι μέθοδοι (HTTP methods ή verbs) δηλώνουν την πρόθεση του πελάτη ως προς τον πόρο που στοχεύει. Κάθε μέθοδος έχει συγκεκριμένη σημασιολογία, και ορίζεται αν είναι *ασφαλής* (safe) ή/και *idempotent*, ιδιότητες που είναι κρίσιμες για τον σχεδιασμό εφαρμογών.

Επεξηγήσεις ιδιοτήτων:

- **Body**: δηλώνει αν η μέθοδος επιτρέπει αποστολή δεδομένων στο σώμα της αίτησης (HTTP request body). Για παράδειγμα, το POST στέλνει φόρμες ή JSON δεδομένα, ενώ το GET δεν έχει σώμα.
- **Safe**: μια μέθοδος θεωρείται ασφαλής όταν δεν μεταβάλλει την κατάσταση του εξυπηρετητή, αλλά περιορίζεται σε ανάγνωση πληροφοριών. Οι μέθοδοι GET, HEAD και OPTIONS είναι *safe*.
- **Idempotent**: μια μέθοδος είναι *idempotent* όταν η επανάληψή της πολλές φορές με τα ίδια δεδομένα οδηγεί στο ίδιο αποτέλεσμα. Για παράδειγμα, η PUT αντικαθιστά πάντα τον πόρο με την ίδια αναπαράσταση, ενώ η DELETE αφαιρεί τον πόρο ανεξάρτητα από το πόσες φορές κληθεί.

Αναπαριστούμε τις βασικές μεθόδους HTTP στον ακόλουθο πίνακα:

Μέθοδος	**Σκοπός / Παράδειγμα**	**Body**	**Safe**	**Idempotent**
GET	Ανάκτηση πόρου. Παράδειγμα: φόρτωση μιας ιστοσελίδας ή λίστας άρθρων.	Όχι	Ναι	Ναι
POST	Δημιουργία νέου πόρου ή εκτέλεση ενέργειας. Παράδειγμα: υποβολή φόρμας εγγραφής.	Ναι	Όχι	Όχι
PUT	Πλήρης αντικατάσταση ενός πόρου με νέα αναπαράσταση. Παράδειγμα: ενημέρωση ολόκληρου προφίλ χρήστη.	Ναι	Όχι	Ναι
PATCH	Μερική τροποποίηση υπάρχοντος πόρου. Παράδειγμα: αλλαγή μόνο του email στο προφίλ χρήστη.	Ναι	Όχι	Όχι (συνήθως)
DELETE	Διαγραφή πόρου. Παράδειγμα: αφαίρεση ενός σχολίου.	Όχι	Όχι	Ναι
HEAD	Όπως το GET, αλλά επιστρέφει μόνο κεφαλίδες. Παράδειγμα: έλεγχος αν υπάρχει αρχείο στον server.	Όχι	Ναι	Ναι
OPTIONS	Επιστρέφει τις διαθέσιμες μεθόδους και δυνατότητες για έναν πόρο.	Όχι	Ναι	Ναι

Πίνακας 2.1: Σύγκριση βασικών μεθόδων HTTP.

Η κατανόηση αυτών των ιδιοτήτων είναι σημαντική για τον σωστό σχεδιασμό RESTful APIs. Για παράδειγμα, οι GET και HEAD αιτήσεις μπορούν να γίνουν cacheable από ενδιάμεσους κόμβους, ενώ οι μέθοδοι PUT και DELETE, όντας idempotent, επιτρέπουν την επανάληψη αιτήσεων χωρίς απροσδόκητα αποτελέσματα [2].

2.4 HTTP Responses

Κάθε μήνυμα απόκρισης (HTTP response) που επιστρέφει ένας εξυπηρετητής περιλαμβάνει τρία βασικά τμήματα:

- **Status line**: περιλαμβάνει την έκδοση του HTTP και τον κωδικό κατάστασης με συνοδευτικό μήνυμα (π.χ. `HTTP/1.1 200 OK`).
- **Headers**: μεταδεδομένα με τη μορφή `΄Ονομα: Τιμή`, που δίνουν πληροφορίες για την απόκριση (π.χ. `Content-Type`, `Content-Length`).
- **Body**: το κύριο περιεχόμενο της απόκρισης, το οποίο μπορεί να είναι κείμενο HTML, έγγραφο JSON, εικόνα, ή άλλο μέσο.

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 123

{ "id": 42, "title": "..." }
```

Σχήμα 2.2: Παράδειγμα HTTP απόκρισης

Στο πλαίσιο των Web APIs, το σώμα της απόκρισης περιέχει συνήθως JSON έγγραφα, τα οποία είναι εύκολο να καταναλωθούν και να επεξεργαστούν από εφαρμογές πελάτη.

2.5 HTTP Status Codes

Οι κωδικοί κατάστασης (status codes) χωρίζονται σε πέντε βασικές κατηγορίες, ανάλογα με το πρώτο ψηφίο. Κάθε κατηγορία καλύπτει ένα εύρος τιμών και εκφράζει διαφορετικό τύπο πληροφορίας ή σφάλματος [3].

Αναλυτικά:

Κατηγορία	**Εύρος**	**Σημασία**	**Ενδεικτικοί κωδικοί**
Informational	100–199	Ο εξυπηρετητής έλαβε την αίτηση και συνεχίζει την επεξεργασία.	`100 Continue`: ο πελάτης μπορεί να συνεχίσει. `101 Switching Protocols`: αλλαγή πρωτοκόλλου.
Success	200–299	Η αίτηση ολοκληρώθηκε επιτυχώς.	`200 OK`: επιτυχής ανάκτηση. `201 Created`: δημιουργήθηκε νέος πόρος. `204 No Content`: επιτυχία χωρίς περιεχόμενο.
Redirection	300–399	Απαιτείται περαιτέρω ενέργεια από τον πελάτη, συνήθως νέα αίτηση σε άλλο URL.	`301 Moved Permanently`: ο πόρος μετακινήθηκε μόνιμα. `302 Found`: προσωρινή μετακίνηση. `304 Not Modified`: τα δεδομένα δεν έχουν αλλάξει.
Client Error	400–499	Σφάλμα στην πλευρά του πελάτη: λάθος αίτηση ή μη εξουσιοδοτημένη πρόσβαση.	`400 Bad Request`: κακή σύνταξη αίτησης. `401 Unauthorized`: απαιτείται ταυτοποίηση. `403 Forbidden`: απαγόρευση πρόσβασης. `404 Not Found`: πόρος δεν βρέθηκε. `429 Too Many Requests`: υπέρβαση ορίου αιτήσεων.
Server Error	500–599	Ο εξυπηρετητής απέτυχε να ικανοποιήσει μια σωστή αίτηση λόγω προβλήματος στην πλευρά του.	`500 Internal Server Error`: γενικό σφάλμα. `502 Bad Gateway`: κακή απόκριση από ενδιάμεσο σερερ. `503 Service Unavailable`: μη διαθέσιμη υπηρεσία. `504 Gateway Timeout`: λήξη χρονικού ορίου.

Πίνακας 2.2: Κατηγορίες κωδικών κατάστασης HTTP με ενδεικτικά παραδείγματα.

2.6 HTTP Cookies

Το HTTP είναι από τη φύση του *stateless* πρωτόκολλο: κάθε αίτηση–απόκριση είναι ανεξάρτητη και ο εξυπηρετητής δεν θυμάται την προηγούμενη κατάσταση του πελάτη. Για να ξεπεραστεί αυτός ο περιορισμός, χρησιμοποιούνται τα cookies.

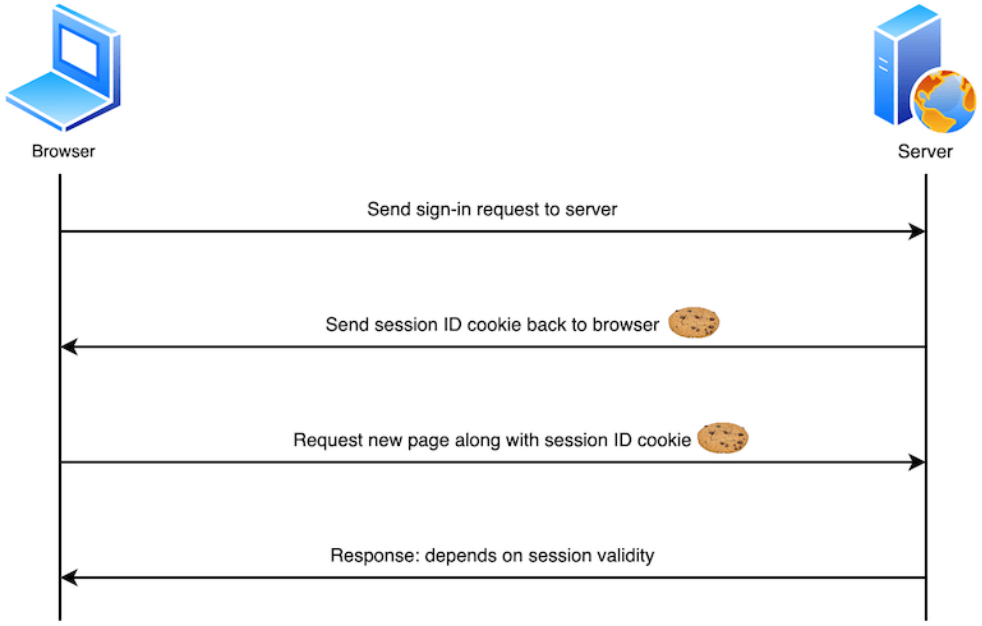
Ένα cookie είναι ένα μικρό κομμάτι δεδομένων που ο εξυπηρετητής στέλνει στον φυλλομετρητή (browser) του χρήστη. Ο φυλλομετρητής το αποθηκεύει και το επιστρέφει αυτόματα σε κάθε επόμενη αίτηση προς τον ίδιο εξυπηρετητή. Έτσι επιτυγχάνεται η διατήρηση πληροφορίας (*state*) ανάμεσα σε διαφορετικές αιτήσεις.

Τα cookies όντας μικρά τμημάτα δεδομένων συνοδεύονται από περιορισμούς. Έχουν μικρό μέγεθος (τυπικά έως 4 KB), γεγονός που τα καθιστά κατάλληλα μόνο για την αποθήκευση κρίσιμων δεδομένων. Επιπλέον, αποστέλλονται σε κάθε αίτηση προς τον εξυπηρετητή, κάτι που ενδέχεται να επιβαρύνει το δίκτυο όταν χρησιμοποιούνται για δεδομένα που δεν είναι απαραίτητα.

Η διάρκεια ζωής ενός cookie καθορίζεται από τις παραμέτρους του. Τα session cookies λήγουν με το κλείσιμο του φυλλομετρητή, ενώ τα persistent cookies έχουν καθορισμένη χρονική ισχύ που ορίζεται μέσω των πεδίων `Expires` ή `Max-Age`, με το δεύτερο να θεωρείται προτιμητέο καθώς βασίζεται στον χρόνο του φυλλομετρητή. Η σωστή διαχείριση της διάρκειας είναι απαραίτητη για την αποτροπή επιθέσεων, όπως η λεγόμενη *session fixation*, όπου ένας επιτιθέμενος επιχειρεί να «επιβάλει» συγκεκριμένο αναγνωριστικό συνεδρίας στον χρήστη.

2.6.1 Παράδειγμα χρήσης cookies για διαχείριση συνεδρίας

Για να γίνει πιο κατανοητός ο τρόπος λειτουργίας των cookies, παρουσιάζεται το ακόλουθο τυπικό παράδειγμα ροής σύνδεσης χρήστη. Στην Εικόνα 2.3 απεικονίζεται η επικοινωνία μεταξύ φυλλομετρητή και εξυπηρετητή κατά τη διαδικασία σύνδεσης.



Σχήμα 2.3: Ροή διαχείρισης συνεδρίας μέσω cookies.

Η αλληλουχία των βημάτων είναι η εξής:

1. Ο χρήστης εισάγει τα διαπιστευτήριά του (credentials) και ο φυλλομετρητής αποστέλλει αίτηση σύνδεσης προς τον εξυπηρετητή.
2. Εφόσον τα στοιχεία είναι έγκυρα, ο εξυπηρετητής δημιουργεί μια νέα συνεδρία και επιστρέφει στον φυλλομετρητή ένα cookie που περιέχει το μοναδικό session ID.
3. Σε κάθε επόμενη αίτηση (π.χ. φόρτωση νέας σελίδας), ο φυλλομετρητής συμπεριλαμβάνει αυτόματα το cookie με το session ID.
4. Ο εξυπηρετητής ελέγχει την ισχύ του session και, εφόσον είναι ενεργό, απαντά με εξατομικευμένο περιεχόμενο· διαφορετικά επιστρέφει μήνυμα σφάλματος ή ζητά νέα σύνδεση.[4]

2.7 Σύνδεση με τα REST APIs

Η γνώση του πρωτοκόλλου HTTP δεν περιορίζεται στη βασική κατανόηση της επικοινωνίας στον Ιστό· αποτελεί τη βάση πάνω στην οποία έχουν σχεδιαστεί τα REST APIs. Οι μέθοδοι, οι κωδικοί κατάστασης και οι κεφαλίδες που παρουσιάστηκαν προηγουμένως χρησιμοποιούνται αυτούσια για την υλοποίηση και την κατανάλωση διαδικτυακών υπηρεσιών. Έτσι, η σε βάθος κατανόηση του HTTP είναι απαραίτητη προϋπόθεση για την κατανόηση των αρχών και της σημασίας των REST APIs, τα οποία θα αποτελέσουν το αντικείμενο της επόμενης ενότητας.

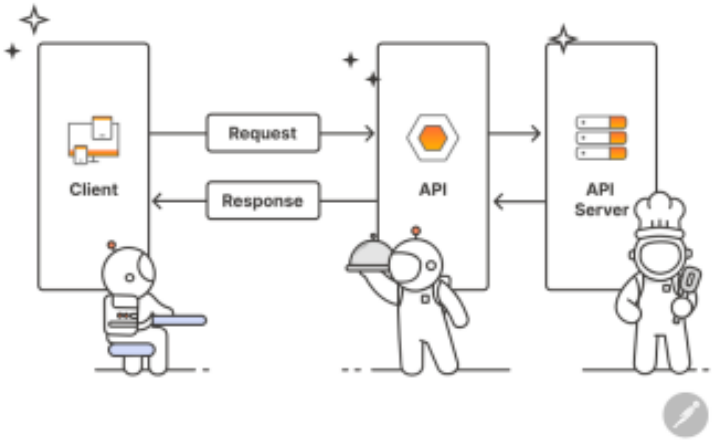
Κεφάλαιο 3

REST APIs

3.1 Εισαγωγή στα APIs

Ο όρος API (Application Programming Interface) αναφέρεται σε ένα σύνολο από κανόνες, πρωτόκολλα και εργαλεία που επιτρέπουν σε διαφορετικά κομμάτια λογισμικού να επικοινωνούν μεταξύ τους [5]. Με απλά λόγια, ένα API λειτουργεί ως «γέφυρα» ανάμεσα σε δύο συστήματα: παρέχει ένα καλά ορισμένο σύνολο μεθόδων και συμβάσεων που επιτρέπει σε έναν προγραμματιστή να αξιοποιεί λειτουργίες μιας υπηρεσίας χωρίς να γνωρίζει τις εσωτερικές λεπτομέρειες της υλοποίησής της.

Η χρησιμότητα των APIs στη σύγχρονη ανάπτυξη λογισμικού είναι καθοριστική. Μέσα από αυτά, οι εφαρμογές μπορούν να συνδυάζουν δεδομένα και λειτουργίες από διαφορετικές πηγές [6]. Ένας ιστότοπος μπορεί να αντλεί χάρτες από την πλατφόρμα Google Maps, ενώ μια εφαρμογή ηλεκτρονικού εμπορίου μπορεί να αξιοποιεί εξωτερικά APIs για την επεξεργασία πληρωμών ή την παρακολούθηση αποστολών. Με τον τρόπο αυτό επιτυγχάνεται η επαναχρησιμοποίηση λογισμικού, η μείωση κόστους ανάπτυξης και η ευκολότερη διασύνδεση ετερογενών συστημάτων.



Σχήμα 3.1: Σχέση πελάτη–API–εξυπηρετητή.

Όπως φαίνεται στην Εικόνα 3.1, ο πελάτης υποβάλλει αιτήσεις (requests) μέσω του API, το οποίο λειτουργεί ως μεσάζοντας για την επικοινωνία με τον εξυπηρετητή. Ο εξυπηρετητής επεξεργάζεται το αίτημα και επιστρέφει την κατάλληλη απόκριση (response) στον πελάτη.

3.2 Ιστορική αναδρομή

Η χρήση των APIs δεν είναι νέα· εμφανίστηκαν ήδη από τη δεκαετία του 1960, όταν οι προγραμματιστές αναζητούσαν τρόπους να απλοποιήσουν την αλληλεπίδραση μεταξύ διαφορετικών προγραμμάτων λογισμικού. Ωστόσο, η πραγματική τους ανάδειξη ήρθε στα τέλη της δεκαετίας του 1990 και στις αρχές του 2000, με την εξάπλωση του Διαδικτύου και των διαδικτυακών εφαρμογών. Η εισαγωγή των πρώτων Web APIs, όπως το SOAP (Simple Object Access Protocol), και αργότερα το REST (Representational State Transfer), άλλαξε ριζικά τον τρόπο με τον οποίο αναπτύσσονται και επικοινωνούν οι εφαρμογές στον Ιστό.

Η εμφάνιση του REST αποτέλεσε σημαντικό ορόσημο, καθώς προσέφερε μια απλούστερη και πιο ευέλικτη εναλλακτική στο SOAP, επιτρέποντας ταχύτερη και πιο εύκολη ανάπτυξη διαδικτυακών υπηρεσιών μέσω της αξιοποίησης του πρωτοκόλλου HTTP. Αργότερα, με την έλευση των microservices και των υποδομών cloud, τα APIs απέκτησαν ακόμα μεγαλύτερη σημασία, καθώς επέτρεψαν στις επιχειρήσεις να υιοθετήσουν πιο ευέλικτες και κλιμακούμενες αρχιτεκτονικές.

Σήμερα, τα APIs αποτελούν αναπόσπαστο κομμάτι του τεχνολογικού οικοσυστήματος, επιτρέποντας την άμεση ενσωμάτωση και συνεργασία μεταξύ εφαρμογών, υπηρεσιών και συστημάτων, από τις εφαρμογές κινητών τηλεφώνων έως τη διαχείριση πολύπλοκων εταιρικών πλατφορμών. [7]

3.3 Ορισμός REST & Βασικές Αρχές

Το REST (Representational State Transfer) είναι ένα αρχιτεκτονικό στυλ που προτάθηκε από τον Roy Fielding το 2000 για τον σχεδιασμό κατανεμημένων συστημάτων και διαδικτυακών υπηρεσιών. Αν και δεν αποτελεί τυπικό πρότυπο, περιγράφει ένα σύνολο αρχών που, αν τηρηθούν, καθιστούν μια υπηρεσία *RESTful* [8], [9].

Ο πυρήνας της ιδέας είναι ότι η επικοινωνία βασίζεται σε *πόρους* (resources), οι οποίοι ταυτοποιούνται μοναδικά μέσω URIs. Η αλληλεπίδραση με τους πόρους γίνεται με χρήση τυποποιημένων μεθόδων του HTTP (GET, POST, PUT, DELETE κ.ά.), ενώ τα δεδομένα ανταλλάσσονται σε απλές και ευρέως υποστηριζόμενες μορφές, όπως JSON ή XML.

3.3.1 Βασικά REST Constraints

Σύμφωνα με τη βιβλιογραφία, οι βασικοί περιορισμοί (constraints) του REST είναι οι ακόλουθοι:

1. **Ομοιόμορφη διεπαφή (Uniform Interface)**: Όλοι οι πόροι εκτίθενται μέσω μιας κοινής και συνεπούς διεπαφής. Αυτό σημαίνει ότι κάθε πόρος ταυτοποιείται με μοναδικό URI, οι αλληλεπιδράσεις γίνονται μέσω τυποποιημένων μεθόδων HTTP, και τα μηνύματα είναι αυτάρκη και κατανοητά χωρίς εξωτερικό πλαίσιο.
2. **Statelessness**: Κάθε αίτηση από τον πελάτη πρέπει να περιέχει όλες τις απαραίτητες πληροφορίες για την επεξεργασία της. Ο εξυπηρετητής δεν διατηρεί καμία πληροφορία σχετικά με προηγούμενες αιτήσεις, γεγονός που βελτιώνει την επεκτασιμότητα και την αξιοπιστία.
3. **Αρχιτεκτονική Client–Server**: Ο πελάτης και ο εξυπηρετητής είναι πλήρως διαχωρισμένοι και επικοινωνούν μόνο μέσω δικτύου. Δεν βρίσκονται στο ίδιο σύστημα ή στο ίδιο αρχείο, αλλά ανταλλάσσουν μηνύματα μέσω αιτήσεων και αποκρίσεων. Αυτός ο σαφής διαχωρισμός ρόλων επιτρέπει σε κάθε πλευρά να εξελίσσεται ανεξάρτητα: η ομάδα που αναπτύσσει το frontend (client) μπορεί να επικεντρώνεται στην εμπειρία χρήστη, ενώ η ομάδα του backend (server) χειρίζεται τη διαχείριση δεδομένων και την επιχειρησιακή λογική. Έτσι, το σύστημα αποκτά μεγαλύτερη ευελιξία και μπορεί να κλιμακωθεί πιο εύκολα, αφού κάθε τμήμα μπορεί να βελτιώνεται ή να αναπτύσσεται ξεχωριστά χωρίς να επηρεάζει το άλλο.
4. **Cacheability**: Οι αποκρίσεις μπορούν να χαρακτηριστούν ως cacheable ή non-cacheable, επιτρέποντας στους πελάτες να αποθηκεύουν προσωρινά δεδομένα για βελτίωση της απόδοσης και μείωση φόρτου στον εξυπηρετητή.
5. **Layered System**: Η αρχιτεκτονική μπορεί να οργανωθεί σε πολλαπλά επίπεδα (π.χ. proxies, gateways, authentication servers). Ο πελάτης δεν χρειάζεται να γνωρίζει εάν αλληλεπιδρά απευθείας με τον τελικό εξυπηρετητή ή με ενδιάμεσα επίπεδα.

6. **Code-on-Demand (προαιρετικό)**: Ο εξυπηρετητής μπορεί να επεκτείνει τις δυνατότητες του πελάτη αποστέλλοντας εκτελέσιμο κώδικα (π.χ. JavaScript). Αν και σπάνια χρησιμοποιείται, δίνει ευελιξία σε πιο προηγμένα σενάρια.

3.3.2 Γιατί το REST έγινε δημοφιλές

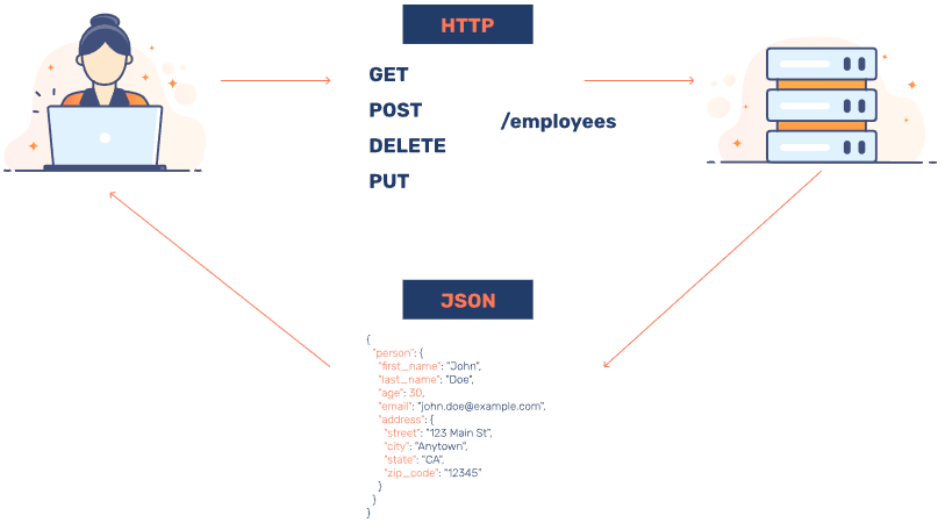
Η δημοτικότητα του REST οφείλεται κυρίως στην απλότητα και στη χρήση τεχνολογιών που ήδη είναι ευρέως διαδεδομένες. Το γεγονός ότι βασίζεται στο πρωτόκολλο HTTP, επιτρέπει εύκολη κατανόηση και υλοποίηση από προγραμματιστές. Η χρήση του JSON ως μορφή ανταλλαγής δεδομένων καθιστά τις αποκρίσεις ελαφριές και αναγνώσιμες από άνθρωπο και μηχανή.

Αυτά τα χαρακτηριστικά το κατέστησαν την κυρίαρχη επιλογή για τον σχεδιασμό κλιμακούμενων, ευέλικτων και αποδοτικών διαδικτυακών υπηρεσιών, σε σύγκριση με πιο περίπλοκα πρότυπα, όπως το SOAP [8]–[10].

3.4 Επιστροφή δεδομένων σε JSON/XML

Στα πρώτα Web Services, τα δεδομένα μεταφέρονταν συνήθως σε μορφή XML. Ωστόσο, με την άνοδο του REST, η χρήση του JSON έγινε κυρίαρχη λόγω της απλότητάς του, της αναγνωσιμότητας από ανθρώπους και της ευκολίας επεξεργασίας από γλώσσες προγραμματισμού. Το JSON είναι ελαφρύ, ενώ τα περισσότερα REST APIs το χρησιμοποιούν ως προεπιλεγμένη μορφή, χωρίς να αποκλείουν εναλλακτικές όπως XML, όπου αυτό κρίνεται απαραίτητο [8], [9].

3.5 Παράδειγμα ολοκληρωμένης αλληλεπίδρασης



Σχήμα 3.2: Απεικόνιση της σχέσης μεταξύ HTTP requests και αποκρίσεων JSON σε ένα REST API.

```
GET /employees HTTP/1.1
Host: example.com
Accept: application/json

HTTP/1.1 200 OK
Content-Type: application/json

[
  {
    "id": 1,
    "first_name": "John",
    "last_name": "Doe",
    "email": "john.doe@example.com"
  },
  {
    "id": 2,
    "first_name": "Jane",
    "last_name": "Smith",
    "email": "jane.smith@example.com"
  }
]
```

Σχήμα 3.3: Παράδειγμα διαχείρησης του συνόλου εργαζομένων μέσω του πόρου `/employees`

Στο παραπάνω παράδειγμα, ο πελάτης ζητά τη λίστα των εργαζομένων. Ο εξυπηρετητής απαντά με κωδικό κατάστασης `200 OK` και επιστρέφει μια αναπαράσταση της λίστας σε μορφή JSON. Η απλή αυτή διαδικασία καταδεικνύει πώς το REST αξιοποιεί την υπάρχουσα υποδομή του HTTP για να προσφέρει τυποποιημένη και κατανοητή επικοινωνία.

3.6 Μέθοδοι HTTP στα REST APIs

Οι μέθοδοι του HTTP, που παρουσιάστηκαν αναλυτικά στην Ενότητα 2.1, αποτελούν τον βασικό μηχανισμό αλληλεπίδρασης με τους πόρους ενός REST API. Η χρήση τους είναι στενά συνδεδεμένη με το μοντέλο CRUD (Create, Read, Update, Delete):

- **GET**: Ανάγνωση (Read) πόρων χωρίς μεταβολή της κατάστασής τους.
- **POST**: Δημιουργία (Create) νέου πόρου στον εξυπηρετητή.
- **PUT**: Πλήρης ενημέρωση (Update) υπάρχοντος πόρου.

- **PATCH**: Μερική ενημέρωση (Update) στοιχείων πόρου.
- **DELETE**: Διαγραφή (Delete) ενός πόρου.

Η τυποποιημένη αυτή αντιστοίχιση επιτρέπει στους προγραμματιστές να γνωρίζουν εκ των προτέρων τον τρόπο αλληλεπίδρασης με έναν πόρο, προάγοντας τη συνέπεια και τη διαλειτουργικότητα μεταξύ διαφορετικών APIs.

3.7 Παράμετροι στα REST APIs

Η επικοινωνία με ένα REST API πραγματοποιείται μέσω *endpoints*, δηλαδή συγκεκριμένων διαδρομών (URLs) που ορίζουν πού βρίσκεται ένας πόρος στον εξυπηρετητή. Για παράδειγμα, το `https://api.example.com/users` αποτελεί ένα endpoint που αναφέρεται στο σύνολο των χρηστών, ενώ το `https://api.example.com/users/5` δείχνει σε έναν συγκεκριμένο χρήστη με ταυτότητα `id = 5`.

Τα endpoints συνδυάζονται με τις μεθόδους HTTP (GET, POST, PUT, DELETE κ.ά.) για να δηλώσουν τη συγκεκριμένη ενέργεια που θα εκτελεστεί πάνω στον πόρο. Ωστόσο, πέρα από το endpoint και τη μέθοδο, κάθε αίτηση περιλαμβάνει και παραμέτρους που μεταφέρουν επιπλέον πληροφορίες. Οι σημαντικότεροι τύποι παραμέτρων είναι οι εξής.

3.7.1 Path Parameters

Οι path parameters ενσωματώνονται απευθείας στη διαδρομή (URL) και καθορίζουν ποιος ακριβώς πόρος ζητείται. Χρησιμοποιούνται κυρίως για την ανάγνωση, ενημέρωση ή διαγραφή συγκεκριμένων αντικειμένων.

```
GET /users/5 HTTP/1.1
Host: api.example.com
```

Σχήμα 3.4: Παράδειγμα path parameter

Η αίτηση αυτή επιστρέφει τον χρήστη με `id = 5`. Τα path parameters αποτελούν μέρος του endpoint και συνήθως δηλώνονται με μορφή `/resource/{id}`.

3.7.2 Query Parameters

Οι query parameters προστίθενται στο τέλος του URL μετά το σύμβολο `?`. Έχουν τη μορφή `key=value` και διαχωρίζονται μεταξύ τους με το σύμβολο `&`. Χρησιμοποιούνται κυρίως για αναζητήσεις, φιλτράρισμα, ταξινόμηση ή σελιδοποίηση.

```
GET /users?sort=asc&limit=10 HTTP/1.1
Host: api.example.com
```

Σχήμα 3.5: Παράδειγμα query parameter

Η παραπάνω αίτηση επιστρέφει τους χρήστες ταξινομημένους σε αύξουσα σειρά και περιορισμένους στους δέκα πρώτους. Τα query parameters είναι ιδιαίτερα χρήσιμα όταν χρειαζόμαστε δυναμικά φίλτρα χωρίς να αλλάζουμε το βασικό endpoint.

Διάκριση Path και Query Parameters: Παρόλο που και οι δύο τύποι παραμέτρων χρησιμοποιούνται για τον καθορισμό της ζητούμενης πληροφορίας, εξυπηρετούν διαφορετικούς σκοπούς. Η διάκριση συνοψίζεται στον Πίνακα 3.1.

Χαρακτηριστικό	**Path Parameters**	**Query Parameters**
Σκοπός	Προσδιορίζουν έναν μοναδικό και συγκεκριμένο πόρο	Φιλτράρουν, ταξινομούν ή περιορίζουν συλλογές πόρων
Παράδειγμα	`/users/5` → ο χρήστης με `id = 5`	`/users?sort=asc&limit=10` → λίστα χρηστών με κριτήρια
Χρήση	Συνήθως με GET, PUT, PATCH, DELETE	Κυρίως με GET, για αναζήτηση ή σελιδοποίηση
Υποχρεωτικότητα	Απαραίτητα για την ταυτοποίηση πόρου — αν λείπουν, το endpoint είναι άκυρο	Προαιρετικά — αν λείπουν, το API συχνά χρησιμοποιεί προεπιλεγμένες τιμές

Πίνακας 3.1: Σύγκριση Path και Query Parameters.

3.7.3 Headers

Τα headers περιέχουν μεταδεδομένα που συνοδεύουν κάθε αίτηση ή απόκριση. Μεταφέρουν κρίσιμες πληροφορίες όπως το είδος του περιεχομένου, τον τρόπο κωδικοποίησης, ή διαπιστευτήρια ασφαλείας.

```
GET /users HTTP/1.1
Host: api.example.com
Authorization: Bearer <token>
Accept: application/json
```

Σχήμα 3.6: Παράδειγμα header

Στο παράδειγμα, το `Authorization` header παρέχει το διακριτικό πρόσβασης (token) που πιστοποιεί τον χρήστη, ενώ το `Accept` δηλώνει ότι η απόκριση πρέπει να είναι σε μορφή JSON.

3.7.4 Request Body

Το σώμα της αίτησης (request body) χρησιμοποιείται όταν πρέπει να μεταφέρουμε πλήρη δεδομένα προς τον εξυπηρετητή. Είναι απαραίτητο στις μεθόδους POST, PUT και PATCH, καθώς επιτρέπει τη δημιουργία ή ενημέρωση αντικειμένων.

```
POST /users HTTP/1.1
Host: api.example.com
Content-Type: application/json

{
  "first_name": "Alice",
  "last_name": "Johnson",
  "email": "alice.johnson@example.com"
}
```

Σχήμα 3.7: Παράδειγμα request body

Στο παράδειγμα, η αίτηση αυτή δημιουργεί έναν νέο χρήστη με τα στοιχεία που παρέχονται στο σώμα. Το `Content-Type` header ενημερώνει τον εξυπηρετητή ότι το περιεχόμενο είναι σε μορφή JSON.

Η κατανόηση της σχέσης μεταξύ *endpoints*, μεθόδων και παραμέτρων είναι θεμελιώδης για τον σωστό σχεδιασμό και τη χρήση REST APIs. Μέσα από αυτήν τη δομή, οι προγραμματιστές μπορούν να αλληλεπιδρούν με σαφήνεια με τους πόρους, εξασφαλίζοντας συνέπεια και επεκτασιμότητα στα συστήματα.

3.7.5 Stateful vs Stateless

Στην αρχιτεκτονική των διαδικτυακών εφαρμογών γίνεται συχνά διάκριση ανάμεσα σε δύο τρόπους διαχείρισης της επικοινωνίας πελάτη–εξυπηρετητή: *stateful* και *stateless*.

Στα stateful συστήματα, ο εξυπηρετητής διατηρεί πληροφορίες για την κατάσταση (state) του πελάτη ανάμεσα σε διαδοχικές αιτήσεις. Αυτό σημαίνει ότι κάθε αίτημα δεν είναι πλήρως αυτόνομο, αλλά εξαρτάται από προηγούμενες αλληλεπιδράσεις. Χαρακτηριστικό παράδειγμα αποτελούν οι εφαρμογές που βασίζονται σε sessions: μόλις ένας χρήστης συνδεθεί, ο εξυπηρετητής αποθηκεύει την κατάστασή του και την αξιοποιεί σε μελλοντικά αιτήματα.

Αντίθετα, στα stateless συστήματα ο εξυπηρετητής δεν κρατά καμία πληροφορία για τον πελάτη. Κάθε αίτημα περιλαμβάνει όλα τα απαραίτητα στοιχεία (διαπιστευτήρια, δεδομένα, παραμέτρους), ώστε να μπορεί να επεξεργαστεί μεμονωμένα, χωρίς γνώση του τι έχει προηγηθεί. Με τον τρόπο αυτό, οι αιτήσεις είναι πλήρως ανεξάρτητες μεταξύ τους.

Τα REST APIs είναι σχεδιασμένα να είναι *stateless*. Η επιλογή αυτή προσφέρει σημαντικά πλεονεκτήματα. Απλοποιεί την υλοποίηση, καθώς ο εξυπηρετητής δεν χρειάζεται να διατηρεί πληροφορίες συνεδρίας για κάθε χρήστη. Επιπλέον, συμβάλλει στη βελτίωση της απόδοσης, αφού κάθε αίτημα μπορεί να επεξεργαστεί ανεξάρτητα, χωρίς να απαιτείται γνώση του ιστορικού προηγούμενων αιτήσεων. Τέλος, η αρχή του *stateless* καθιστά ευκολότερη την κλιμάκωση (scalability), επιτρέποντας σε πολλούς εξυπηρετητές να εξυπηρετούν ταυτόχρονα αιτήματα χωρίς την ανάγκη κοινόχρηστης μνήμης κατάστασης.

Ωστόσο, στην πράξη υπάρχουν σενάρια όπου είναι απαραίτητη η διατήρηση κατάστασης, όπως στην ταυτοποίηση και την εξουσιοδότηση χρηστών. Για να καλυφθεί αυτή η ανάγκη χωρίς να παραβιαστεί η *stateless* αρχή, χρησιμοποιούνται μηχανισμοί όπως τα cookies και τα sessions. Τα cookies αποθηκεύουν μικρά τμήματα πληροφορίας στον φυλλομετρητή και τα αποστέλλουν σε κάθε αίτημα, ενώ τα sessions επιτρέπουν στον εξυπηρετητή να συνδέει αυτά τα cookies με δεδομένα που κρατά προσωρινά στη μνήμη του.

Η λειτουργία των μηχανισμών αυτών αναλύθηκε ήδη στην Ενότητα 2.6, όπου παρουσιάστηκε και η τυπική ροή διαχείρισης συνεδρίας με cookies. Έτσι, τα REST APIs παραμένουν *stateless* σε επίπεδο πρωτοκόλλου, ενώ η αναγκαία πληροφορία για τον χρήστη «μεταφέρεται» μαζί με κάθε αίτημα.

Κεφάλαιο 4

Παρουσίαση Εφαρμογής και Σχεδίαση

4.1 Απαιτήσεις

Η ανάλυση και η καταγραφή των απαιτήσεων (*requirements*) αποτελεί την πρώτη και ίσως σημαντικότερη διαδικασία στην ανάπτυξη λογισμικού. Η φάση αυτή αντιστοιχεί στην *προδιαγραφή* του συστήματος και έχει ως σκοπό τον σαφή καθορισμό των λειτουργιών που θα επιτελεί το λογισμικό, καθώς και των περιορισμών και παραδοχών που το διέπουν. Ανεξάρτητα από το μοντέλο κύκλου ζωής που ακολουθείται , η προδιαγραφή αποτελεί το πρώτο βήμα και θέτει τα θεμέλια για την επιτυχία ή, αντιστρόφως, την αποτυχία του έργου.

Μια **απαίτηση από το λογισμικό** ορίζεται ως μια λειτουργία που αυτό πρέπει να επιτελεί ή ως μια συνθήκη που οφείλει να ικανοποιεί όταν ολοκληρωθεί η κατασκευή του. Οι απαιτήσεις λειτουργούν ως «συμβόλαιο» μεταξύ του πελάτη και της ομάδας ανάπτυξης: ο πελάτης εκφράζει τις ανάγκες του σε υψηλό επίπεδο γενικότητας, ενώ ο κατασκευαστής τις μετατρέπει σε σαφείς και λεπτομερείς προδιαγραφές που μπορούν να υλοποιηθούν.

Η σαφής περιγραφή των απαιτήσεων είναι κρίσιμη για την επιτυχία ενός έργου λογισμικού. Λειτουργεί ως σημείο αναφοράς για όλες τις επόμενες φάσεις του κύκλου ζωής, όπως είναι η σχεδίαση, η υλοποίηση και ο έλεγχος, παρέχοντας μια συνεκτική βάση πάνω στην οποία οικοδομείται η ανάπτυξη. Συμβάλλει στη μείωση του κινδύνου παρερμηνειών μεταξύ των εμπλεκόμενων φορέων, καθώς εξασφαλίζει ότι όλοι μοιράζονται μια κοινή κατανόηση των στόχων του συστήματος. Επιπλέον, διευκολύνει την αξιολόγηση του τελικού προϊόντος με κριτήριο την πιστότητά του στις αρχικές προδιαγραφές, γεγονός που επιτρέπει την αντικειμενική εκτίμηση της ποιότητας του αποτελέσματος. Τέλος, σε μελλοντικό επίπεδο, η σωστή τεκμηρίωση των απαιτήσεων διασφαλίζει ότι το λογισμικό θα μπορεί να συντηρηθεί και να επεκταθεί στο μέλλον,καθώς προσφέρει σαφείς κατευθύνσεις για πιθανές μελλοντικές βελτιώσεις.

Οι απαιτήσεις διακρίνονται σε δύο βασικές κατηγορίες:

Οι **λειτουργικές απαιτήσεις** (***Functional Requirements***) περιγράφουν τι πρέπει να κάνει το σύστημα, δηλαδή τις συγκεκριμένες εργασίες και λειτουργίες που καλείται να εκτελεί. Ουσιαστικά καθορίζουν τη συμπεριφορά του λογισμικού και τις αποκρίσεις του σε δεδομένες συνθήκες, παρέχοντας ένα σαφές πλαίσιο για την ανάπτυξη και αξιολόγησή του.

Αντίθετα, οι **μη λειτουργικές απαιτήσεις** (***Non-Functional Requirements***) δεν εστιάζουν σε συγκεκριμένες λειτουργίες, αλλά σε ποιοτικά χαρακτηριστικά που αφορούν τον τρόπο λειτουργίας του συστήματος. Στην κατηγορία αυτή εντάσσονται στοιχεία όπως η χρηστικότητα, η αξιοπιστία και η συμβατότητα. Οι απαιτήσεις αυτές δεν περιγράφουν τι κάνει το λογισμικό, αλλά καθορίζουν ιδιότητες που χαρακτηρίζουν τη συνολική του ποιότητα και επηρεάζουν άμεσα την εμπειρία του χρήστη και την αποδοτικότητα του συστήματος.[11]

Στο πλαίσιο της παρούσας εργασίας, οι απαιτήσεις της εφαρμογής λογοθεραπείας που αναπτύχθηκε διαμορφώθηκαν βάσει των στόχων και της ανάλυσης του προβλήματος που έθεσαν οι φοιτητές λογοθεραπείας και παρουσιάζεται στα επόμενα κεφάλαια. Ακολουθεί η παρουσίαση των λειτουργικών απαιτήσεων του συστήματος.

4.1.1 Ανάλυση Λειτουργικών Απαιτήσεων

Η βασική λειτουργικότητα της εφαρμογής, και ο κύριος λόγος ύπαρξής της, είναι η δυνατότητα **παρουσίασης, δημιουργίας, τροποποίησης και διαγραφής ασκήσεων λογοθεραπείας**. Οι ασκήσεις οργανώνονται σε *bundles*, τα οποία αποτελούνται από διαδοχικά βήματα. Κάθε βήμα διαθέτει τίτλο, οδηγία και οπτικοακουστικό υλικό (εικόνα, ηχογράφηση ή βίντεο), ώστε να καθίσταται σαφές στον ασθενή με ποιον τρόπο πρέπει να εκτελεστεί η άσκηση. Η χρήση οπτικοακουστικού υλικού αποτελεί κρίσιμο στοιχείο, καθώς διευκολύνει την κατανόηση και την ορθή εκτέλεση της θεραπευτικής διαδικασίας.

Η εφαρμογή υποστηρίζει τη δημιουργία λογαριασμών και τη σύνδεση χρηστών με δύο βασικούς ρόλους: *κλινικός* και *ασθενής*. Οι λογαριασμοί διαχειριστών (*admins*) δημιουργούνται αποκλειστικά με χειροκίνητη διαδικασία από την ομάδα υποστήριξης, για λόγους ασφάλειας.

Οι κλινικοί έχουν τη δυνατότητα να διαχειρίζονται πλήρως το θεραπευτικό έργο με τους ασθενείς τους. Συγκεκριμένα:

- έχουν πρόσβαση στα προσωπικά στοιχεία των ασθενών τους και μπορούν να καταγράφουν ή να ενημερώνουν προσωπικές σημειώσεις,
- συνδέονται με συγκεκριμένους ασθενείς και διαχειρίζονται τις ασκήσεις που τους έχουν αναθέσει,
- δημιουργούν νέες ασκήσεις, εφόσον οι υπάρχουσες δεν τους ικανοποιούν, προσθέτοντας βήματα και οδηγίες με εικόνες, ηχογραφήσεις ή βίντεο,
- διατηρούν τον έλεγχο των ασκήσεων που δημιουργούν
- έχουν τη δυνατότητα να τροποποιούν τα προσωπικά τους στοιχεία ή να διαγράφουν τον λογαριασμό τους.

Οι ασθενείς έχουν πρόσβαση αποκλειστικά στις ασκήσεις που τους έχουν ανατεθεί από τον κλινικό τους. Οι ασκήσεις παρουσιάζονται βήμα προς βήμα με οπτικοακουστικό τρόπο, ώστε να είναι σαφείς και εύκολα εκτελέσιμες. Επιπλέον, ο ασθενής μπορεί να τροποποιεί τα προσωπικά του στοιχεία.

Οι διαχειριστές διαθέτουν εκτεταμένα δικαιώματα εποπτείας και ελέγχου του συστήματος. Πιο αναλυτικά:

- διαχειρίζονται τους λογαριασμούς χρηστών, εκτελώντας ενέργειες όπως επαναφορά κωδικών πρόσβασης ή διαγραφή κακόβουλων λογαριασμών,
- δημιουργούν και τροποποιούν ασκήσεις που είναι διαθέσιμες για όλους τους κλινικούς,
- εποπτεύουν τις ασκήσεις που έχουν δημιουργήσει οι κλινικοί,
- έχουν τη δυνατότητα να τροποποιούν τα δικά τους στοιχεία.

Με τον τρόπο αυτό, οι λειτουργικές απαιτήσεις της εφαρμογής καλύπτουν τις ανάγκες όλων των εμπλεκόμενων ρόλων, προσφέροντας στον κλινικό εργαλεία διαχείρισης και εξατομίκευσης, στον ασθενή σαφή και κατανοητή παρουσίαση των ασκήσεων και στον διαχειριστή πλήρη έλεγχο και εποπτεία του συστήματος.

4.1.2 Ανάλυση Μη Λειτουργικών Απαιτήσεων

Πρώτιστη σημασία έχει η **ασφάλεια**. Οι κωδικοί πρόσβασης αποθηκεύονται με κρυπτογραφημένη μορφή στη βάση δεδομένων, ενώ το σύστημα ενσωματώνει μηχανισμούς ελέγχου πρόσβασης, ώστε κάθε χρήστης να διαθέτει δικαιώματα που αντιστοιχούν στον ρόλο του. Η προστασία των δεδομένων είναι κρίσιμη, καθώς αφορούν τα προσωπικά δεδομένα των ασθενών.

Σημαντική απαίτηση αποτελεί επίσης η **φορητότητα και η ευκολία ανάπτυξης**. Η εφαρμογή υλοποιείται με χρήση Docker, με κάθε συνιστώσα (βάση δεδομένων, backend, frontend) να εκτελείται σε ξεχωριστό container. Με αυτόν τον τρόπο διασφαλίζεται η ομοιόμορφη λειτουργία σε διαφορετικά περιβάλλοντα και, κυρίως, απλοποιείται για τους developers η διαδικασία εγκατάστασης, συντήρησης και μελλοντικών ενημερώσεων. Η απαίτηση αυτή στοχεύει στη διευκόλυνση της ομάδας ανάπτυξης και όχι σε λειτουργικό χαρακτηριστικό που αφορά άμεσα τους τελικούς χρήστες.

Η **χρηστικότητα** είναι καθοριστική για την αποδοχή του συστήματος. Το περιβάλλον χρήστη έχει σχεδιαστεί ώστε να είναι απλό και κατανοητό, επιτρέποντας σε λογοθεραπευτές και ασθενείς να πλοηγούνται με ευκολία και να αλληλεπιδρούν με το πολυμεσικό υλικό των ασκήσεων. Έμφαση δίνεται στη σαφή παρουσίαση του περιεχομένου και στην ευχρηστία της πρόσβασης μέσω διαδικτύου.

Παράλληλα, η εφαρμογή πρέπει να εξασφαλίζει **συμβατότητα** με διαφορετικά περιβάλλοντα χρήσης. Η υποστήριξη σύγχρονων φυλλομετρητών (Chrome, Firefox, Safari, Edge) και η προσαρμογή σε πολλαπλές συσκευές (υπολογιστές, tablets, φορητά συστήματα) καθιστούν την πλατφόρμα ευέλικτη και προσβάσιμη σε διαφορετικές κατηγορίες χρηστών.

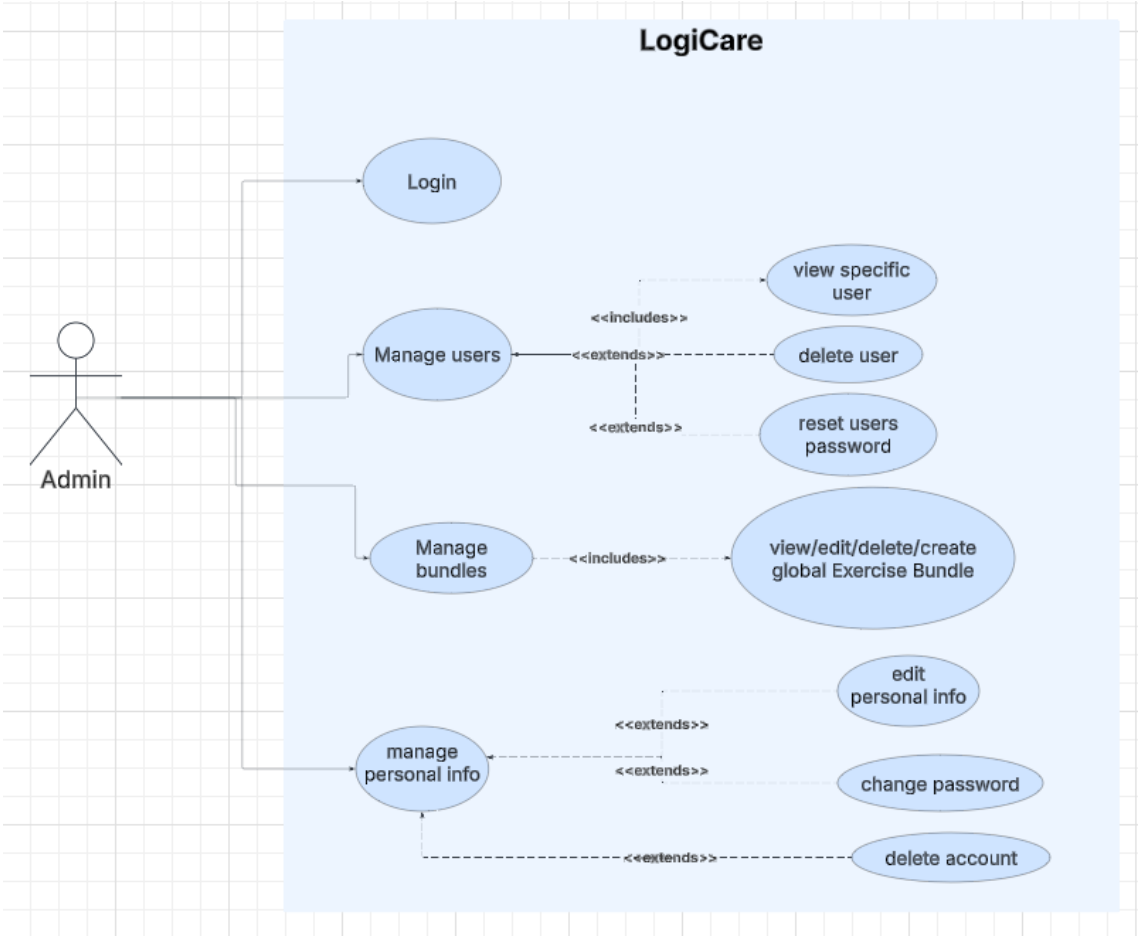
Τέλος, ιδιαίτερη σημασία έχει η **συντηρησιμότητα** του λογισμικού. Ο κώδικας έχει οργανωθεί σε επιμέρους modules, ενώ η χρήση σύγχρονων frameworks όπως η React και το Express επιτρέπει την ευκολότερη κατανόηση, την επαναχρησιμοποίηση και την επέκταση της λειτουργικότητας σε μελλοντικές εκδόσεις.

4.2 Use Case Diagrams

Στο πλαίσιο της σχεδίασης συστημάτων λογισμικού, τα Use Case Diagrams αποτελούν βασικό εργαλείο της γλώσσας UML (Unified Modeling Language). Μέσω αυτών απεικονίζονται οι λειτουργίες που παρέχει το σύστημα (*use cases*) και οι κατηγορίες χρηστών που αλληλεπιδρούν με αυτό (*actors*). Στόχος των διαγραμμάτων είναι να δοθεί μια καθαρή και συνοπτική εικόνα των δυνατοτήτων της εφαρμογής, καθώς και των σχέσεων ανάμεσα στους χρήστες και τις επιμέρους λειτουργίες. Με αυτόν τον τρόπο διευκολύνεται η επικοινωνία μεταξύ αναλυτών, προγραμματιστών και τελικών χρηστών, καθώς όλοι μοιράζονται ένα κοινό οπτικό σημείο αναφοράς.[11]

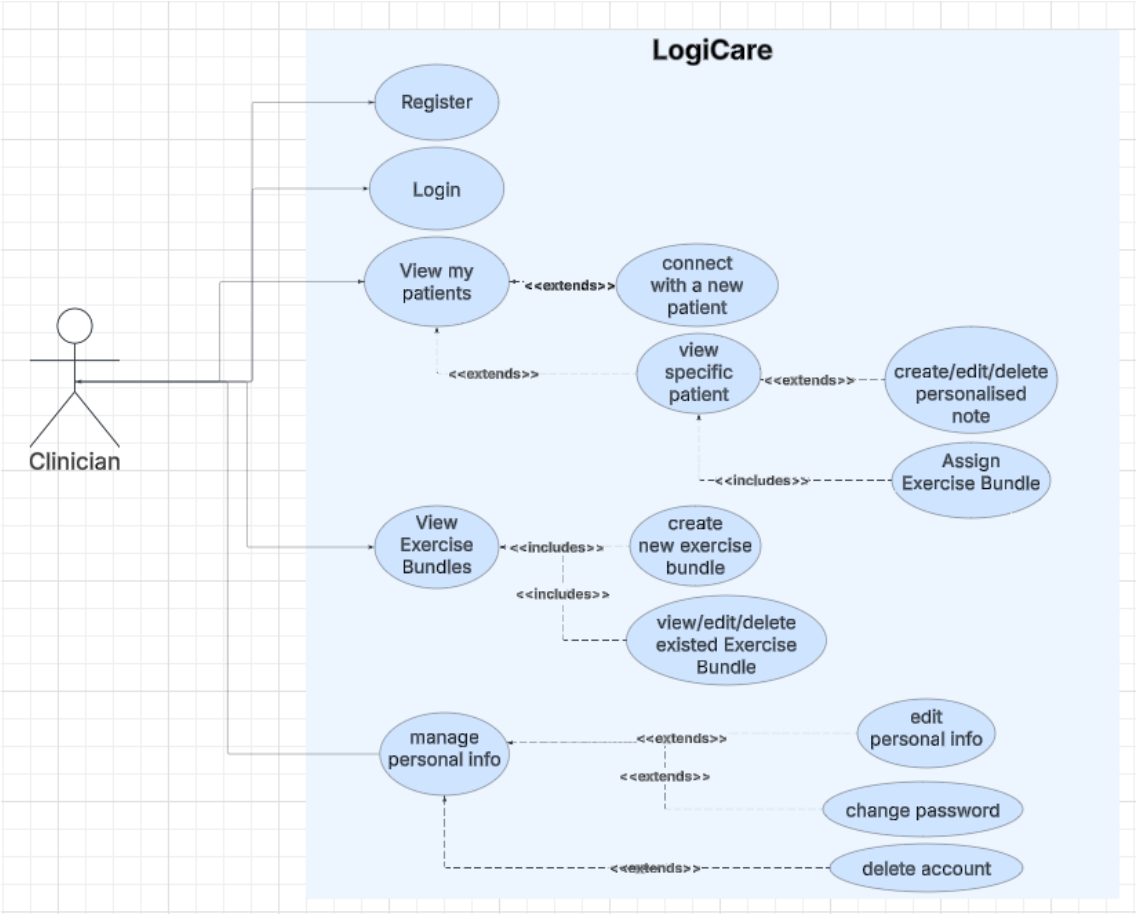
Στην εφαρμογή λογοθεραπείας που αναπτύχθηκε, οι βασικοί ρόλοι είναι τρεις: *Διαχειριστής*, *Κλινικός* και *Ασθενής*. Για κάθε ρόλο δημιουργήθηκε ξεχωριστό Use Case Diagram, το οποίο αποτυπώνει τις ενέργειες που μπορεί να εκτελέσει και τα δικαιώματα που διαθέτει μέσα στο σύστημα. Με τον τρόπο αυτό καθίσταται σαφής η διάκριση των αρμοδιοτήτων και των αλληλεπιδράσεων που υποστηρίζει η πλατφόρμα.

Στο Σχήμα 4.1 παρουσιάζεται το διάγραμμα use case για τον **Διαχειριστή**. Ο ρόλος αυτός έχει εκτεταμένες δυνατότητες εποπτείας, όπως η διαχείριση χρηστών και η δημιουργία ή τροποποίηση ασκήσεων που είναι διαθέσιμες σε όλους τους κλινικούς.



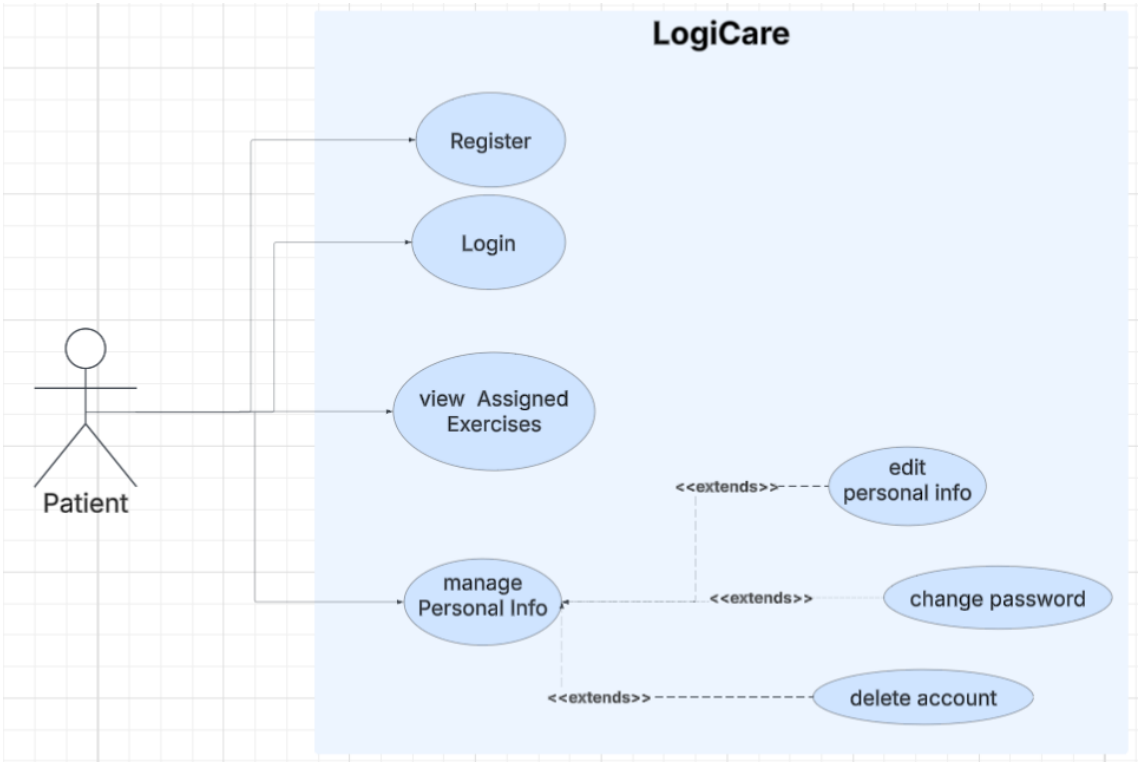
Σχήμα 4.1: Use Case Diagram για τον Διαχειριστή

Στο Σχήμα 4.2 απεικονίζεται το διάγραμμα για τον **Κλινικό**. Ο κλινικός έχει τη δυνατότητα να διαχειρίζεται τους ασθενείς του, να δημιουργεί και να αναθέτει εξατομικευμένες ασκήσεις, καθώς και να καταγράφει προσωπικές σημειώσεις για την πορεία της θεραπείας.



Σχήμα 4.2: Use Case Diagram για τον Κλινικό

Τέλος, στο Σχήμα 4.3 παρουσιάζεται το διάγραμμα για τον **Ασθενή**. Ο ασθενής έχει πρόσβαση στις ασκήσεις που του έχουν ανατεθεί από τον κλινικό του και μπορεί να διαχειρίζεται τα προσωπικά του στοιχεία, όπως την επεξεργασία πληροφοριών ή την αλλαγή κωδικού πρόσβασης.



Σχήμα 4.3: Use Case Diagram για τον Ασθενή

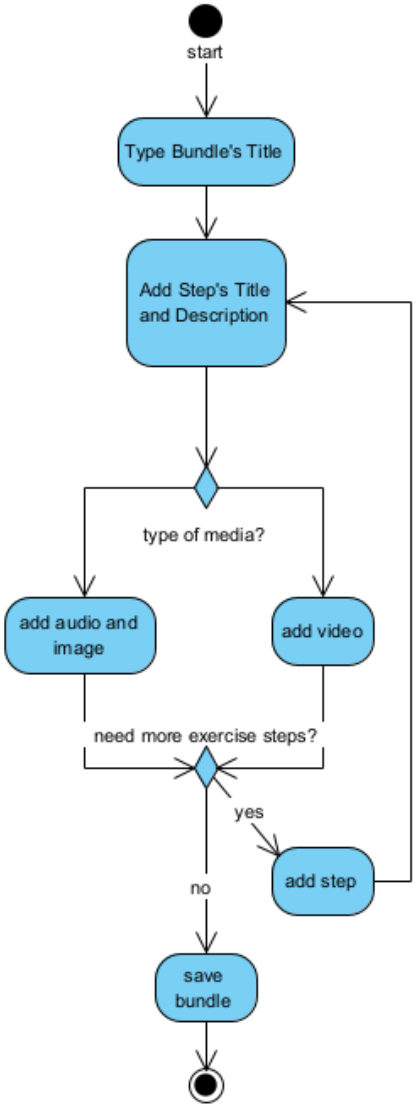
4.3 Activity Diagrams

Ύστερα από την ανάλυση των απαιτήσεων του συστήματος και την αξιοποίησή τους για τη διαμόρφωση των Use Case Diagrams, προχωρούμε σε ένα επόμενο επίπεδο λεπτομέρειας με τη χρήση Activity Diagrams. Ενώ τα use cases αποτυπώνουν ποιοι ρόλοι (actors) αλληλεπιδρούν με το σύστημα και ποιες λειτουργίες εκτελούν, τα activity diagrams επικεντρώνονται στον τρόπο με τον οποίο αυτές οι λειτουργίες υλοποιούνται βήμα προς βήμα. Με άλλα λόγια, περιγράφουν τη ροή δραστηριοτήτων που πραγματοποιούνται στο εσωτερικό κάθε σεναρίου χρήσης.

Η ένταξη των activity diagrams στη μελέτη μάς επιτρέπει να αναδείξουμε με μεγαλύτερη σαφήνεια την εσωτερική λογική της εφαρμογής, να εντοπίσουμε κρίσιμα σημεία απόφασης και να κατανοήσουμε πώς οι χρήστες πλοηγούνται μέσα στο σύστημα.[11] Επιπλέον, τα διαγράμματα αυτά λειτουργούν ως γέφυρα με το επόμενο στάδιο σχεδίασης, το οποίο αφορά τα wireframes, δηλαδή την απεικόνιση των βασικών οθονών της εφαρμογής. Με τον τρόπο αυτό, κάθε διάγραμμα δραστηριοτήτων μπορεί να συνοδεύεται από τα αντίστοιχα wireframes, ώστε να αποτυπώνεται όχι μόνο η λογική ροή αλλά και η οπτική διάσταση της αλληλεπίδρασης.

4.3.1 Διαγράμματα Βασικών Λειτουργιών

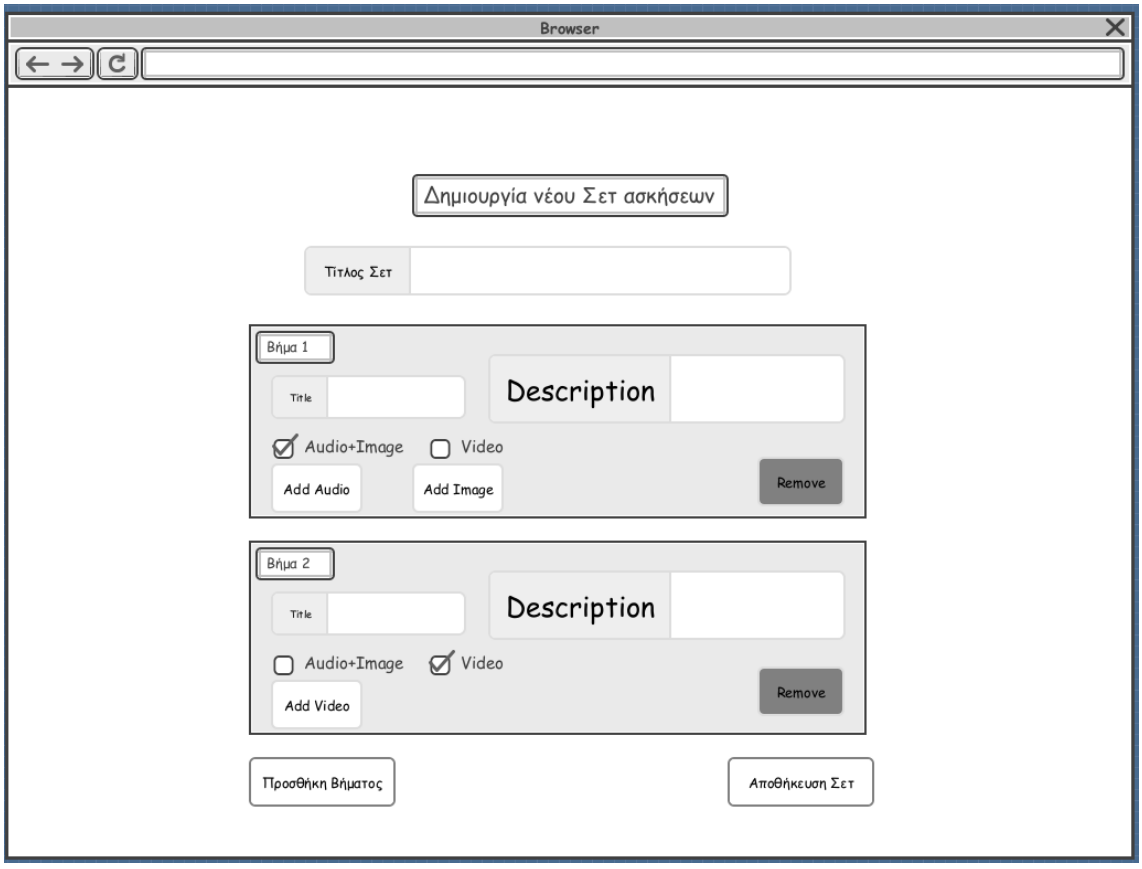
Η κεντρική λειτουργικότητα της εφαρμογής σχετίζεται με τη δημιουργία και την εκτέλεση ασκήσεων λογοθεραπείας. Για τον λόγο αυτό παρουσιάζονται δύο activity diagrams που απεικονίζουν τα βήματα αυτών των διαδικασιών.



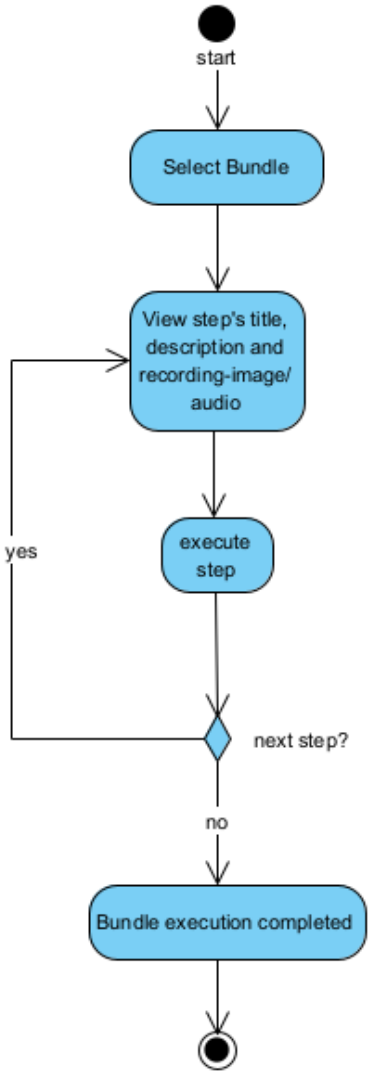
Σχήμα 4.4: Activity Diagram: Δημιουργία Άσκησης (Bundle)

Στο Σχήμα 4.4 παρουσιάζεται η διαδικασία δημιουργίας ενός σετ ασκήσεων (*bundle*). Το διάγραμμα δείχνει αναλυτικά τα βήματα που ακολουθεί ο κλινικός: εισαγωγή τίτλου, προσθήκη βημάτων με περιγραφή και πολυμεσικό υλικό, καθώς και τελική αποθήκευση.

Η ίδια διαδικασία αποτυπώνεται και οπτικά στο αντίστοιχο wireframe του Σχήματος 4.5. Η διεπαφή παρέχει πεδία για τον τίτλο του σετ, τη δημιουργία επιμέρους βημάτων, την προσθήκη πολυμεσικού υλικού (εικόνα, ήχος ή βίντεο) και κουμπιά για την αφαίρεση ή την προσθήκη νέων βημάτων. Τέλος, το κουμπί «Αποθήκευση Σετ» ολοκληρώνει τη διαδικασία, σε πλήρη αντιστοιχία με τα βήματα που περιγράφονται στο activity diagram.



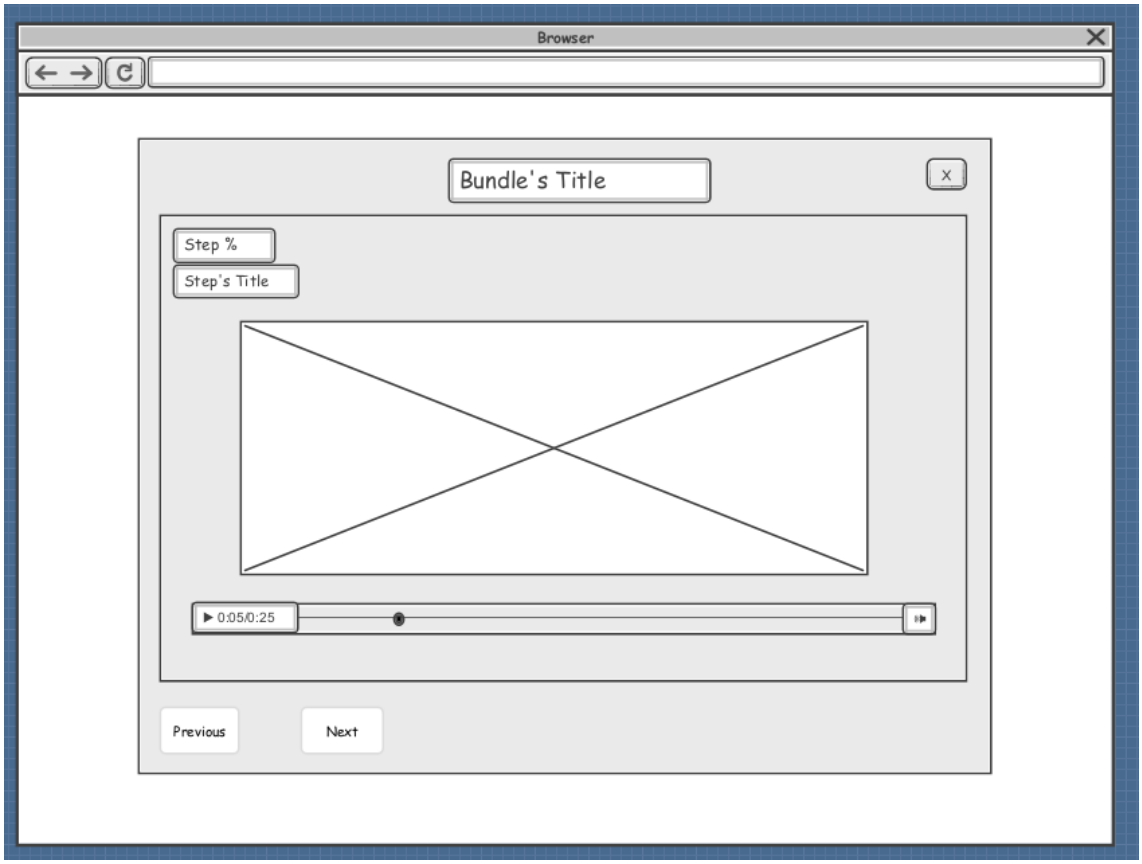
Σχήμα 4.5: Wireframe: Δημιουργία Άσκησης (Bundle)



Σχήμα 4.6: Activity Diagram: Εκτέλεση Άσκησης (Bundle)

Στο Σχήμα 4.6 παρουσιάζεται η διαδικασία εκτέλεσης ενός σετ ασκήσεων. Ο ασθενής επιλέγει το bundle που του έχει ανατεθεί και ακολουθεί βήμα προς βήμα τις οδηγίες που περιλαμβάνουν τίτλο, περιγραφή και πολυμεσικό υλικό (εικόνα, ηχογράφηση ή βίντεο). Για κάθε βήμα έχει τη δυνατότητα να αλληλεπιδράσει με το υλικό και, όταν ολοκληρώσει, να μεταβεί στο επόμενο μέχρι την ολοκλήρωση όλου του σετ.

Η λειτουργικότητα αυτή αποτυπώνεται στο αντίστοιχο wireframe του Σχήματος 4.7. Η διεπαφή περιλαμβάνει τον τίτλο του bundle, την ένδειξη προόδου και το πολυμεσικό περιεχόμενο του κάθε βήματος, συνοδευόμενο από κουμπιά πλοήγησης (*Previous, Next*) για τη μετάβαση στα διαδοχικά βήματα. Με τον τρόπο αυτό, η ροή που απεικονίζεται στο activity diagram βρίσκει την πρακτική της υλοποίηση στην οθόνη χρήσης του ασθενούς.

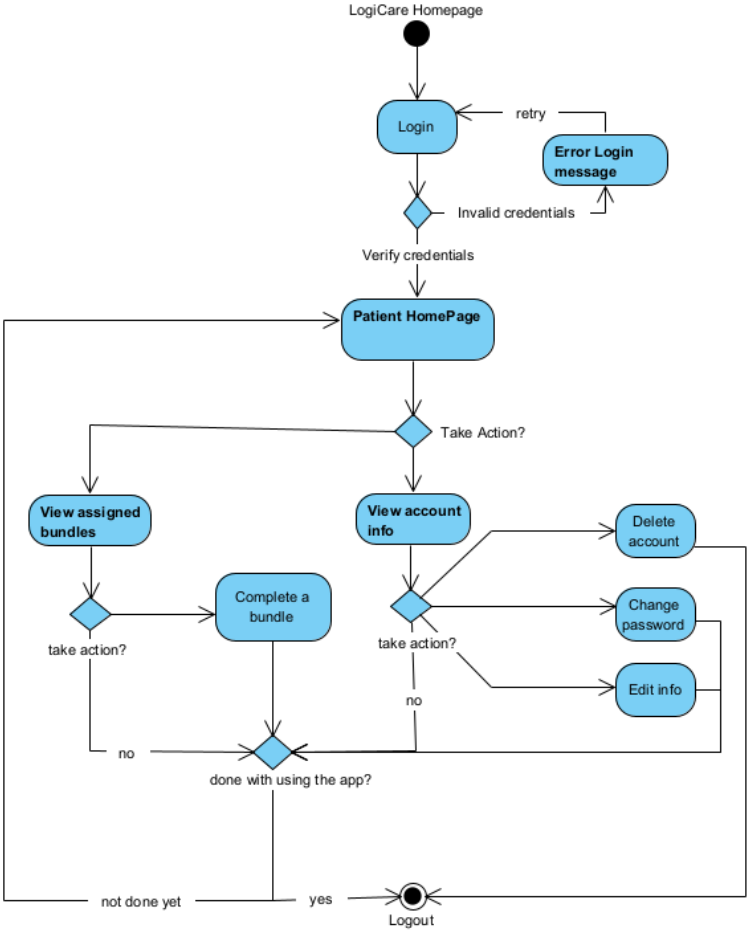


Σχήμα 4.7: Wireframe: Εκτέλεση Άσκησης (Bundle)

4.3.2 Διαγράμματα ανά Ρόλο

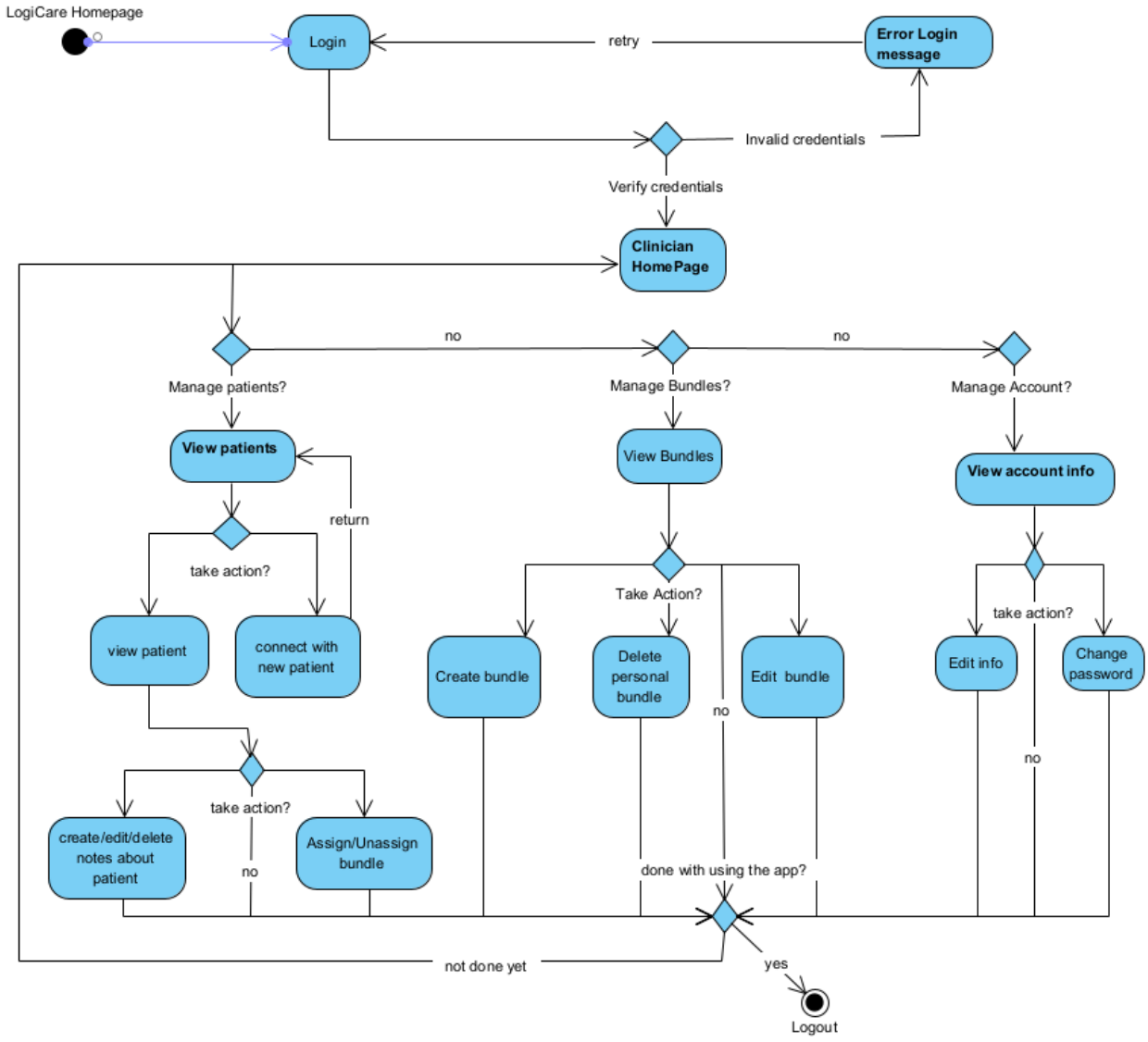
Εκτός από τις βασικές λειτουργίες, κρίθηκε απαραίτητο να αποτυπωθούν και οι συνολικές δραστηριότητες που μπορεί να εκτελέσει κάθε κατηγορία χρήστη. Με τον τρόπο αυτό παρουσιάζεται μια ολοκληρωμένη εικόνα της εμπειρίας χρήσης για κάθε ρόλο, καθώς και οι διαφορετικές ευθύνες και δικαιώματα που διαθέτει.

Στο Σχήμα 4.8 αποτυπώνεται η ροή δραστηριοτήτων ενός ασθενούς. Ο χρήστης συνδέεται στην πλατφόρμα, έχει τη δυνατότητα να προβάλει τα σετ ασκήσεων που του έχουν ανατεθεί και να τα εκτελέσει βήμα προς βήμα. Παράλληλα, μπορεί να διαχειριστεί τα προσωπικά του στοιχεία (τροποποίηση, αλλαγή κωδικού, διαγραφή λογαριασμού) και να αποσυνδεθεί όταν ολοκληρώσει τη χρήση της εφαρμογής.



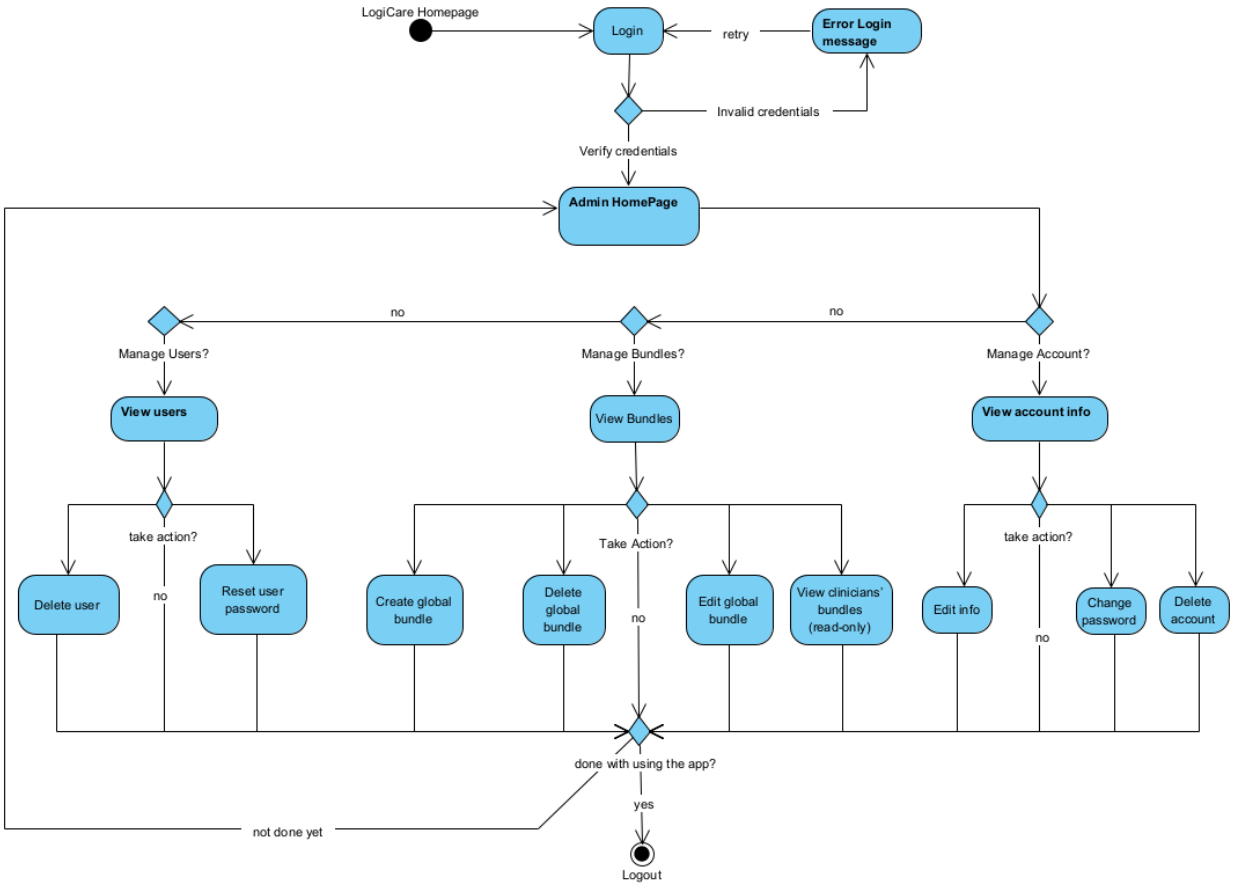
Σχήμα 4.8: Activity Diagram: Ασθενής

Στο Σχήμα 4.9 παρουσιάζεται η δραστηριότητα του κλινικού. Ο κλινικός μπορεί να συνδέεται, να διαχειρίζεται τους ασθενείς του (προβολή στοιχείων, καταγραφή σημειώσεων, ανάθεση/αφαίρεση bundles), καθώς και να δημιουργεί, επεξεργάζεται ή διαγράφει δικά του σετ ασκήσεων. Επιπλέον, έχει πρόσβαση στη διαχείριση του λογαριασμού του και τη δυνατότητα αποσύνδεσης από την εφαρμογή.



Σχήμα 4.9: Activity Diagram: Κλινικός

Στο Σχήμα 4.10 αποτυπώνεται η ροή δραστηριοτήτων του διαχειριστή. Ο διαχειριστής έχει ευρύτερες αρμοδιότητες, όπως η προβολή και διαχείριση όλων των χρηστών (διαγραφή, επαναφορά κωδικών), η δημιουργία και εποπτεία «global» bundles που είναι διαθέσιμα σε όλους τους κλινικούς, καθώς και η ανάγνωση bundles που έχουν δημιουργήσει οι ίδιοι οι κλινικοί. Επιπλέον, διαθέτει πρόσβαση στις βασικές λειτουργίες διαχείρισης λογαριασμού και αποσύνδεσης.



Σχήμα 4.10: Activity Diagram: Διαχειριστής

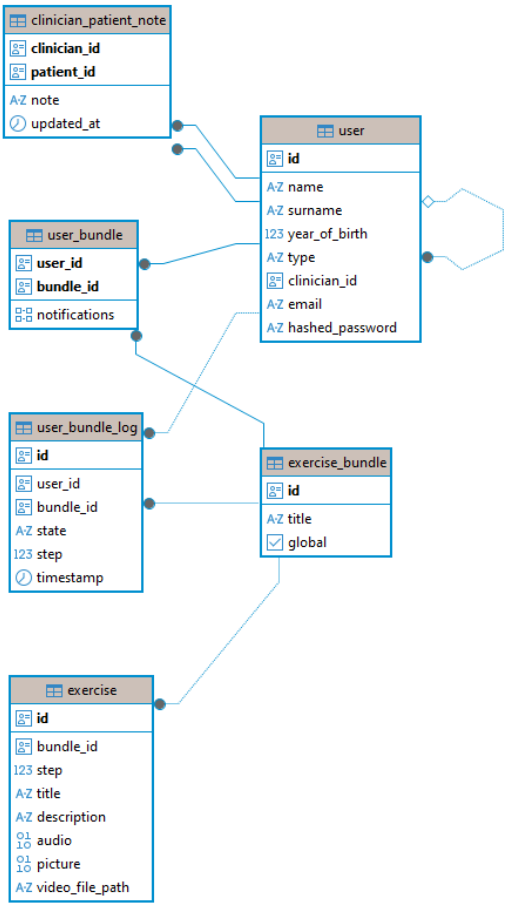
4.4 Επιλογή Τεχνολογιών και Component Diagrams

4.4.1 Βάση Δεδομένων

Επιλογή Τεχνολογιών: PostgreSQL.

Η PostgreSQL είναι ένα ώριμο, ανοιχτού κώδικα, αντικειμενοστρεφές ΣΔΒΔ (RDBMS) με πλήρη υποστήριξη ACID συναλλαγών και συμμόρφωση στα πρότυπα SQL. Προσφέρει πλούσια χαρακτηριστικά όπως ισχυρούς περιορισμούς ακεραιότητας (foreign keys, check, unique), προηγμένα ευρετήρια (B-Tree, GIN/GiST), views και επεκτάσεις. Είναι ευρέως διαδεδομένη, καλοτεκμηριωμένη και υποστηρίζεται από εκτενές οικοσύστημα εργαλείων (psql, pgAdmin, κ.ά.).[12]

Η PostgreSQL ανταποκρίνεται άριστα στις απαιτήσεις της εφαρμογής μας. Παρέχει υψηλή αξιοπιστία και ακεραιότητα δεδομένων μέσω ACID συναλλαγών και περιορισμών αναφορών, στοιχείο κρίσιμο για τις ροές ενημέρωσης πολλαπλών πινάκων (πχ στην εφαρμογή μας: ανάθεση bundles, δημιουργία/διαγραφή ασκήσεων). Το σχεσιακό της μοντέλο ταιριάζει φυσικά στη δομή του προβλήματος, όπου συνυπάρχουν σαφείς σχέσεις 1–N και N–M μεταξύ χρηστών, bundles και ασκήσεων. Επιπλέον, η ευρεία διάδοση και η διαθεσιμότητα ώριμων εργαλείων διαχείρισης διευκολύνουν την ανάπτυξη και τη συντήρηση, ενώ η ενσωμάτωση σε περιβάλλοντα Docker/CI είναι ομαλή, υποστηρίζοντας την αυτοματοποίηση της ανάπτυξης.



Σχήμα 4.11: Σχήμα βάσης δεδομένων της εφαρμογής (οντότητες και συσχετίσεις).

Το σχήμα της βάσης δεδομένων αποτυπώνει τις βασικές οντότητες του συστήματος και τις μεταξύ τους σχέσεις. Κεντρικό ρόλο έχει ο πίνακας `user`, στον οποίο αποθηκεύονται τα στοιχεία όλων των χρηστών της εφαρμογής, συμπεριλαμβανομένων των ασθενών, των κλινικών και των διαχειριστών. Μέσα από τα πεδία του, όπως το `type` και το `clinician_id`, υποστηρίζεται τόσο η διάκριση των ρόλων όσο και η σύνδεση ασθενών με τον υπεύθυνο κλινικό.

Η λειτουργικότητα των προγραμμάτων αποκατάστασης υλοποιείται μέσω των πινάκων `exercise_bundle` και `exercise`. Κάθε `exercise_bundle` αναπαριστά μία ασκήση, ενώ ο πίνακας `exercise` αποθηκεύει τα επιμέρους βήματα της με τα χαρακτηριστικά τους (τίτλος, περιγραφή, σειρά εκτέλεσης και αρχεία πολυμέσων όπως εικόνες, ήχο και βίντεο). Η αντιστοίχιση των bundles σε χρήστες επιτυγχάνεται μέσω του πίνακα `user_bundle`, ο οποίος υλοποιεί μια σχέση πολλών–προς–πολλά (N–M). Επιπλέον, ο πίνακας `user_bundle_log` κα-

ταγράφει το ιστορικό αλληλεπίδρασης των χρηστών με τα bundles, παρέχοντας πληροφορίες για την πρόοδο και τον χρόνο εκτέλεσης κάθε βήματος.

Τέλος, ο πίνακας `clinician_patient_note` επιτρέπει την αποθήκευση σημειώσεων από κλινικούς για τους ασθενείς τους, ενισχύοντας την παρακολούθηση της θεραπείας και την τεκμηρίωση των παρεμβάσεων.

4.4.2 Backend

Επιλογή Τεχνολογιών: Node.js/Express.js

Το Node.js είναι ένα cross-platform, open-source *runtime* για JavaScript που βασίζεται στη μηχανή V8. Υιοθετεί ασύγχρονη, *event-driven* και *non-blocking I/O* αρχιτεκτονική, γεγονός που το καθιστά ιδιαίτερα αποδοτικό για δικτυακές υπηρεσίες και REST APIs όπου κυριαρχούν λειτουργίες εισόδου/εξόδου. Επιλέχθηκε διότι παρέχει υψηλή απόδοση σε I/O-δεσμευμένα φορτία, απλότητα ανάπτυξης και τη δυνατότητα ενιαίας γλώσσας προγραμματισμού σε όλη τη στοίβα (*frontend–backend*), μειώνοντας το γνωσιακό κόστος και επιταχύνοντας την υλοποίηση.[13]

Το Express.js είναι ένα μινιμαλιστικό και ευέλικτο web framework για Node.js, που προσφέρει βασικές δυνατότητες δρομολόγησης, χειρισμού αιτήσεων/απαντήσεων και οργάνωσης *middleware*. Προτιμήθηκε επειδή δεν επιβάλλει "βαριά" δομή, επιτρέποντας καθαρή αρχιτεκτονική και σταδιακή εξέλιξη της εφαρμογής. Παρέχει τα αναγκαία δομικά στοιχεία για την υλοποίηση καθαρών REST endpoints, με μικρό αποτύπωμα και καμπύλη μάθησης που ταιριάζει σε ταχείες, επεκτάσιμες υλοποιήσεις.[14]

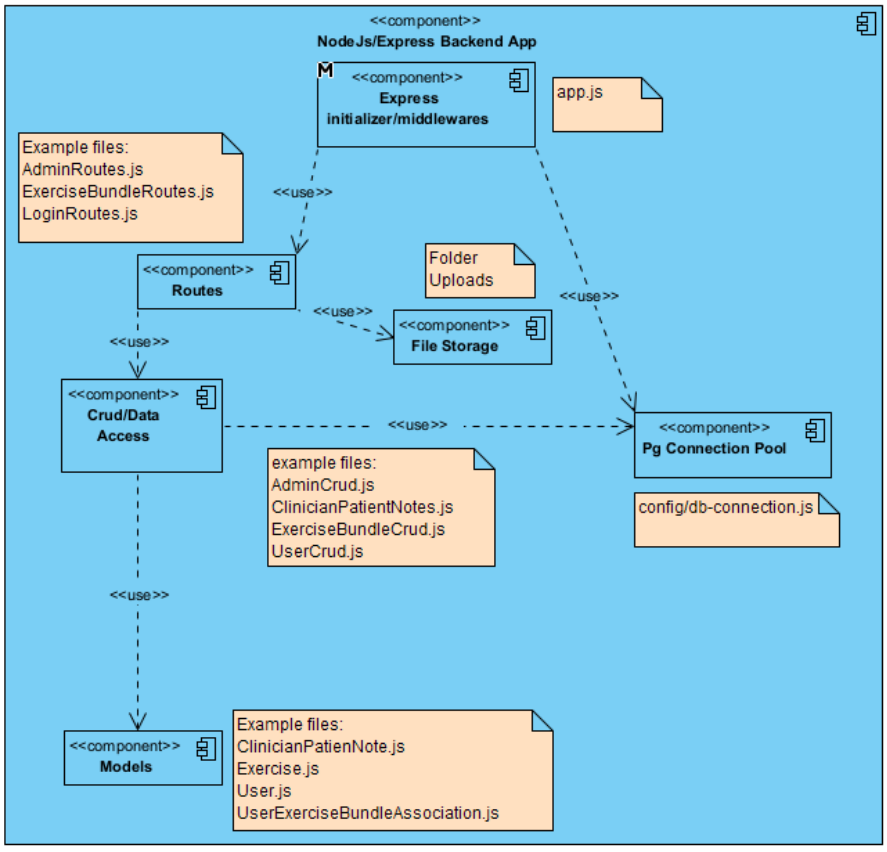
Βασικές βιβλιοθήκες. Για την υλοποίηση του backend αξιοποιήθηκαν επίσης ορισμένες βιβλιοθήκες που κάλυψαν κρίσιμες λειτουργίες:

- **Passport.js (Local Strategy) & `express-session`**: Η ασφάλεια και η ταυτοποίηση χρηστών στο σύστημα υλοποιήθηκαν με το Passport.js, ένα δημοφιλές middleware για το Node.js. Χρησιμοποιήθηκε η *local strategy*, όπου οι χρήστες συνδέονται με όνομα χρήστη και κωδικό, οι οποίοι ελέγχονται απευθείας στη βάση δεδομένων. Για να διατηρείται η κατάσταση του χρήστη μεταξύ των αιτημάτων, αξιοποιείται η βιβλιοθήκη `express-session`, η οποία δημιουργεί και διαχειρίζεται ένα session για κάθε συνδεδεμένο χρήστη. Με αυτόν τον τρόπο, ο χρήστης δεν χρειάζεται να δίνει τα στοιχεία του σε κάθε αίτημα. Η πρόσβαση σε ευαίσθητα endpoints προστατεύεται με ενδιάμεσα *middleware*, όπως οι συναρτήσεις `ensureAuthenticated` και `ensureAdmin`, που ελέγχουν αν ο χρήστης έχει συνδεθεί και αν διαθέτει τον κατάλληλο ρόλο (π.χ. διαχειριστής) πριν του επιτρέψουν να συνεχίσει.[15]
- **bcrypt**: Οι κωδικοί χρηστών αποθηκεύονται σε μορφή *hash* με χρήση της βιβλιοθήκης `bcrypt`, ώστε να μην διατηρούνται ποτέ σε καθαρό κείμενο. Κατά τη διαδικασία σύνδεσης, ο παρεχόμενος κωδικός μετατρέπεται σε ηαση και συγκρίνεται με το αποθηκευμένο ηαση για την επαλήθευση. [16]
- **Multer (File Uploads)**: Για τη μεταφόρτωση πολυμέσων χρησιμοποιείται το `multer` με *disk storage*. Τα αρχεία ταξινομούνται αυτόματα σε φακέλους ανά τύπο και ονοματοδοτούνται μοναδικά, ώστε να αποφεύγονται συγκρούσεις.[17]
- **CORS**: Το πακέτο `cors` χρησιμοποιείται για την ενεργοποίηση του μηχανισμού Cross-Origin Resource Sharing (CORS). Με αυτόν τον τρόπο, το frontend που τρέχει σε

διαφορετικό origin (π.χ. άλλο domain ή port από τον server του backend) μπορεί να επικοινωνεί ασφαλώς με το API. Χωρίς το CORS, ο φυλλομετρητής θα μπλόκαρε τις αιτήσεις λόγω πολιτικής ασφάλειας (Same-Origin Policy). Η ρύθμιση του πακέτου επιτρέπει μόνο τις επιτρεπόμενες πηγές και μεθόδους, διασφαλίζοντας ότι η πρόσβαση στο API γίνεται με ελεγχόμενο και ασφαλή τρόπο από το frontend της εφαρμογής.[17]

Δομή Backend

Το backend είναι χωρισμένο δομικά σε ξεκάθαρα τμήματα, όπως φαίνεται και στο Σχήμα 4.12, ώστε κάθε κομμάτι να έχει σαφώς καθορισμένο ρόλο.



Σχήμα 4.12: Σχηματική απεικόνιση των components του backend και των μεταξύ τους σχέσεων.

Η επικοινωνία με τη βάση δεδομένων PostgreSQL γίνεται μέσω του Pg Connection Pool, το οποίο αναλαμβάνει τη διαχείριση των συνδέσεων και εξασφαλίζει την αποδοτική χρήση τους.

Τα Models αντιστοιχούν σε πίνακες της βάσης και περιγράφουν τις βασικές οντότητες της εφαρμογής, όπως χρήστες, ασκήσεις και σύνολα ασκήσεων. Το στρώμα CRUD/Data Access αξιοποιεί τα Models και το Pg Pool για την υλοποίηση λειτουργιών δημιουργίας, ανάγνωσης, ενημέρωσης και διαγραφής, επιστρέφοντας τα αποτελέσματα σε μορφή αντικειμένων JavaScript.

Η αλληλεπίδραση με το σύστημα οργανώνεται μέσω των Routes, όπου συγκεντρώνονται τα API endpoints. Κάθε αίτημα προωθείται στις κατάλληλες συναρτήσεις του CRUD, ενώ το υποσύστημα File Storage αναλαμβάνει τη διαχείριση αρχείων πολυμέσων.

Τέλος, το Express Initializer/Middlewares περιλαμβάνει τις βασικές ρυθμίσεις της εφαρμογής, την ενεργοποίηση απαραίτητων middlewares και την εκκίνηση του διακομιστή μέσω του αρχείου `app.js`. Με αυτήν τη διάρθρωση, το backend παραμένει οργανωμένο, επεκτάσιμο και εύκολο στη συντήρηση.

4.4.3 Frontend

Επιλογή Τεχνολογιών: React.

Η React είναι μία δημοφιλής ανοιχτού κώδικα βιβλιοθήκη για την ανάπτυξη διαδραστικών διεπαφών χρήστη (*User Interfaces*), αναπτυγμένη από την Meta. Υιοθετεί *component-based* αρχιτεκτονική, όπου η διεπαφή χωρίζεται σε επαναχρησιμοποιούμενα δομικά στοιχεία (*components*) που διαχειρίζονται την εσωτερική τους κατάσταση (*state*) και αποδίδονται αποδοτικά μέσω του *Virtual DOM*. Η φιλοσοφία αυτή επιτρέπει την εύκολη οργάνωση πολύπλοκων UI, την επαναχρησιμοποίηση κώδικα και την ταχύτερη ανάπτυξη.[18]

Επιλέξαμε τη React για την υλοποίηση του frontend καθώς είναι ευρέως διαδεδομένη, καλοτεκμηριωμένη και υποστηρίζεται από μια μεγάλη κοινότητα. Η αρχιτεκτονική με *components* ευνοεί την καθαρή οργάνωση της εφαρμογής ανάμεσα σε οθόνες (π.χ. για ασθενή, κλινικού και διαχειριστή) και επαναχρησιμοποιούμενα στοιχεία (π.χ. `Header`, `Footer`, `MediaViewer`). Η ύπαρξη έτοιμων βιβλιοθηκών και η απρόσκοπτη ενσωμάτωση με το React Router και το Context API επιταχύνουν την ανάπτυξη, ενώ η δημοφιλία της τεχνολογίας καθιστά ευκολότερη τη μελλοντική συντήρηση και εξέλιξη.

Βασικές βιβλιοθήκες. Για την υλοποίηση του frontend αξιοποιήθηκαν δύο βασικές βιβλιοθήκες:

- **React Router**: Χρησιμοποιήθηκε για τη διαχείριση της πλοήγησης ανάμεσα στις σελίδες της εφαρμογής. Επιτρέπει τον ορισμό διαφορετικών routes που αντιστοιχούν σε οθόνες (π.χ. σελίδες ασθενή, κλινικού, διαχειριστή) και υποστηρίζει μηχανισμούς προστασίας μέσω του `ProtectedRoute`, ώστε η πρόσβαση σε ευαίσθητες σελίδες να επιτρέπεται μόνο σε αυθεντικοποιημένους χρήστες με τον κατάλληλο ρόλο.[19]
- **Context API**: Χρησιμοποιήθηκε για την κεντρική διαχείριση της κατάστασης ταυτοποίησης μέσω του `AuthContext`. Με αυτόν τον τρόπο, οι πληροφορίες σύνδεσης του χρήστη (π.χ. αν είναι ασθενής ή κλινικός) γίνονται άμεσα διαθέσιμες σε όλα τα components. Έτσι διασφαλίζεται συνέπεια και απλοποιείται η ανάπτυξη.[20]

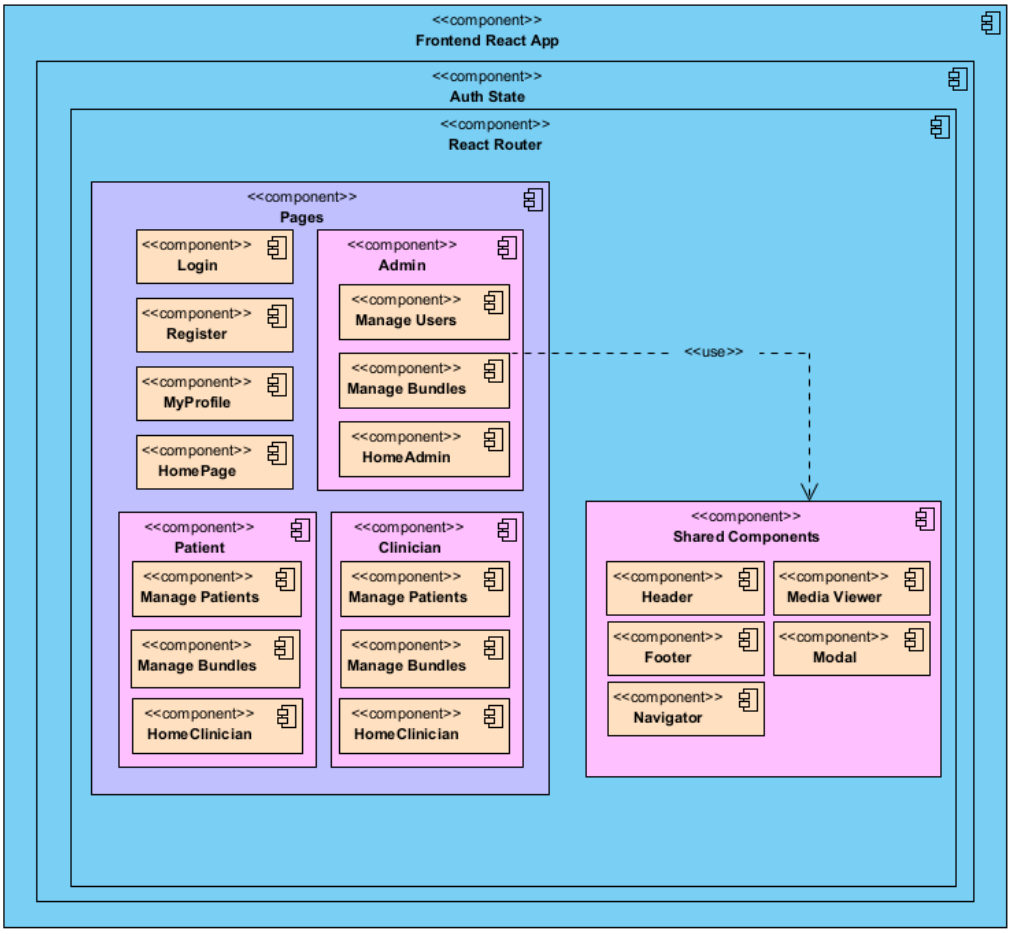
Δομή Frontend.

Στο Σχήμα 4.13 παρουσιάζεται η δομή του frontend, οργανωμένη σε τρία βασικά επίπεδα. Το `Frontend React App` αποτελεί τον εξωτερικό «container» μέσα στον οποίο εντάσσεται το `Auth State`, υπεύθυνο για τη διαχείριση της κατάστασης σύνδεσης του χρήστη, και το `React Router`, που καθορίζει την πλοήγηση ανάμεσα στις σελίδες.

Τα `Pages` περιλαμβάνουν τις βασικές οθόνες της εφαρμογής, χωρισμένες ανά ρόλο: ασθενής, κλινικός και διαχειριστής, καθώς και κοινές σελίδες όπως η είσοδος (*Login/Register*), το *MyProfile* και η αρχική σελίδα. Οι σελίδες αξιοποιούν επαναχρησιμοποιούμενα στοιχεία που

βρίσκονται στο πακέτο `Shared Components`, όπως το `Header`, το `Footer`, το `Navigator`, τα `Modal` και το `Media Viewer`, διασφαλίζοντας ομοιομορφία στο UI και μείωση του κώδικα.

Η ροή λειτουργεί ως εξής: όταν ένας χρήστης συνδέεται, το `Auth State` ενημερώνεται με τις πληροφορίες του και καθορίζει ποια σελίδα μπορεί να δει. Το `React Router` ελέγχει τα δικαιώματα πρόσβασης (μέσω `ProtectedRoute`) και προβάλλει την κατάλληλη ομάδα σελίδων ανάλογα με τον ρόλο του χρήστη. Σε όλες τις σελίδες, τα `Shared Components` εμφανίζονται με συνεπή τρόπο, παρέχοντας κοινά στοιχεία πλοήγησης και παρουσίασης.



Σχήμα 4.13: Σχηματική απεικόνιση των components του frontend και της μεταξύ τους σχέσης.

4.4.4 Deployment

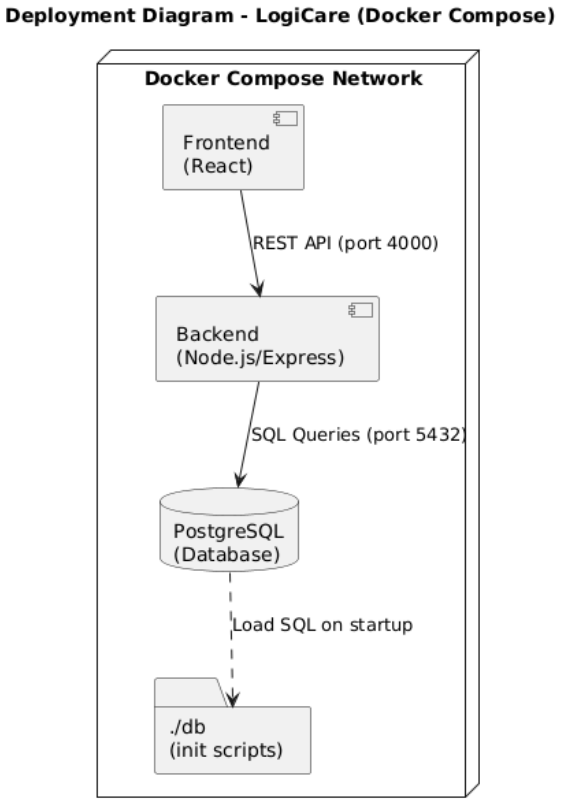
Επιλογή Τεχνολογιών: Docker

Το Docker είναι μία πλατφόρμα ανοιχτού κώδικα για την ανάπτυξη, διανομή και εκτέλεση εφαρμογών μέσα σε ελαφριά απομονωμένα περιβάλλοντα, τα *containers*. Κάθε container περιέχει όλα τα απαραίτητα εξαρτήματα (κώδικα, βιβλιοθήκες, ρυθμίσεις), εξασφαλίζοντας συνέπεια στην εκτέλεση ανεξαρτήτως λειτουργικού συστήματος. Η χρήση Docker Compose επιτρέπει τον ορισμό και τη διαχείριση πολλών containers ως ενιαία εφαρμογή.[21]

Η επιλογή του Docker για το deployment της εφαρμογής κρίθηκε ιδανική, καθώς εξασφαλίζει επαναληψιμότητα και φορητότητα. Με ένα μόνο αρχείο `docker-compose.yml`, μπορούμε να δημιουργούμε γρήγορα ένα πλήρες περιβάλλον που περιλαμβάνει τη βάση δεδομένων PostgreSQL, το backend (Node.js/Express) και το frontend (React). Με αυτόν τον τρόπο, αποφεύγονται προβλήματα «διαφορετικών εκδόσεων» ή σύνθετων τοπικών ρυθμίσεων, ενώ η ενσωμάτωση σε CI/CD pipelines γίνεται ομαλά.

Διάγραμμα Deployment

Στο Σχήμα 4.14 παρουσιάζεται το διάγραμμα των components του deployment της εφαρμογής μέσω Docker Compose. Η υλοποίηση περιλαμβάνει τρία βασικά containers: το frontend (React), το backend (Node.js/Express) και τη βάση δεδομένων PostgreSQL. Όλα τα containers εκκινούν και συντονίζονται από το Docker Compose, το οποίο φροντίζει για τη δημιουργία του εσωτερικού δικτύου και τον καθορισμό των παραμέτρων εκτέλεσης.



Σχήμα 4.14: Deployment diagram της εφαρμογής

Κατά την εκτέλεση, το frontend επικοινωνεί με το backend μέσω REST API (θύρα 4000), ενώ το backend συνδέεται με τη βάση δεδομένων PostgreSQL (θύρα 5432) για την εκτέλεση ερωτημάτων. Η βάση δεδομένων φορτώνει αυτόματα αρχικοποιητικά δεδομένα, όπως τα στοιχεία του αδμιν και όλο το δοσμένο από το σύστημα ασκησεολόγιο από τον φάκελο `./db`, τα οποία εκτελούνται κατά την εκκίνηση του container. Με αυτήν τη διάρθρωση, η εφαρμογή μπορεί να σηκωθεί πλήρως σε οποιοδήποτε περιβάλλον με μία μόνο εντολή `docker-compose up`.

4.5 Endpoints του API

Μετά την ανάλυση των απαιτήσεων, τη μοντελοποίηση της ροής με *Activity Diagram* και την παρουσίαση των *Component* διαγραμμάτων, όπου τεκμηριώθηκαν οι επιλογές τεχνολογιών και οι αλληλεπιδράσεις των υποσυστημάτων, σε αυτή την ενότητα περνάμε στη δημόσια διεπαφή της εφαρμογής: τα *endpoints* του API. Στόχος είναι να δείξουμε πώς οι ροές που περιγράψαμε μετατρέπονται σε συγκεκριμένες κλήσεις HTTP, ώστε το θεωρητικό μοντέλο να γίνεται πρακτικός οδηγός για τον πελάτη (frontend).

Στην υλοποίησή μας, τα αιτήματα και οι απαντήσεις ανταλλάσσονται σε JSON (με εξαίρεση `multipart/form-data` για μεταφορτώσεις αρχείων), η αυθεντικοποίηση είναι *session-based* και υπάρχει διαχωρισμός ανάμεσα σε μεταδεδομένα και πολυμεσικό περιεχόμενο.

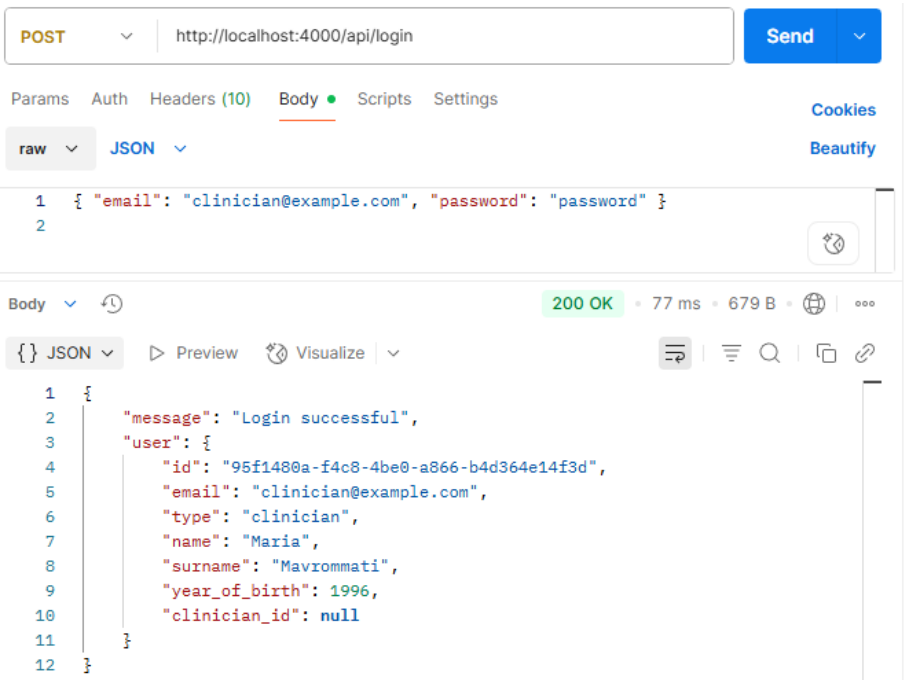
4.5.1 Βασικές κατηγορίες endpoints

Παρουσιάζουμε μερικά βασικά endpoints, με ενδεικτικές κλήσεις και στιγμιότυπα από το Postman.

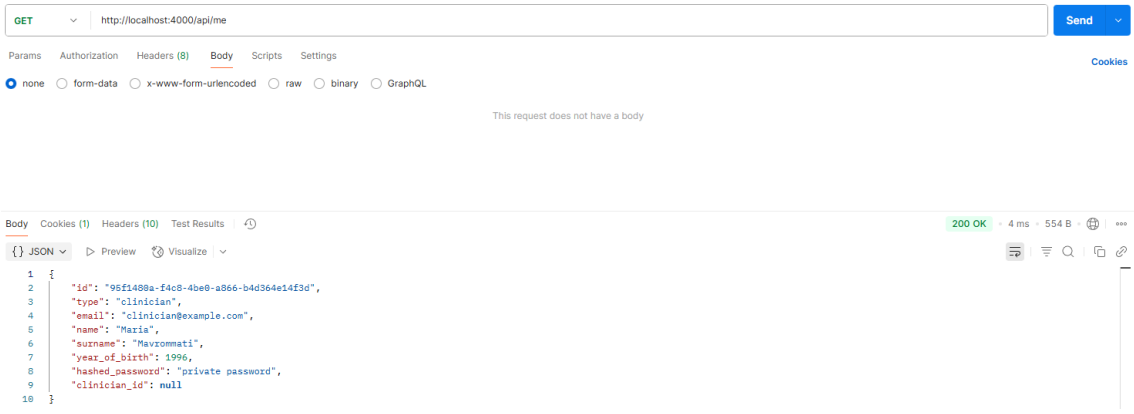
Αυθεντικοποίηση και ροές κλινικού/ασθενούς

Η κατηγορία αυτή περιλαμβάνει τη διαχείριση σύνδεσης και ταυτοποίησης, καθώς και λειτουργίες που συνδέουν τον κλινικό με τους ασθενείς του.

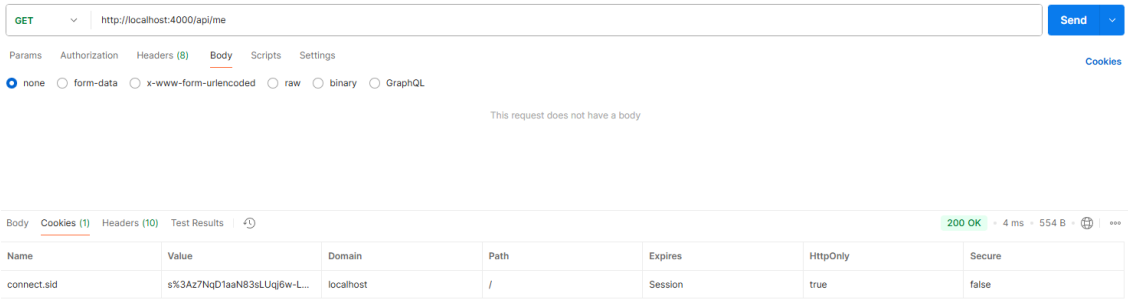
Είσοδος χρήστη (login). Η κλήση `POST /api/login` επιστρέφει τα στοιχεία του χρήστη και εγκαθιστά session, το οποίο αποθηκεύεται ως cookie στον πελάτη (Σχ. 4.15–4.17). Αντίστοιχα, η `GET /api/me` επιστρέφει τα στοιχεία του ενεργού χρήστη, ενώ η `POST /api/logout` τερματίζει τη συνεδρία. Σε περίπτωση που ένας μη εξουσιοδοτημένος χρήστης καλέσει admin-only endpoint, όπως `GET /api/admin/users`, ο διακομιστής επιστρέφει 403 `Forbidden` (Σχ. 4.18).



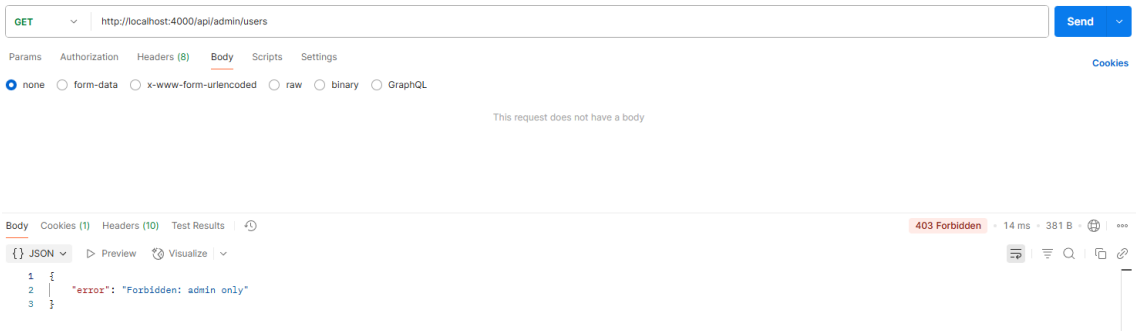
Σχήμα 4.15: Κλήση του `POST /api/login` στο Postman: επιτυχής ταυτοποίηση.



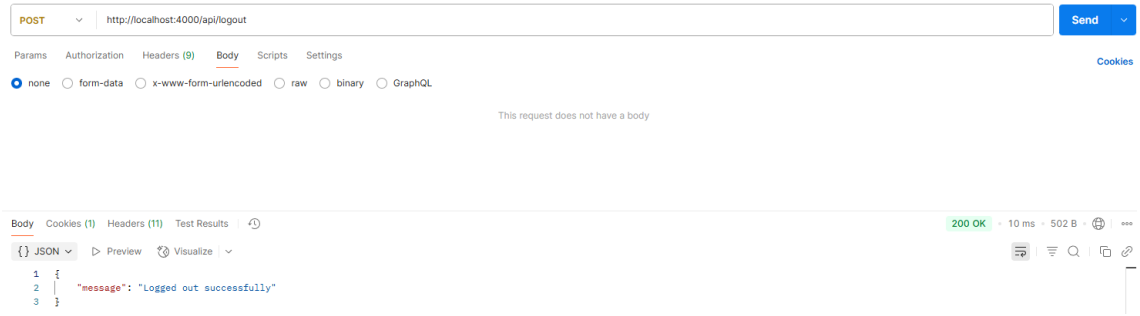
Σχήμα 4.16: Κλήση του `GET /api/me`: επιστροφή στοιχείων συνδεδεμένου χρήστη.



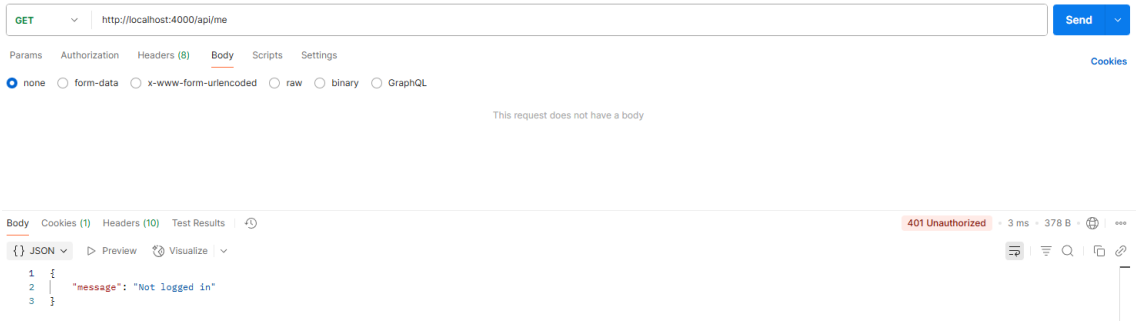
Σχήμα 4.17: Μετά το επιτυχές `POST /api/login`, το Postman αποθηκεύει το cookie συνεδρίας που επιστρέφει ο διακομιστής.



Σχήμα 4.18: Δοκιμή admin-only κλήσης `GET /api/admin/users`: επιστρέφεται σφάλμα εξουσιοδότησης.



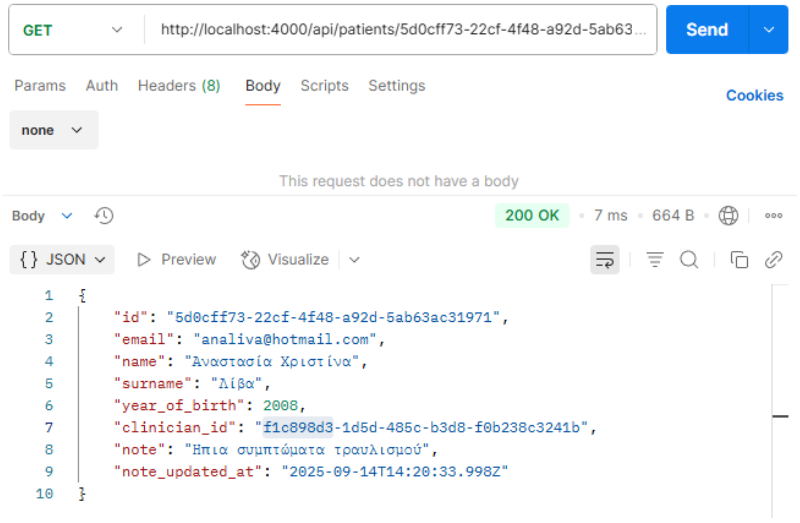
Σχήμα 4.19: Κλήση `POST /api/logout`: λήξη συνεδρίας.



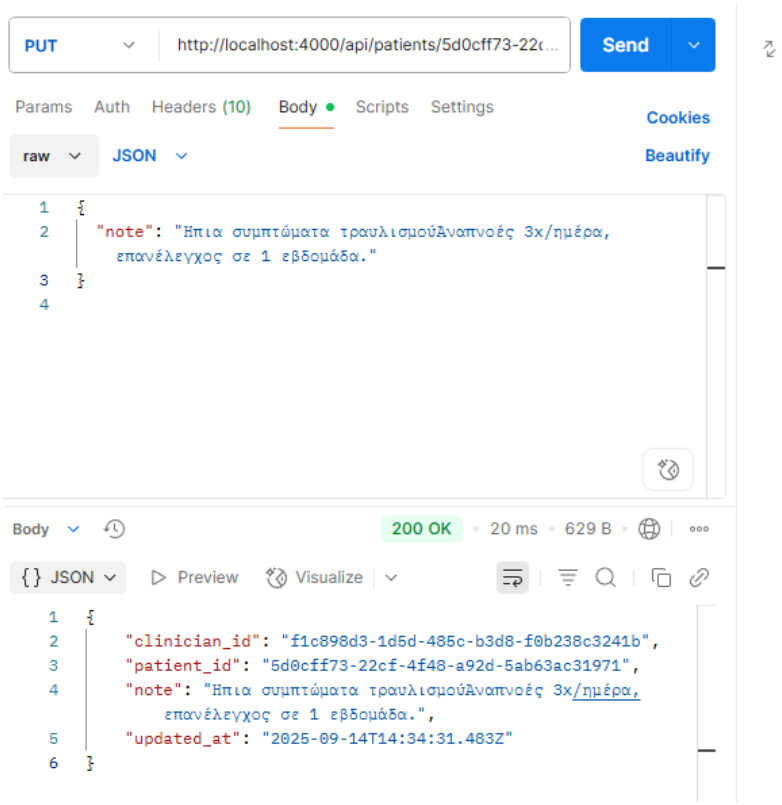
Σχήμα 4.20: Μετά το `logout`, κλήση `GET /api/me`: αναμενόμενο `401 Unauthorized`.

Σημειώσεις κλινικού.

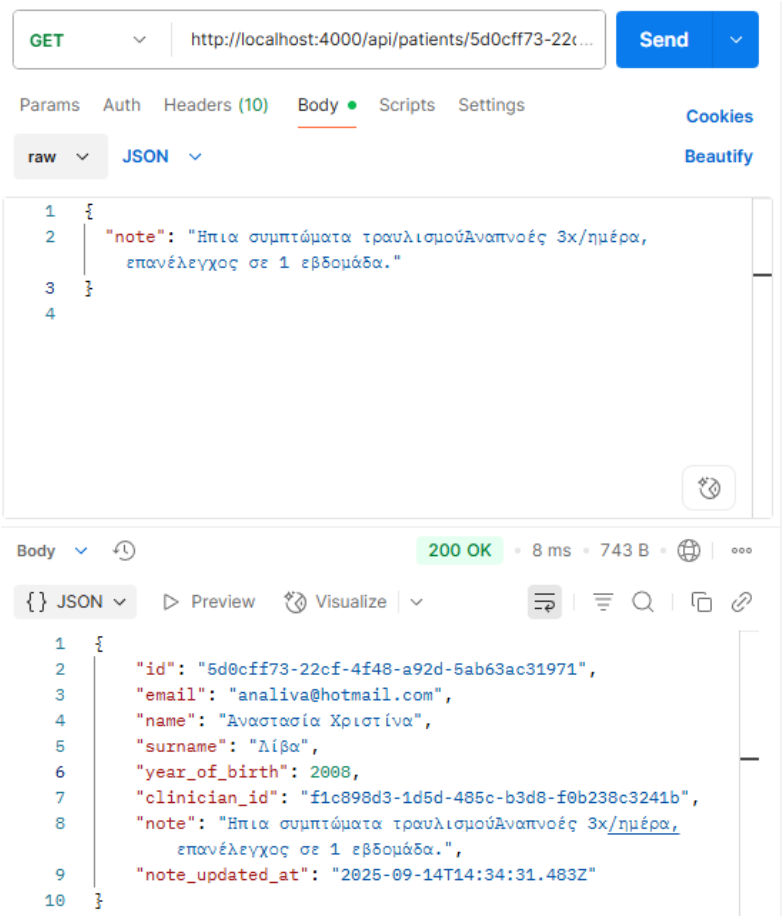
Η κλήση `PUT /api/patients/:id/note` επιτρέπει στον κλινικό να καταγράφει προσωπικές σημειώσεις για κάθε ασθενή (Σχ. 4.21–4.23). Οι σημειώσεις αυτές αποθηκεύονται στη βάση δεδομένων και είναι ορατές αποκλειστικά στον ίδιο, ώστε να λειτουργούν ως προσωπικό εργαλείο παρακολούθησης της θεραπευτικής πορείας.



Σχήμα 4.21: Προβολή ασθενούς πριν την ανανέωση σημείωσης.



Σχήμα 4.22: Κλήση του `PUT /api/patients/:id/note` με σώμα αιτήματος σε μορφή JSON.



Σχήμα 4.23: Προβολή ασθενούς μετά την επιτυχή ενημέρωση της σημείωσης.

Δεδομένα ασκήσεων Με το `GET /api/exercises/:id` επιστρέφονται τα βασικά στοιχεία μιας άσκησης, όπως `id`, τίτλος, σειρά εκτέλεσης και περιγραφή (Σχ. 4.24). Για λόγους σαφήνειας, τα περιεχόμενα των `Buffer` που αντιστοιχούν σε πολυμέσα έχουν αποσιωπηθεί.

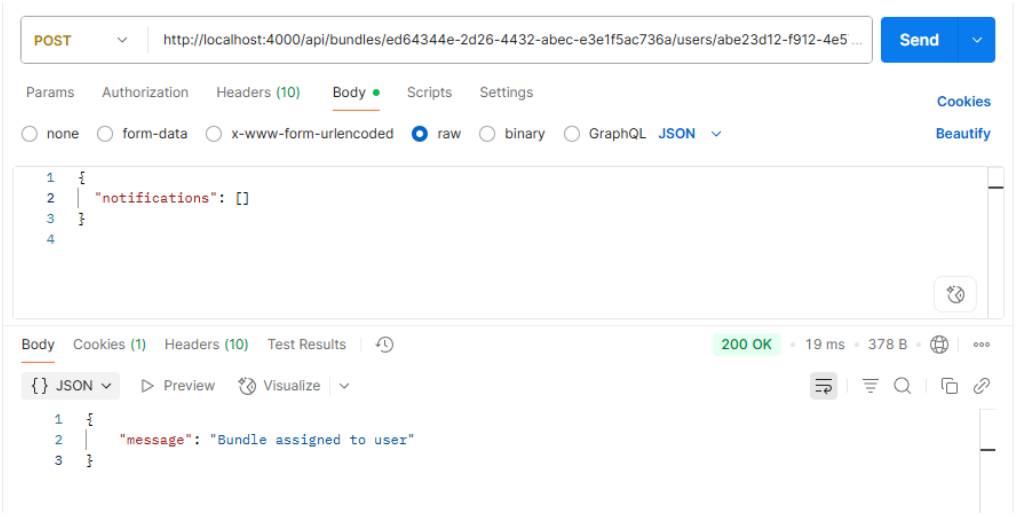
```
{
  "id": "e35a024c-98fc-4871-b5cb-a3919073359f",
  "title": "1",
  "exercises": [
    {
      "id": "4e04f249-05ef-458d-8835-bce217ac9a95",
      "bundle_id": "e35a024c-98fc-4871-b5cb-a3919073359f",
      "step": 1,
      "title": "Βήμα 1",
      "description": "Παράγετε ένα συνεχόμενο /a/ με όσο πιο δυνατή φωνή μπορείτε",
      "audio": {
        "type": "Buffer",
        "data": [0, 0, 0, 24, ...]
      },
      "picture": {
        "type": "Buffer",
        "data": [255, 0, 0, 24, ...]
      },
      "video_file_path": null
    },
    {
      "id": "e9228e52-f560-499c-81d2-eb52faaaa3b0",
      "bundle_id": "e35a024c-98fc-4871-b5cb-a3919073359f",
      "step": 2,
      "title": "Βήμα 2",
      "description": "Προσπαθήστε να ακούγεστε ίδια σε όλη τη διάρκεια της άσκησης",
      "audio": {
        "type": "Buffer",
        "data": [0, 0, 0, 24, ...]
      },
      "picture": {
        "type": "Buffer",
        "data": [255, 0, 0, 24, ...]
      },
      "video_file_path": null
    }
  ],
  "global": false
}
```


Σχήμα 4.24: Ενδεικτική απάντηση του `GET /api/exercises/:id`

Bundles

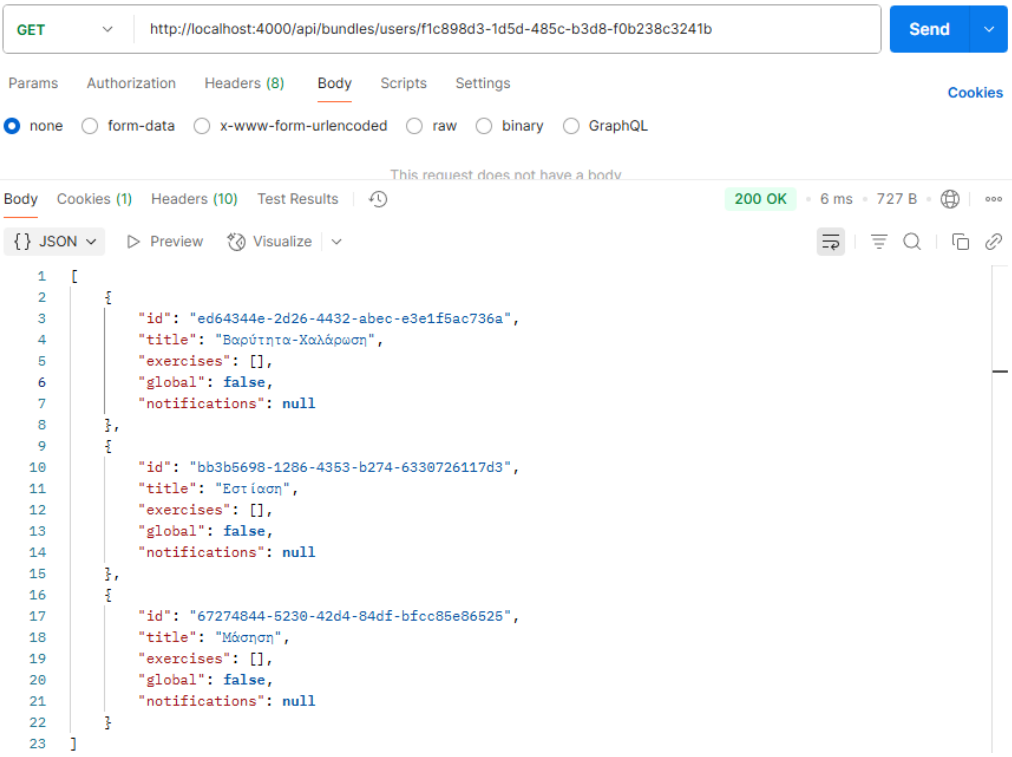
Οι ασκήσεις οργανώνονται σε bundles, ώστε να διευκολύνεται η συστηματική ανάθεση θεραπευτικών πακέτων σε ασθενείς.

Ανάθεση bundle σε ασθενή. Η κλήση `POST /api/bundles/:bundleId/users/:userId` συσχετίζει ένα συγκεκριμένο bundle με έναν χρήστη (Σχ. 4.25). Το σώμα του αιτήματος μπορεί να περιέχει το πεδίο `notifications`, ώστε να οριστούν οι μέρες ενημέρωσης του ασθενούς.



Σχήμα 4.25: Κλήση του `POST /api/bundles/:bundleId/users/:userId` στο Postman. Επιστροφή μηνύματος επιτυχούς ανάθεσης bundle σε χρήστη.

Λίστα bundles χρήστη. Η κλήση `GET /api/bundles/users/:userId` επιστρέφει τα bundles στα οποία ο χρήστης έχει δικαίωμα πρόσβασης (Σχ. 4.26). Η ορατότητα των bundles διαφοροποιείται ανάλογα με το ρόλο: ο ασθενής βλέπει μόνο τα bundles που του έχει αναθέσει ο κλινικός, ο κλινικός βλέπει τόσο όσα του έχουν ανατεθεί όσο και όσα έχει δημιουργήσει ο ίδιος, ενώ ο διαχειριστής (admin) έχει πρόσβαση σε όλα τα διαθέσιμα bundles του συστήματος.



Σχήμα 4.26: Κλήση του `GET /api/bundles/users/:userId` στο Postman. Επιστροφή λίστας bundles που σχετίζονται με τον χρήστη.

4.5.2 Πίνακες με όλα τα endpoints

Για λόγους πληρότητας αλλά και αποφυγής επαναλήψεων, δεν παρουσιάζουμε αναλυτικά στιγμιότυπα (screenshots) για κάθε κλήση, καθώς πολλά endpoints ακολουθούν την ίδια λογική με όσα έχουν ήδη δειχθεί. Αντί γι' αυτό, όλα τα υπόλοιπα endpoints παρουσιάζονται συγκεντρωτικά στους παρακάτω πίνακες, με συνοπτική περιγραφή, ρόλους και παραμέτρους αιτήματος.

Παρακάτω παρουσιάζονται συγκεντρωτικά όλοι οι πίνακες με τα endpoints της εφαρμογής. Οι πίνακες έχουν χωριστεί σε κατηγορίες για λόγους σαφήνειας: User, Exercise, Auth/Login, Bundle και Admin.

Endpoint	**Roles**	**Request**	**Περιγραφή**
POST /api/users	Κλινικός / Ασθενής	**Body**: type, email, password, year, name, surname	Εγγραφή νέου χρήστη στο σύστημα
GET /api/users/:id	Όλοι	**Path**: id	Ανάκτηση στοιχείων χρήστη βάσει `id`
PUT /api/users/:id	Όλοι	**Body**: type, email, name, surname, year_of_birth	Ενημέρωση στοιχείων χρήστη
DELETE /api/users/:id	Διαχειριστής	**Path**: id	Διαγραφή χρήστη
GET /api/my-patients	Κλινικός	—	Λίστα ασθενών που παρακολουθεί ο κλινικός
POST /api/assign-patient	Κλινικός	**Body**: name, surname, year_of_birth, note	Ανάθεση νέου ασθενούς σε κλινικό
GET /api/patients/:id	Κλινικός	**Path**: id	Προβολή στοιχείων ασθενούς (και σημειώσεων)
PUT /api/patients/:id/note	Κλινικός	**Body**: note	Καταχώριση ή ενημέρωση σημείωσης κλινικού
PUT /api/users/:id/password	Όλοι	**Body**: oldPassword, newPassword	Αλλαγή κωδικού πρόσβασης χρήστη
GET /api/users/notifications/:id	Όλοι	**Path**: id	Επιστροφή εβδομαδιαίων ειδοποιήσεων χρήστη

Table 4.1: Endpoints Χρήστη

Endpoint	**Roles**	**Request**	**Περιγραφή**
GET /api/exercises/:id	Όλοι	**Path**: id	Ανάκτηση μεταδεδομένων άσκησης (περιλαμβάνει και πολυμέσα ως buffers)
PUT /api/exercises/:id	Κλινικός / Διαχειριστής	**Body**: title, description, step, ...	Ενημέρωση μεταδεδομένων άσκησης
DELETE /api/exercises/:id	Κλινικός / Διαχειριστής	**Path**: id	Διαγραφή άσκησης
POST /api/exercises	Κλινικός / Διαχειριστής	**Body (multipart)**: title, description, step, audio/picture/video	Δημιουργία νέας άσκησης με πολυμέσα
GET /api/exercises/:id/audio	Όλοι	**Path**: id	Λήψη αρχείου ήχου
GET /api/exercises/:id/picture	Όλοι	**Path**: id	Λήψη εικόνας
GET /api/exercises/:id/video	Όλοι	**Path**: id	Λήψη βίντεο

Table 4.2: Exercise Endpoints

Endpoint	Roles	Request	Περιγραφή
POST /api/login	Όλοι	**Body**: email, password	Είσοδος χρήστη και δημιουργία session
POST /api/logout	Όλοι	—	Έξοδος χρήστη και λήξη session
GET /api/me	Όλοι	—	Επιστροφή στοιχείων του τρέχοντος συνδεδεμένου χρήστη

Table 4.3: Auth / Login Endpoints

Endpoint	Roles	Request	Περιγραφή
POST /api/bundles	Διαχειριστής	**Body**: title, global	Δημιουργία νέου bundle
GET /api/bundles/:id	Όλοι	**Path**: id	Ανάκτηση πλήρους bundle μαζί με όλες τις ασκήσεις του
PUT /api/bundles/:id	Διαχειριστής	**Body**: title, global	Ενημέρωση μεταδεδομένων bundle
DELETE /api/bundles/:id	Κλινικός / Διαχειριστής	**Path**: id	Διαγραφή bundle από το σύστημα
GET /api/bundles/users/:userId	Όλοι	**Path**: userId	Επιστροφή όλων των bundles που είναι ορατά στον χρήστη

Table 4.4: Bundle Endpoints (Μέρος Α)

Endpoint	**Roles**	**Request**	**Περιγραφή**
POST /api/bundles/:bundleId/users/:userId	Κλινικός	**Body**: notifications	Ανάθεση bundle σε χρήστη, με δυνατότητα ειδοποιήσεων
DELETE /api/bundles/:bundleId/users/:userId	Κλινικός	—	Αφαίρεση bundle από χρήστη
POST /api/bundles/-clinician	Κλινικός	**Body**: title	Δημιουργία προσωπικού bundle στον χώρο του κλινικού
POST /api/bundles/:id/clone-for-me	Κλινικός	**Path**: id	Κλωνοποίηση global bundle στον χώρο του κλινικού
GET /api/globalexercises	Κλινικός / Διαχειριστής	—	Λίστα όλων των global bundles
POST /api/log/bundles/:bundleId/users/:userId	Ασθενής	**Body**: state, step, timestamp	Καταγραφή εκτέλεσης bundle από χρήστη

Table 4.5: Bundle Endpoints (Μέρος Β)

Endpoint	**Request**	**Περιγραφή**
GET /api/admin/users	**Query**: role, search	Επιστροφή λίστας χρηστών με φίλτρα
DELETE /api/admin/users/:id	**Path**: id	Οριστική διαγραφή χρήστη
PUT /api/admin/users/:id/reset-password	**Path**: id	Επαναφορά κωδικού χρήστη με προσωρινό password
GET /api/admin/bundles	—	Λίστα όλων των global bundles
POST /api/admin/bundles	**Body**: title	Δημιουργία νέου global bundle
PUT /api/admin/bundles/:id	**Body**: title	Ενημέρωση μεταδεδομένων global bundle
DELETE /api/admin/bundles/:id	**Path**: id	Διαγραφή global bundle
GET /api/admin/stats	—	Ανάκτηση βασικών στατιστικών (χρήστες, bundles, αναθέσεις)

Table 4.6: Admin Endpoints

4.6 Sequence Diagrams

Έχοντας προηγουμένως αναλύσει το σύνολο των διαθέσιμων endpoints της εφαρμογής και τον τρόπο με τον οποίο κάθε ρόλος αλληλεπιδρά με το σύστημα μέσω αυτών, στη συνέχεια περνάμε σε μία πιο δυναμική και λειτουργική αποτύπωση της ροής. Συγκεκριμένα, παρουσιάζουμε τα *Sequence Diagrams*, τα οποία αποτελούν ένα από τα βασικά διαγράμματα της UML για την κατανόηση και τεκμηρίωση της συμπεριφοράς ενός πληροφοριακού συστήματος.

Τα Sequence Diagrams επικεντρώνονται στον χρονικό άξονα και αποτυπώνουν τη διαδοχική ροή μηνυμάτων μεταξύ διαφορετικών οντοτήτων (lifelines), όπως ο τελικός χρήστης, το Frontend, το Backend και η Βάση Δεδομένων. Με γραφικό τρόπο δείχνουν ποιος καλεί ποιον, με ποια κλήση (π.χ. HTTP method/path) και ποια απάντηση επιστρέφεται. Η οπτική αυτή προσφέρει όχι μόνο μία σαφή εικόνα της αλληλεπίδρασης μεταξύ των επιμέρους επιπέδων της αρχιτεκτονικής, αλλά και μια άμεση κατανόηση των εναλλακτικών ροών, σεναρίων αποτυχίας ή επαναλήψεων ενεργειών.

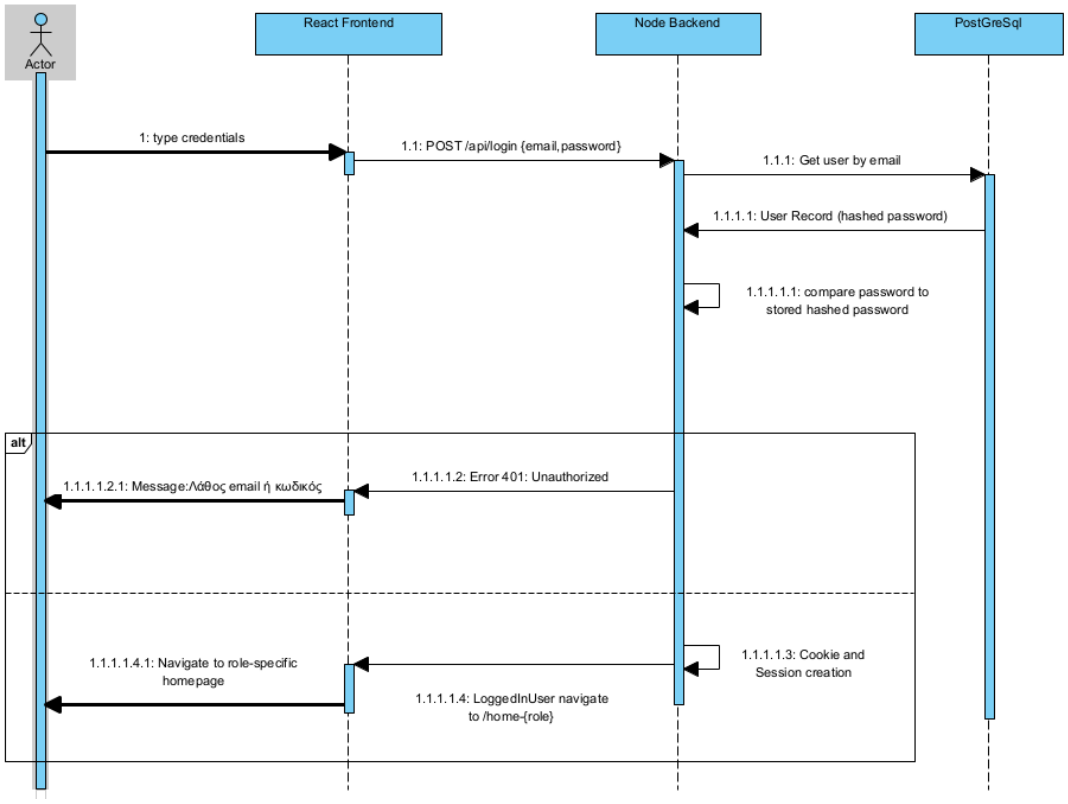
Ειδικά στη δική μας εφαρμογή, τα διαγράμματα αυτά είναι πολύτιμα για να φανεί πώς οι διαφορετικοί ρόλοι (ασθενής, κλινικός, διαχειριστής) ακολουθούν συγκεκριμένες ροές και ποια είναι τα βασικά σημεία ελέγχου και επικοινωνίας με το σύστημα. Μέσα από στοιχεία όπως τα alt fragments (εναλλακτικές ροές, π.χ. μη έγκυρα διαπιστευτήρια) και τα loop fragments (επαναλαμβανόμενες ενέργειες, π.χ. προσθήκη πολλών βημάτων), μπορούμε να περιγράψουμε με ακρίβεια την πραγματική λειτουργία του συστήματος.

Στο παρόν κεφάλαιο δεν παρουσιάζουμε όλα τα πιθανά διαγράμματα (καθώς θα ήταν υπερβολικά πολλά και σε μεγάλο βαθμό πλεονάζοντα), αλλά επιλέγουμε ορισμένα χαρακτηριστικά και ενδεικτικά σενάρια. Αρχικά περιγράφουμε δύο βασικά διαγράμματα που αφορούν όλους τους χρήστες (Login, Edit Profile), στη συνέχεια παρουσιάζουμε διαγράμματα που επικεντρώνονται σε ενέργειες του ασθενή και του κλινικού, και τέλος καταλήγουμε σε ένα σενάριο διαχειριστή (Admin), που αφορά τη δημιουργία ενός global bundle. Με τον τρόπο αυτό δίνεται μία πλήρης αλλά και ισορροπημένη εικόνα της λειτουργικότητας του συστήματος.

4.6.1 Γενικά διαγράμματα (κοινά σε όλους τους ρόλους)

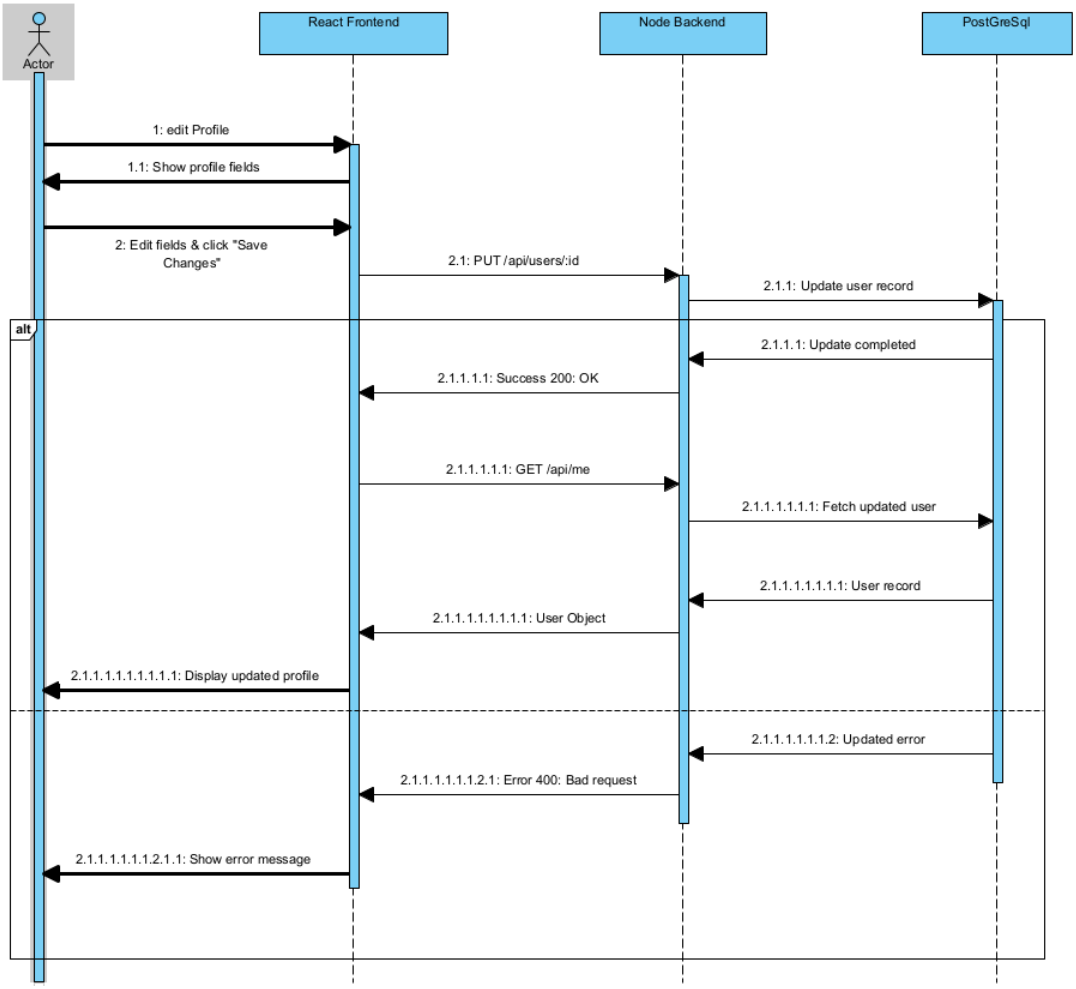
Στην κατηγορία αυτή περιλαμβάνονται βασικά σενάρια που αφορούν όλους τους χρήστες ανεξαρτήτως ρόλου, καθώς αποτελούν κοινά σημεία εισόδου και αλληλεπίδρασης με το σύστημα. Δύο χαρακτηριστικά παραδείγματα είναι η διαδικασία σύνδεσης (Login) και η τροποποίηση προσωπικών στοιχείων (Edit Profile). Τα διαγράμματα που ακολουθούν παρουσιάζουν αναλυτικά τη ροή των ενεργειών και τις εναλλακτικές περιπτώσεις σφάλματος.

Login. Το πρώτο διάγραμμα αποτυπώνει τη βασική διαδικασία σύνδεσης. Ο χρήστης εισάγει τα διαπιστευτήριά του στο Frontend, το οποίο αποστέλλει αίτημα `POST /api/login` προς το Backend. Εκεί γίνεται αναζήτηση του χρήστη στη βάση δεδομένων και έλεγχος του κωδικού με τον αποθηκευμένο (hashed) κωδικό. Αν η σύγκριση αποτύχει, επιστρέφεται `401 Unauthorized` με μήνυμα λάθους (alt fragment). Στην περίπτωση επιτυχίας, δημιουργείται session/cookie και ο χρήστης ανακατευθύνεται αυτόματα στην αρχική σελίδα που αντιστοιχεί στον ρόλο του.



Σχήμα 4.27: Διάγραμμα ακολουθίας Login με alt (σφάλμα) και επιτυχή δημιουργία session.

Edit Profile. Το δεύτερο διάγραμμα παρουσιάζει τη διαδικασία τροποποίησης στοιχείων προφίλ. Ο χρήστης ανοίγει τη φόρμα επεξεργασίας και υποβάλλει τις αλλαγές μέσω `PUT /api/users/:id`. Το Backend ενημερώνει τη βάση δεδομένων και, σε επιτυχία, το Frontend ανακτά τα επικαιροποιημένα δεδομένα με κλήση `GET /api/me`, τα οποία και εμφανίζονται στον χρήστη. Η εναλλακτική ροή alt αναπαριστά περίπτωση σφάλματος επικύρωσης (400), όπου το σύστημα επιστρέφει σχετικό μήνυμα λάθους αντί για ενημερωμένο προφίλ.

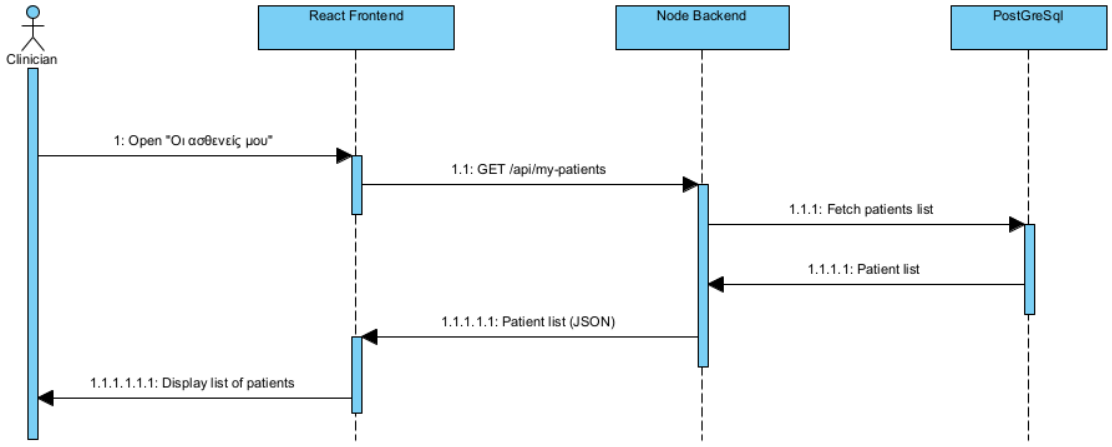


Σχήμα 4.28: Ενημέρωση προφίλ (PUT /api/users/:id → GET /api/me) με εναλλακτική ροή σφάλματος.

Διαγράμματα για συγκεκριμένους ρόλους

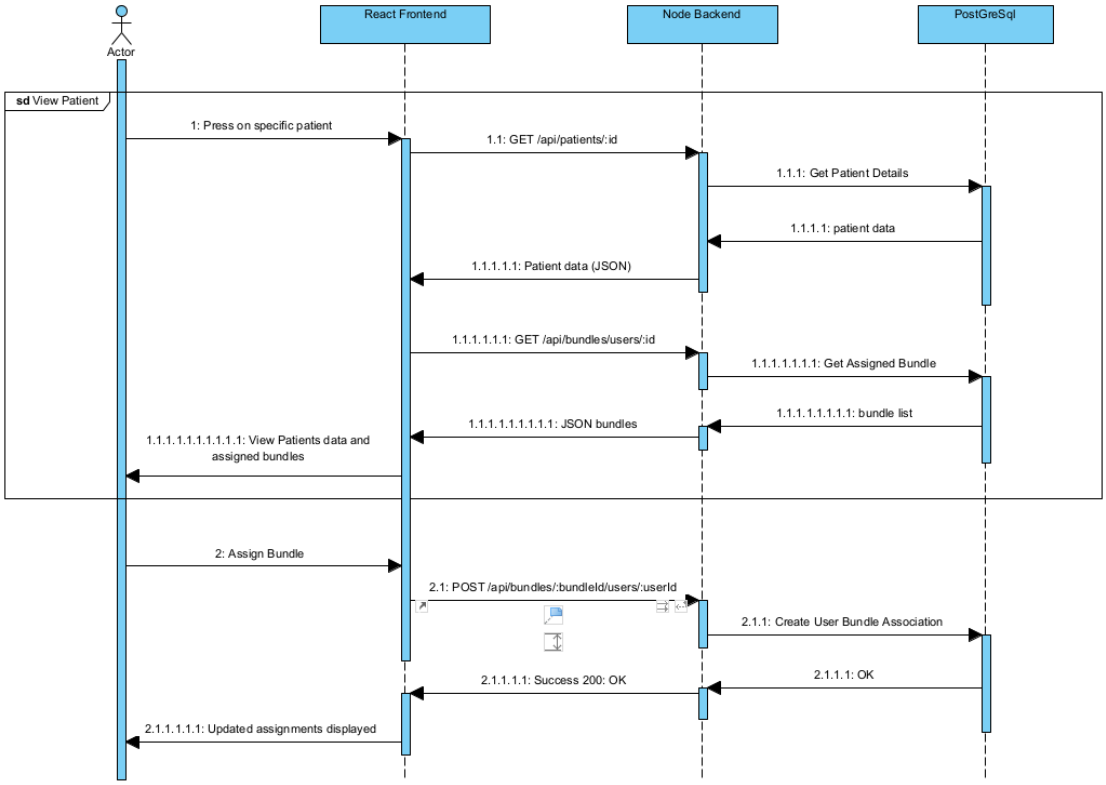
Πέρα από τα γενικά σενάρια που αφορούν κάθε χρήστη, υπάρχουν και πιο εξειδικευμένες ροές που συνδέονται με τον ρόλο του καθενός. Στη συνέχεια παρουσιάζουμε ένα χαρακτηριστικό διάγραμμα για τον ασθενή, δύο για τον κλινικό και ένα για τον διαχειριστή. Η επιλογή έγινε με γνώμονα την αντιπροσωπευτικότητα: τα παραδείγματα δείχνουν την αλληλουχία ενεργειών σε βασικές λειτουργίες της εφαρμογής, χωρίς να καταγράφεται κάθε πιθανή παραλλαγή που θα οδηγούσε σε πλεονασμό.

Κλινικός: Προβολή ασθενών. Ο κλινικός επιλέγει «Οι ασθενείς μου» και το Frontend στέλνει αίτημα `GET /api/my-patients` στο Backend. Η βάση δεδομένων επιστρέφει τη λίστα με τους ασθενείς και τα στοιχεία τους, τα οποία αποτυπώνονται στο περιβάλλον χρήστη.



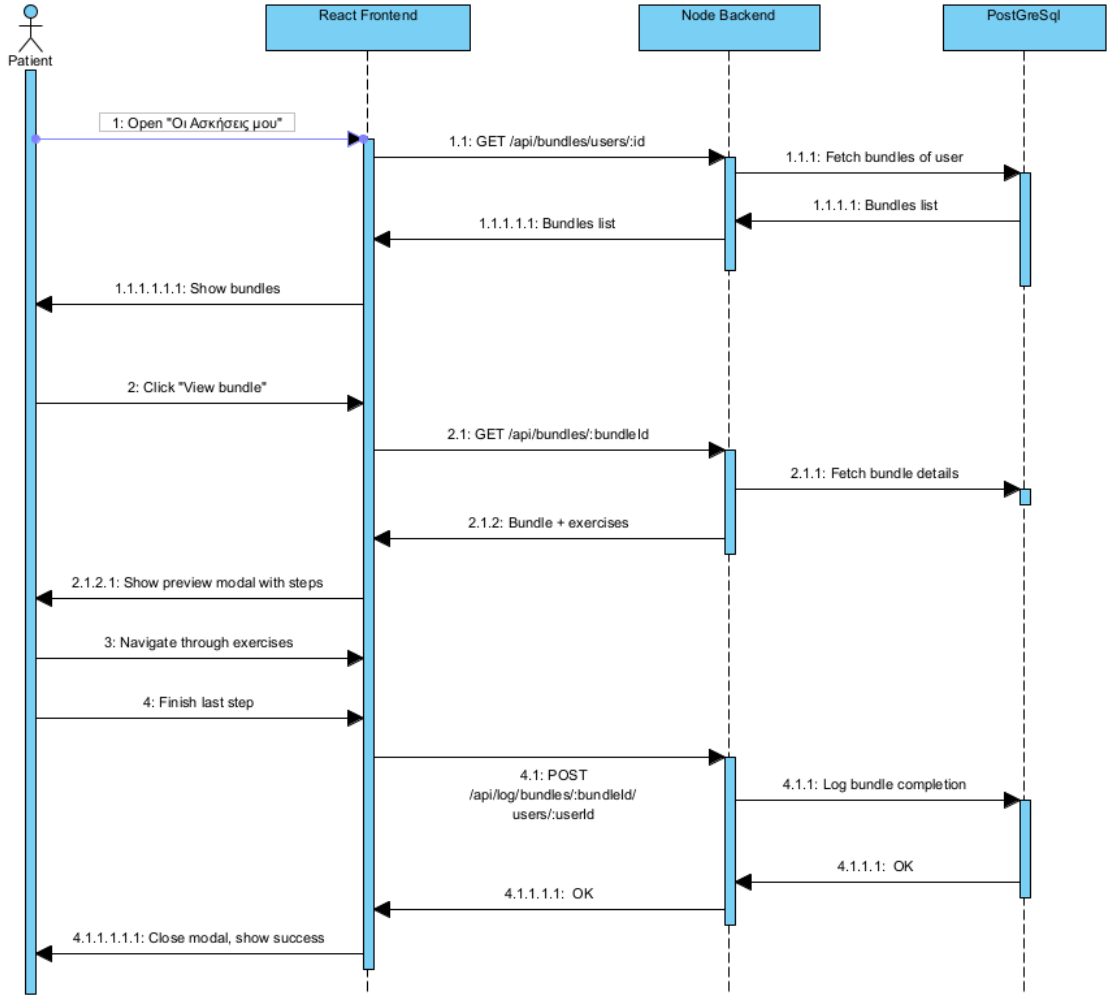
Σχήμα 4.29: Διάγραμμα ακολουθίας κλινικού: προβολή λίστας ασθενών.

Κλινικός: Ανάθεση bundle σε ασθενή. Σε δεύτερο παράδειγμα, ο κλινικός επιλέγει έναν συγκεκριμένο ασθενή και το σύστημα ανακτά τα στοιχεία του μαζί με τα bundles που του έχουν ήδη ανατεθεί. Έπειτα, με την επιλογή «Ανάθεση bundle», αποστέλλεται αίτημα `POST /api/bundles/:bundleId/users/:userId` και η βάση ενημερώνεται για τη νέα ανάθεση. Σε επιτυχία, η λίστα των αναθέσεων ανανεώνεται αυτόματα στο Frontend.



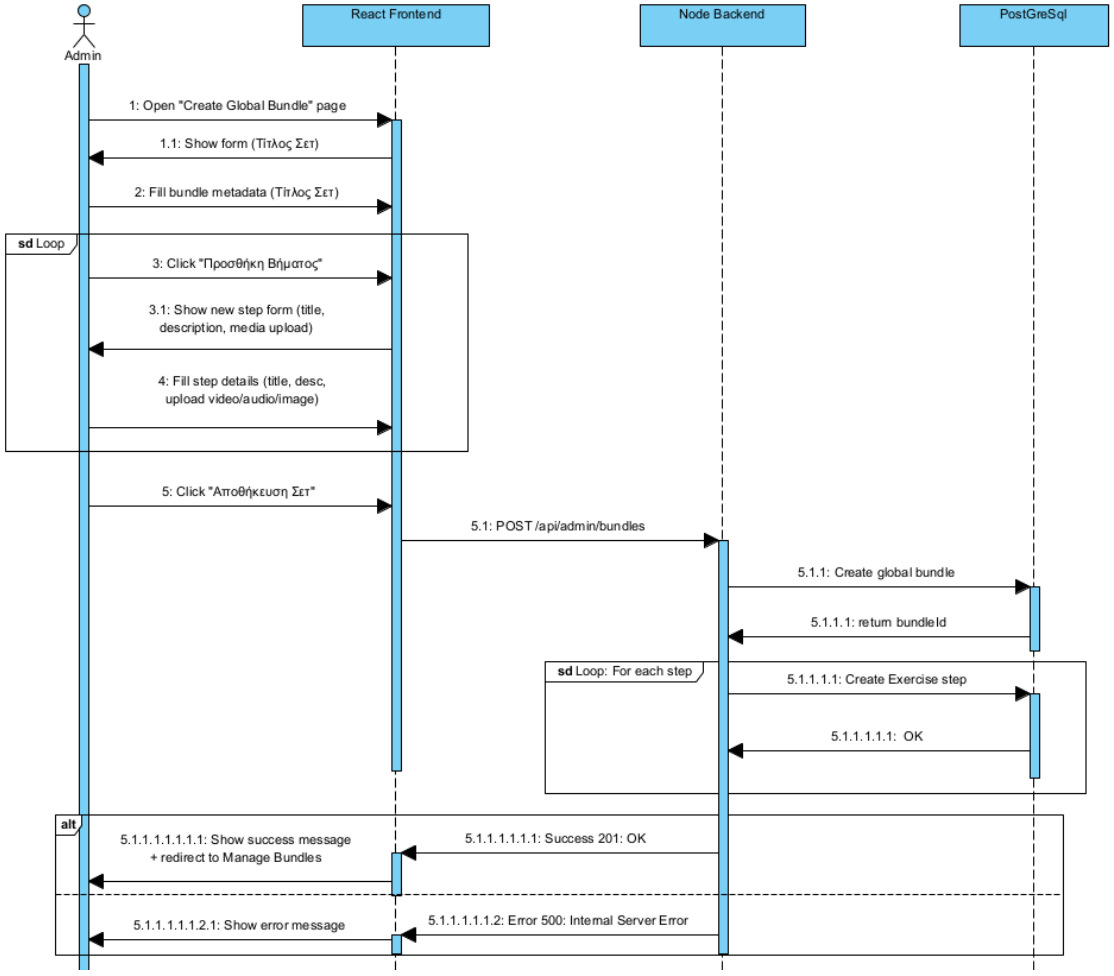
Σχήμα 4.30: Διάγραμμα ακολουθίας κλινικού: ανάθεση bundle σε ασθενή.

Ασθενής: Εκτέλεση άσκησης. Ο ασθενής ανοίγει τη σελίδα «Οι Ασκήσεις μου», από όπου ανακτώνται τα bundles που του έχουν ανατεθεί. Επιλέγοντας ένα συγκεκριμένο bundle, το Frontend καλεί το Backend για να φέρει τα βήματα και τις ασκήσεις που περιέχει. Ο χρήστης πλοηγείται διαδοχικά στα βήματα και με την ολοκλήρωση του τελευταίου αποστέλλεται αίτημα `POST /api/log/bundles/:bundleId/users/:userId`, ώστε να καταγραφεί η πρόοδος στη βάση δεδομένων. Η ροή ολοκληρώνεται με επιβεβαίωση επιτυχίας.



Σχήμα 4.31: Διάγραμμα ακολουθίας ασθενούς: εκτέλεση και ολοκλήρωση bundle ασκήσεων.

Διαχειριστής: Δημιουργία Global Bundle. Τέλος, για τον ρόλο του διαχειριστή παρουσιάζεται η διαδικασία δημιουργίας ενός Global Bundle. Ο διαχειριστής ανοίγει τη σχετική φόρμα, συμπληρώνει τα βασικά μεταδεδομένα (τίτλο, περιγραφή) και προσθέτει δυναμικά βήματα, στα οποία ανεβάζει αρχεία πολυμέσων (βίντεο, ήχο, εικόνες). Με την ολοκλήρωση της φόρμας, το Frontend αποστέλλει αίτημα `POST /api/admin/bundles`, το οποίο δημιουργεί το νέο bundle και εισάγει τα επιμέρους βήματα στη βάση δεδομένων. Η εναλλακτική ροή (alt) αποτυπώνει πιθανό σφάλμα (`500 Internal Server Error`), ενώ στην κανονική πορεία εμφανίζεται μήνυμα επιτυχίας και ο διαχειριστής ανακατευθύνεται στη σελίδα διαχείρισης.



Σχήμα 4.32: Διάγραμμα ακολουθίας διαχειριστή: δημιουργία νέου Global Bundle με δυναμικά βήματα και αρχεία πολυμέσων.

Κεφάλαιο 5

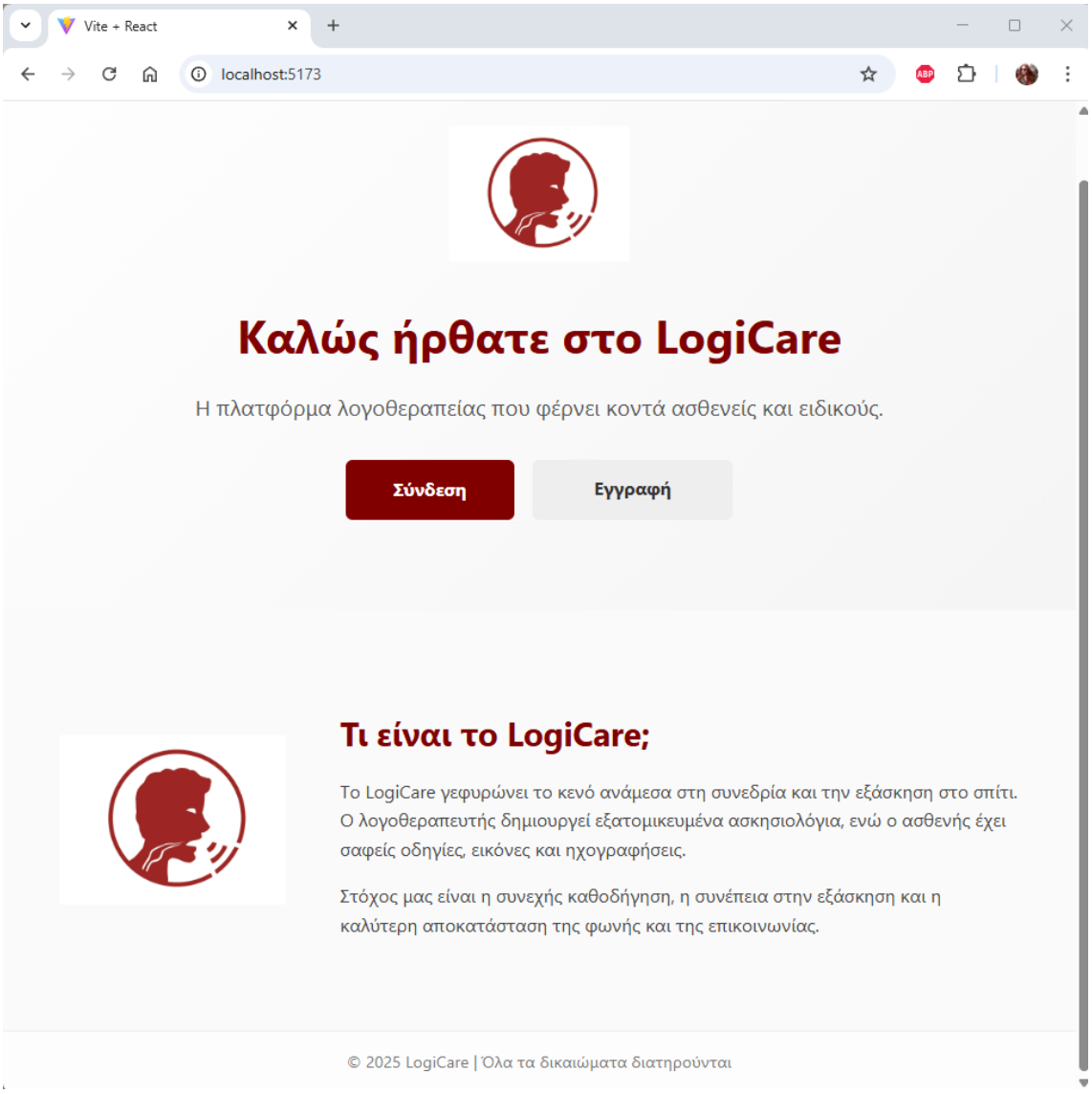
Επίδειξη Frontend

5.1 Στιγμιότυπα Οθόνης (Screenshots)

Στην ενότητα αυτή παρουσιάζονται χαρακτηριστικά στιγμιότυπα από το περιβάλλον χρήστη της εφαρμογής LogiCare, με στόχο την ανάδειξη της λειτουργικότητας και της ευχρηστίας που παρέχει στους διαφορετικούς ρόλους χρηστών (ασθενής, κλινικός, διαχειριστής). Τα στιγμιότυπα που ακολουθούν οργανώνονται με βάση τη ροή αλληλεπίδρασης, ξεκινώντας από τις βασικές οθόνες εισόδου και πλοήγησης, και συνεχίζοντας με πιο εξειδικευμένες οθόνες για κάθε ρόλο.

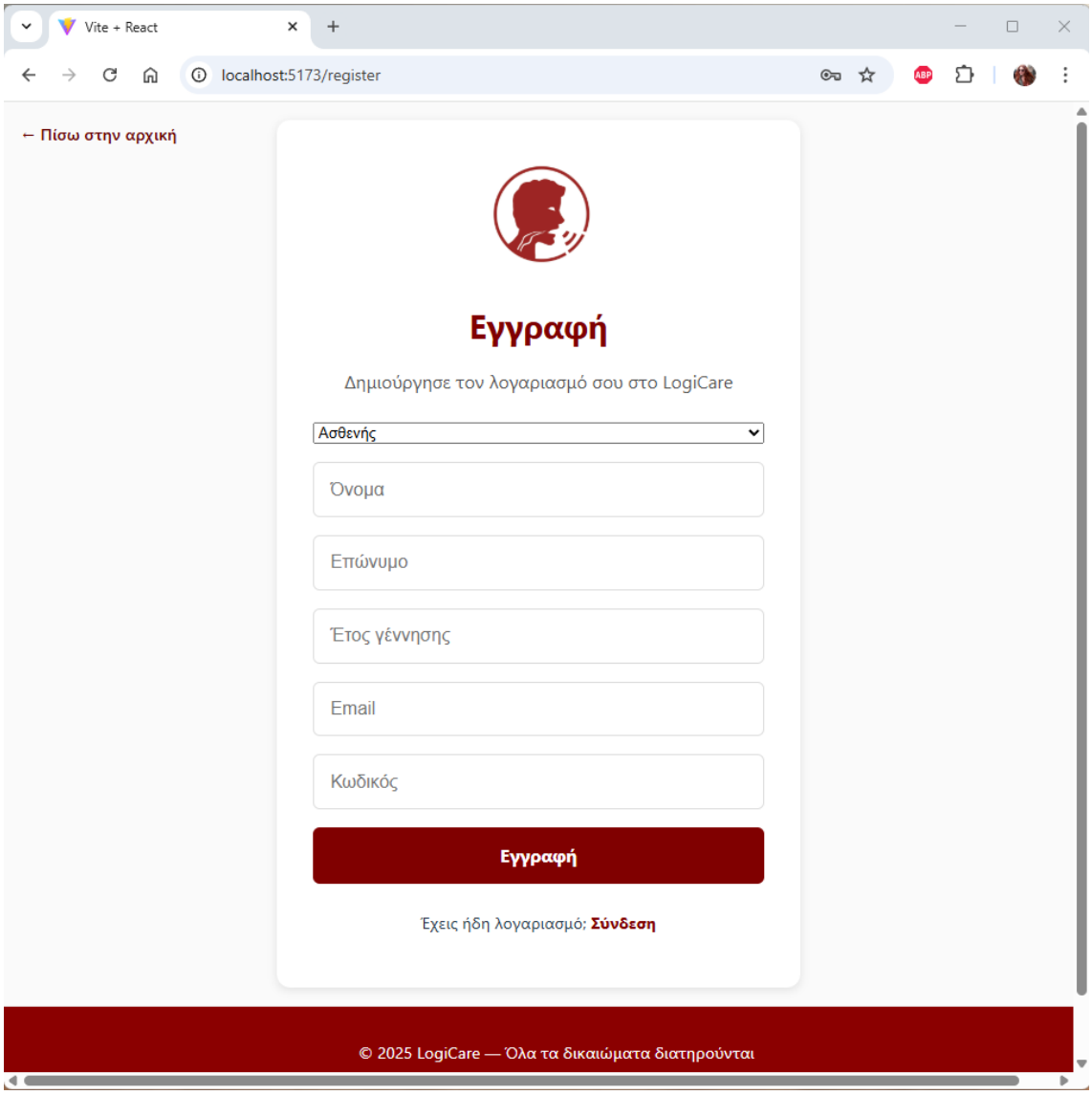
5.1.1 Γενικές Σελίδες

Homepage. Η αρχική σελίδα της εφαρμογής παρουσιάζει συνοπτικά τον σκοπό και τα βασικά οφέλη του LogiCare, προσφέροντας στον επισκέπτη δύο κύριες επιλογές: Σύνδεση για υπάρχοντες χρήστες και Εγγραφή για νέους.



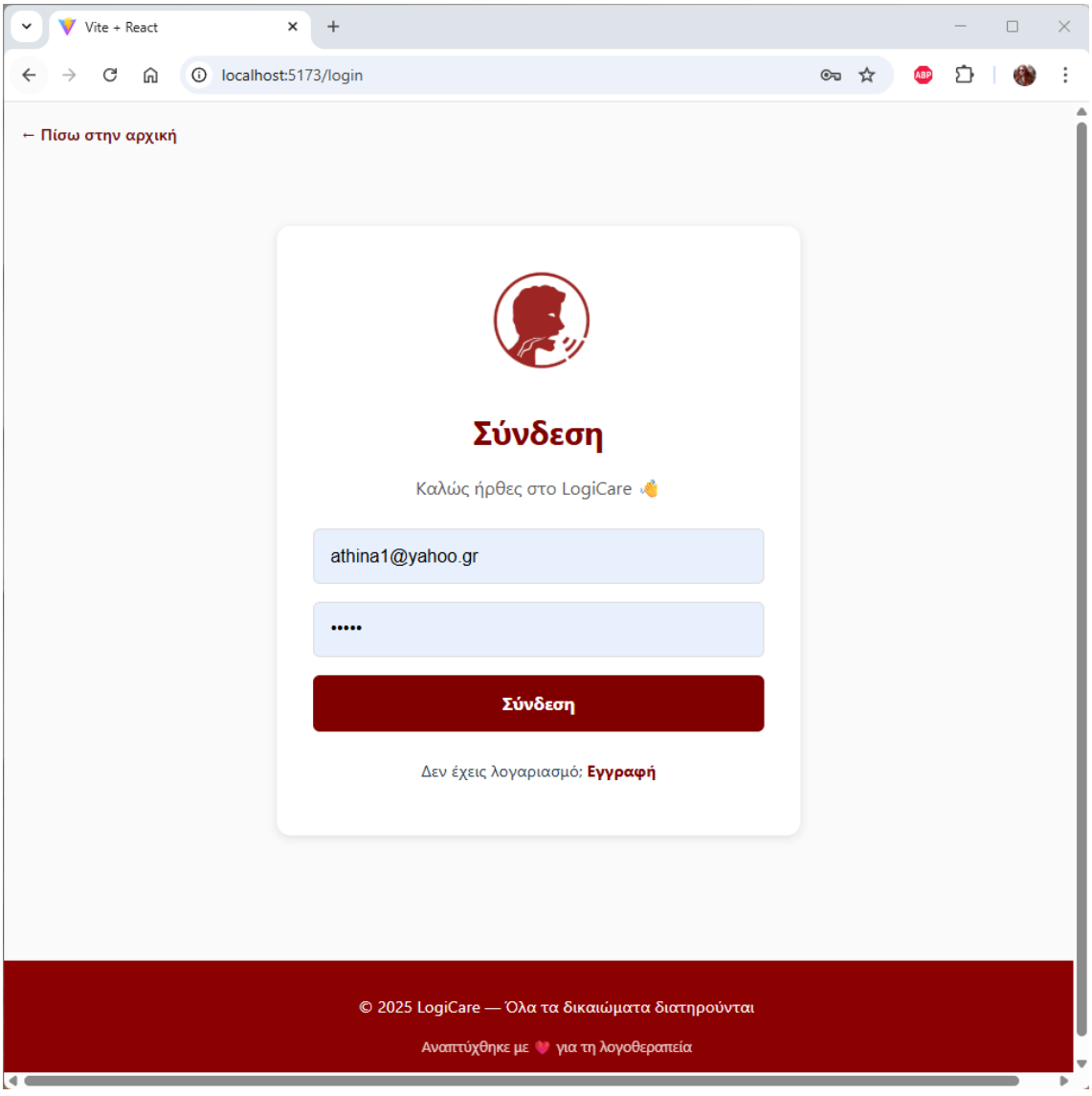
Σχήμα 5.1: (Homepage) της πλατφόρμας LogiCare.

Register. Στη φόρμα εγγραφής (Register), ο νέος χρήστης καλείται να εισάγει βασικές πληροφορίες (ρόλος, όνομα, επώνυμο, έτος γέννησης, email, κωδικό).



Σχήμα 5.2: Φόρμα εγγραφής νέου χρήστη στην πλατφόρμα.

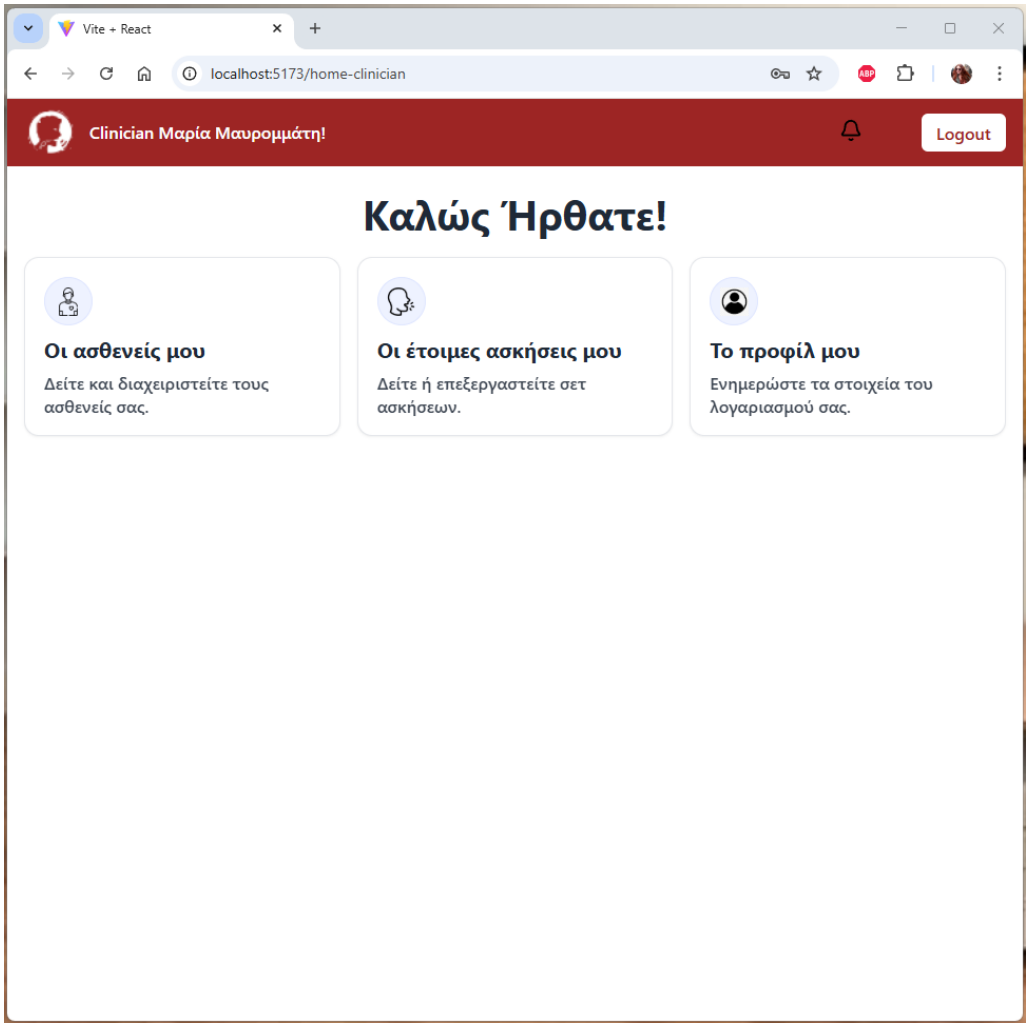
Login. Η σελίδα σύνδεσης (Login) επιτρέπει σε υπάρχοντες χρήστες να εισάγουν το email και τον κωδικό τους για να αποκτήσουν πρόσβαση στο σύστημα.



Σχήμα 5.3: Οθόνη σύνδεσης χρήστη (Login) με δυνατότητα ελέγχου λαθών.

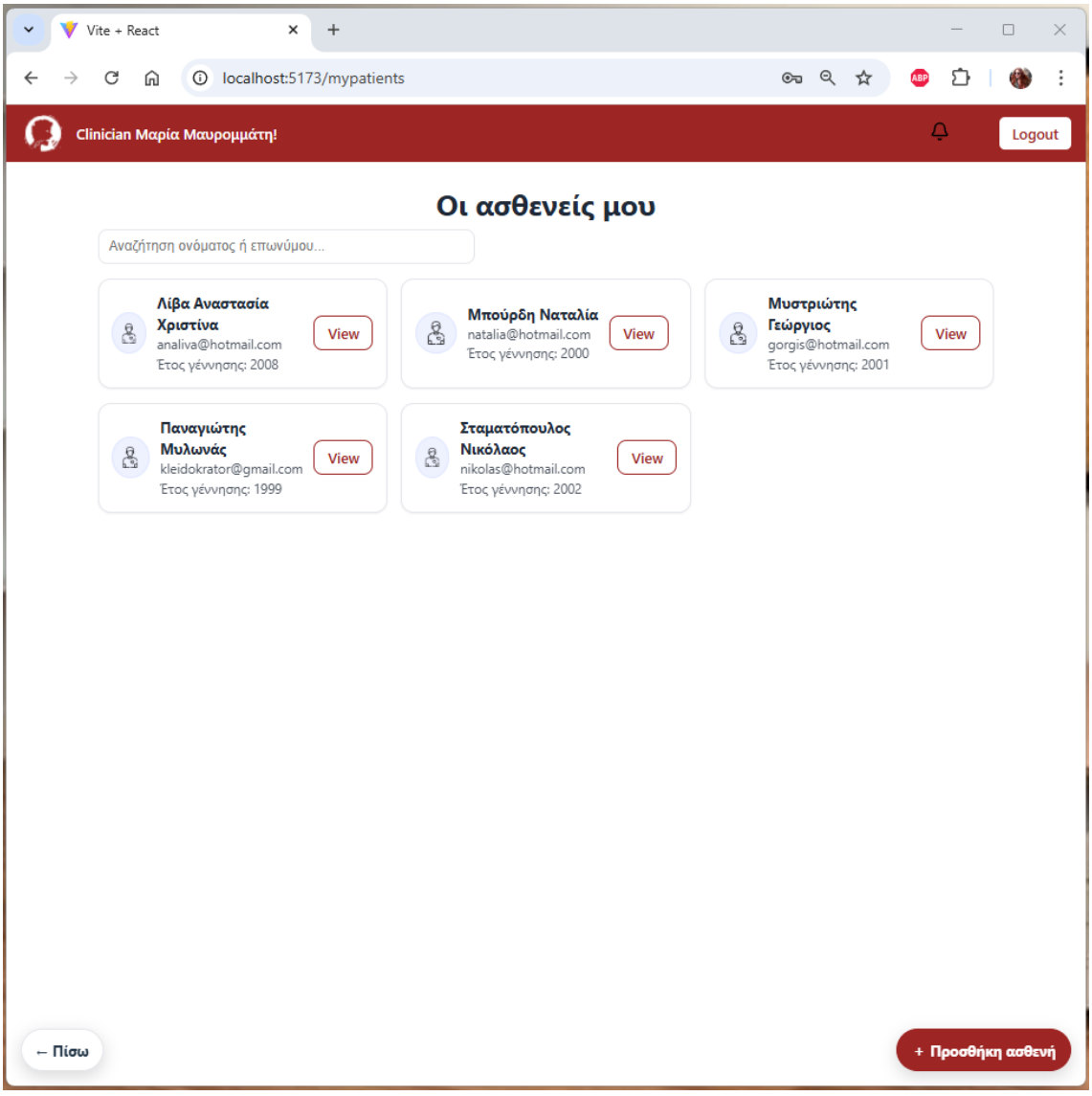
5.1.2 Οθόνες Κλινικού

Αρχική σελίδα. Ο κλινικός μετά τη σύνδεση οδηγείται στην αρχική σελίδα, όπου εμφανίζονται οι βασικές επιλογές: «Οι ασθενείς μου», «Οι έτοιμες ασκήσεις μου» και «Το προφίλ μου».



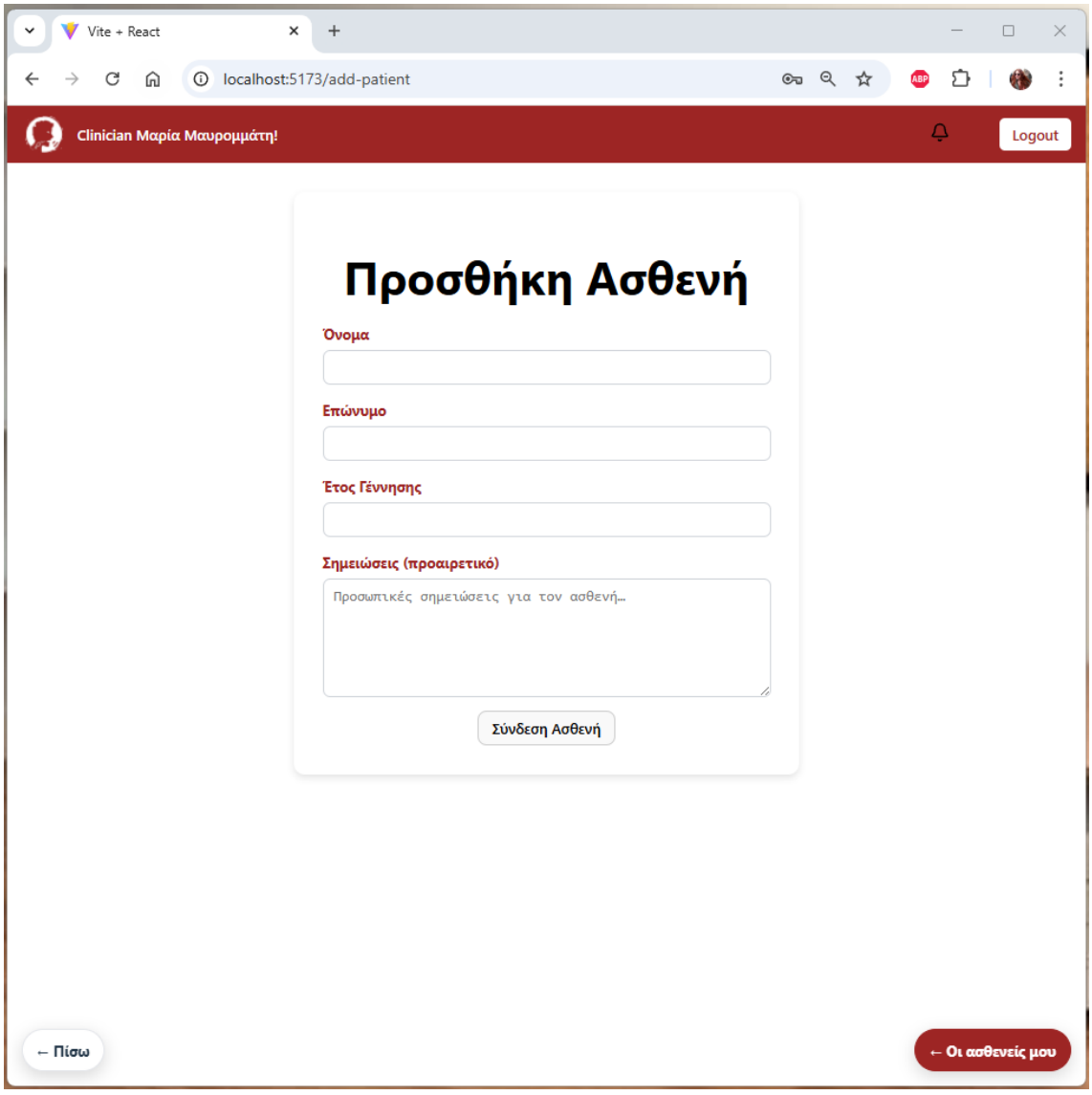
Σχήμα 5.4: Αρχική σελίδα κλινικού.

Οι ασθενείς μου. Στη σελίδα αυτή εμφανίζονται όλοι οι ασθενείς που έχει καταχωρήσει ο κλινικός, με δυνατότητα αναζήτησης ανά όνομα/επώνυμο και επιλογή «Προσθήκη ασθενή».



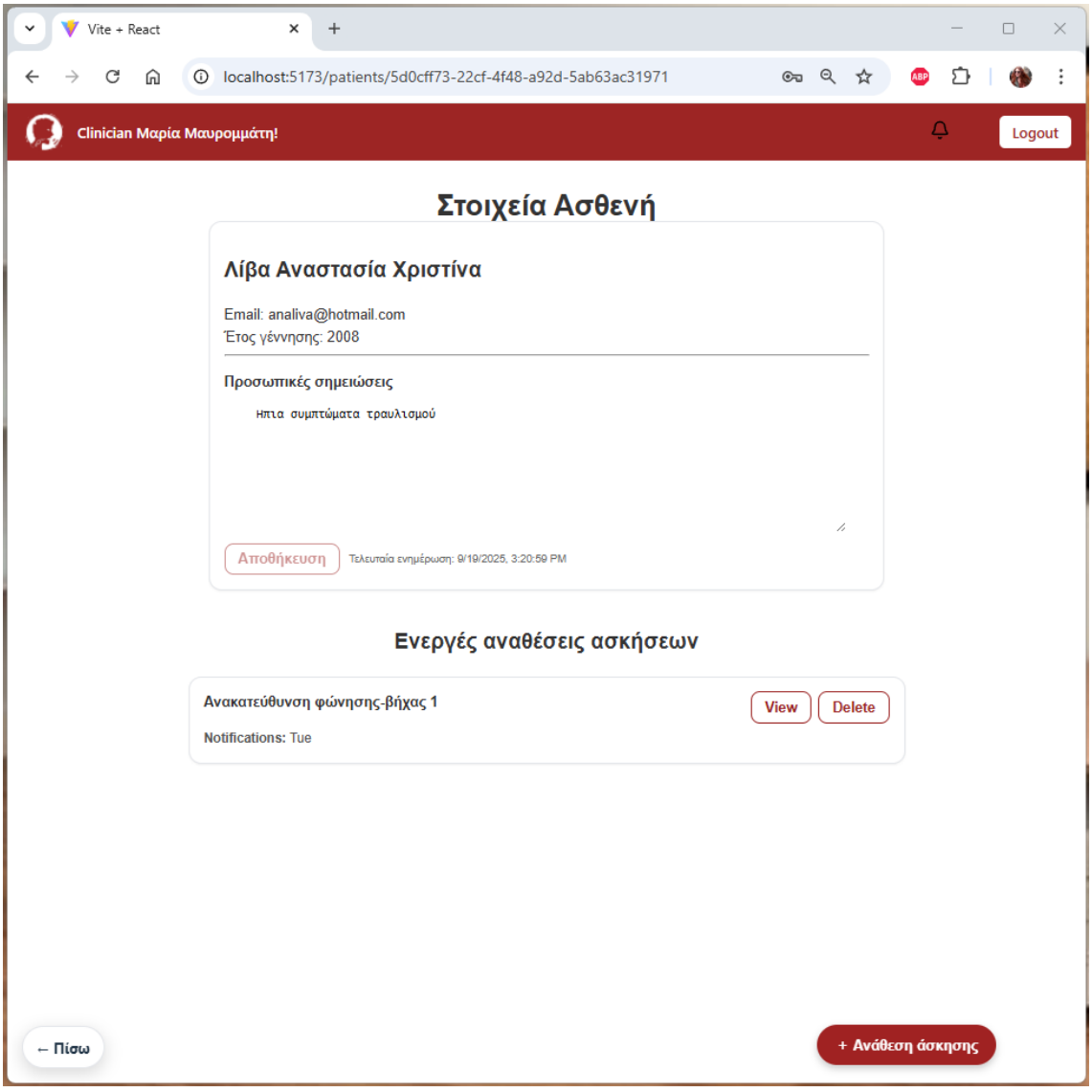
Σχήμα 5.5: Λίστα ασθενών κλινικού.

Προσθήκη ασθενή. Η φόρμα προσθήκης νέου ασθενή περιλαμβάνει πεδία για όνομα, επώνυμο, έτος γέννησης και προαιρετικές προσωπικές σημειώσεις. Με τη σωστή συμπλήρωση των στοιχείων, δημιουργείται αυτόματα η σύνδεση με τον κλινικό που τον καταχώρησε.



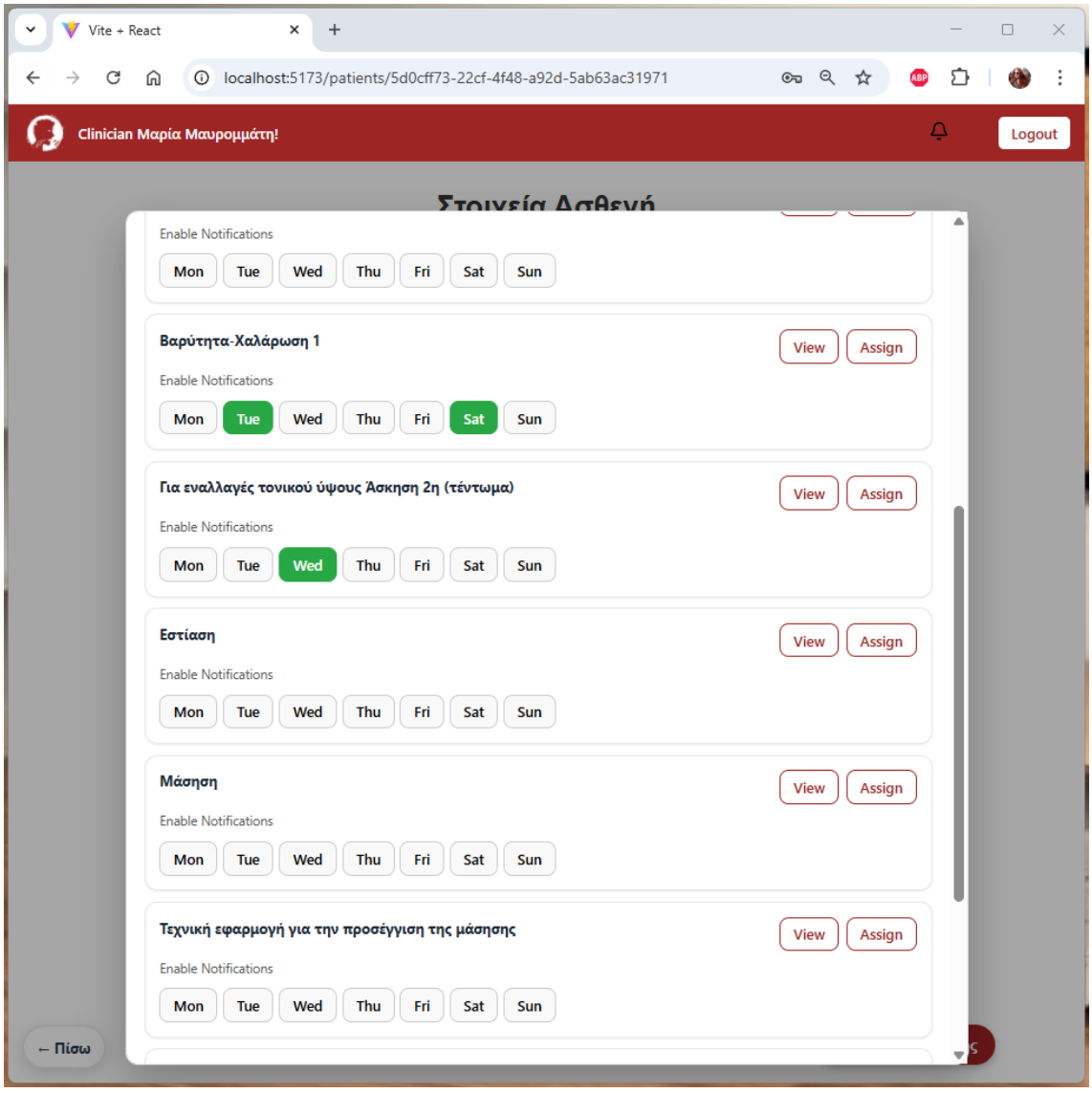
Σχήμα 5.6: Φόρμα προσθήκης νέου ασθενή και δημιουργία σύνδεσης με τον κλινικό.

Στοιχεία ασθενή. Με την επιλογή συγκεκριμένου ασθενή εμφανίζεται η καρτέλα με τα στοιχεία του, τις προσωπικές σημειώσεις του κλινικού και τις ενεργές αναθέσεις ασκήσεων.



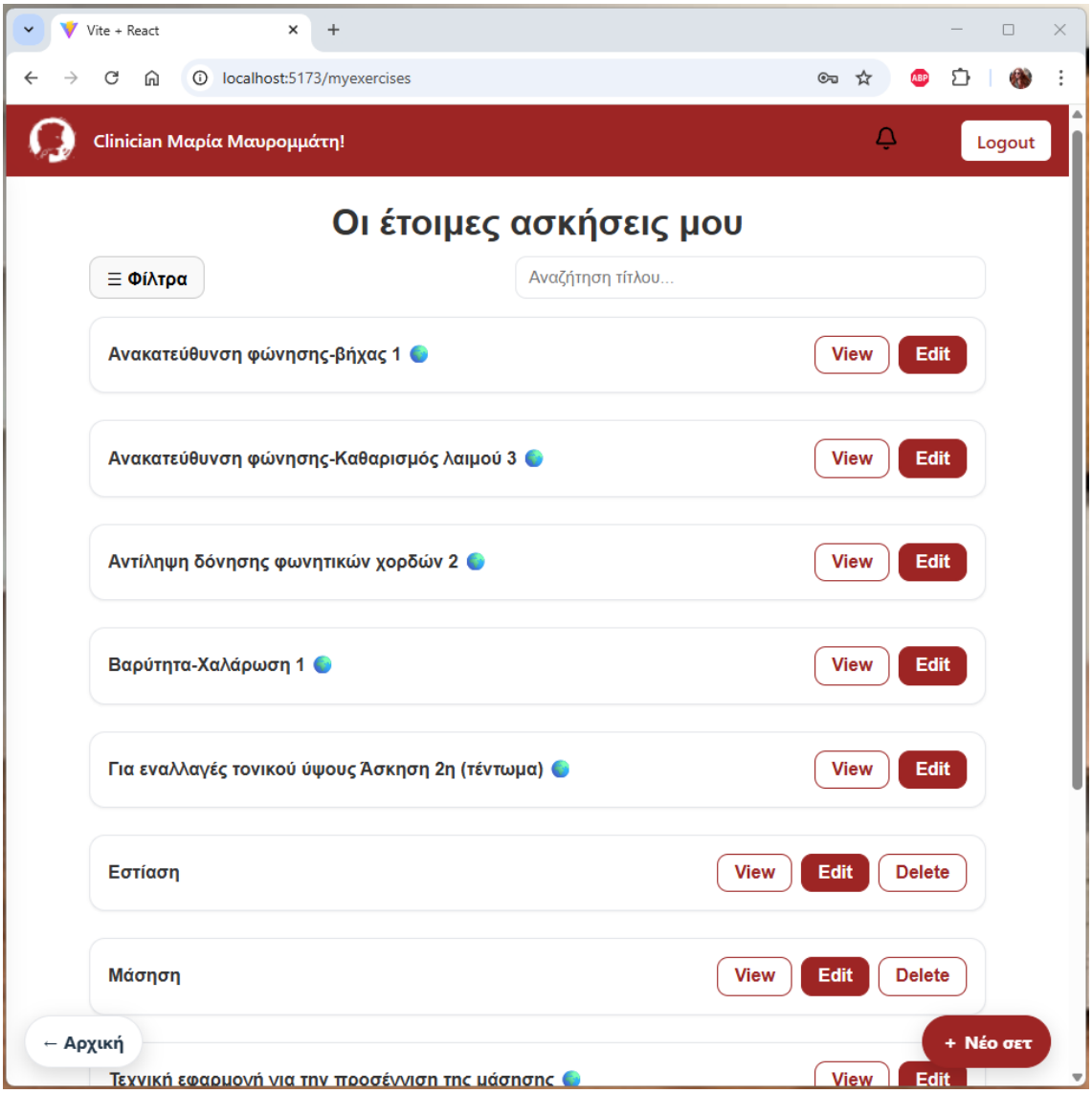
Σχήμα 5.7: Σελίδα προβολής στοιχείων ασθενή και ενεργών αναθέσεων.

Ανάθεση άσκησης. Ο κλινικός μπορεί να εκχωρήσει νέα άσκηση σε ασθενή επιλέγοντας από τα διαθέσιμα σετ, καθώς και να ορίσει μέρες ειδοποιήσεων. Οι νέες αναθέσεις εμφανίζονται στην καρτέλα του ασθενή.



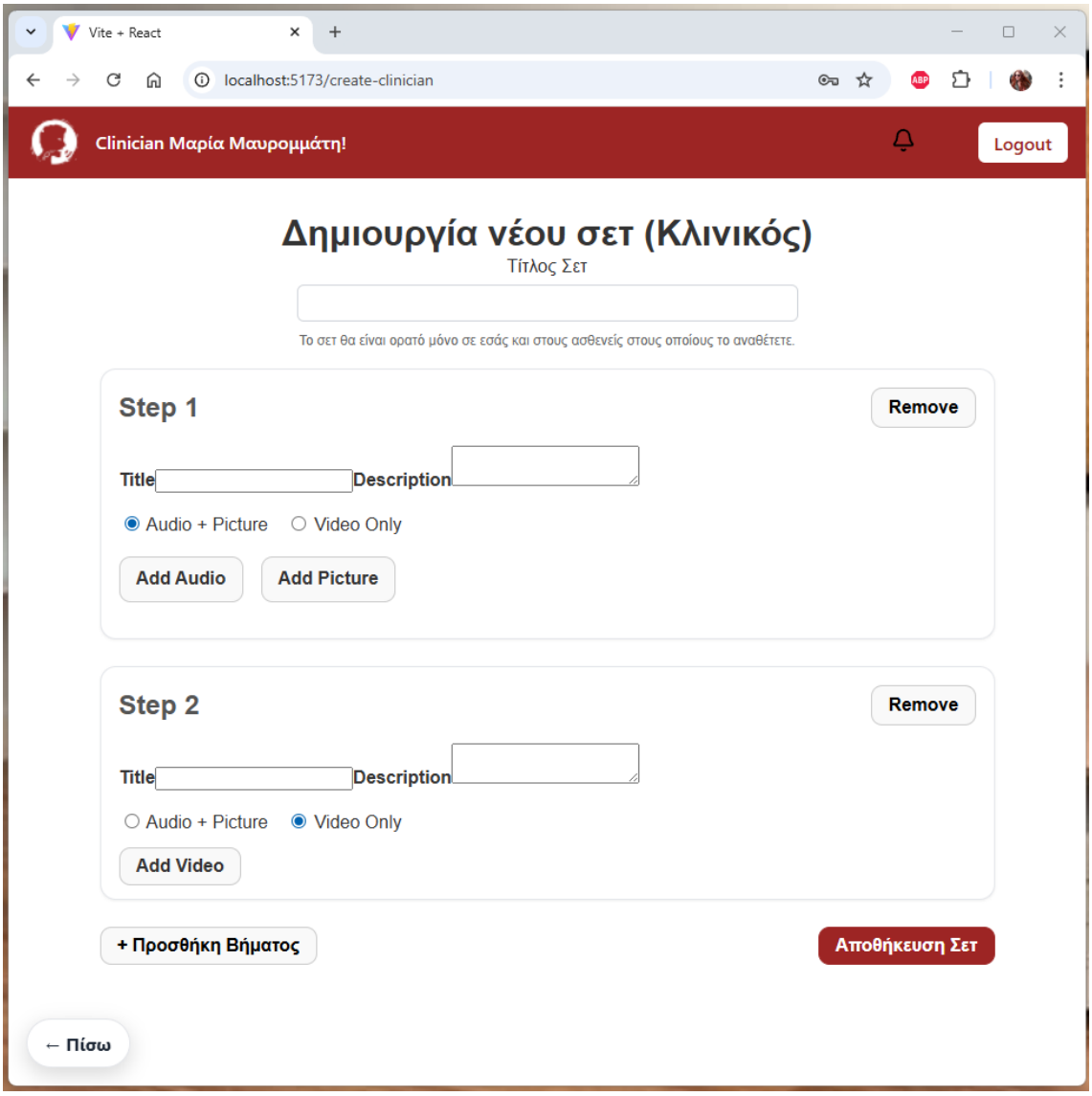
Σχήμα 5.8: Ανάθεση νέας άσκησης σε ασθενή με επιλογή ημερών ειδοποίησης.

Οι έτοιμες ασκήσεις μου. Ο κλινικός έχει πρόσβαση σε ολόκληρο το ασκησεολόγιο της εφαρμογής, καθώς και σε όλες τις ασκήσεις που έχει δημιουργήσει ο ίδιος. Από τη σελίδα αυτή μπορεί να προβάλλει, να επεξεργαστεί ή να διαγράψει ασκήσεις, ενώ παρέχεται επίσης η δυνατότητα δημιουργίας νέου σετ.



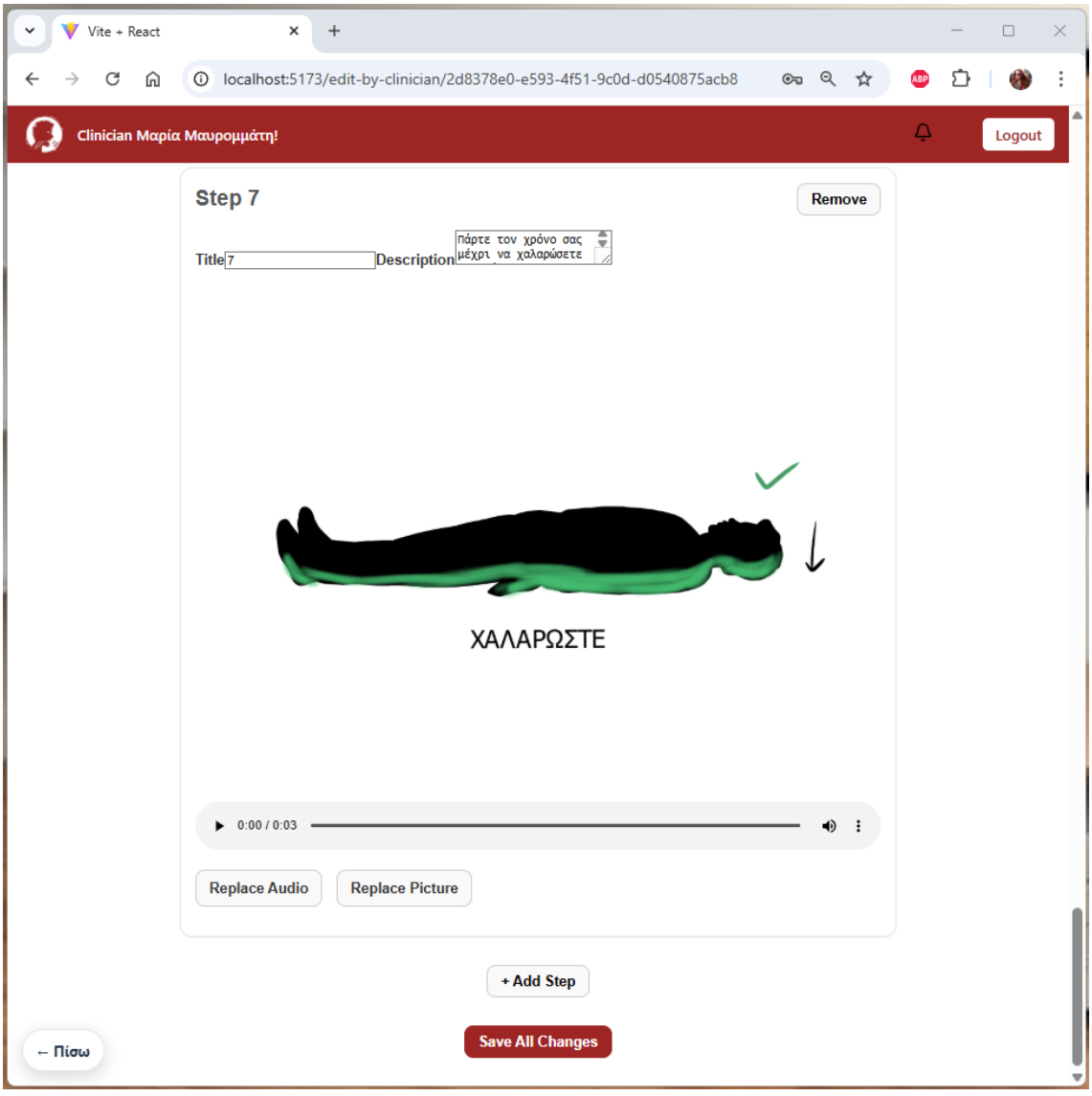
Σχήμα 5.9: Σελίδα με το συνολικό ασκησεολόγιο και τις προσωπικές δημιουργίες του κλινικού.

Δημιουργία νέου σετ. Η φόρμα δημιουργίας επιτρέπει την εισαγωγή τίτλου και περιγραφής, καθώς και την προσθήκη δυναμικών βημάτων με αρχεία πολυμέσων (ήχος, εικόνα, βίντεο).



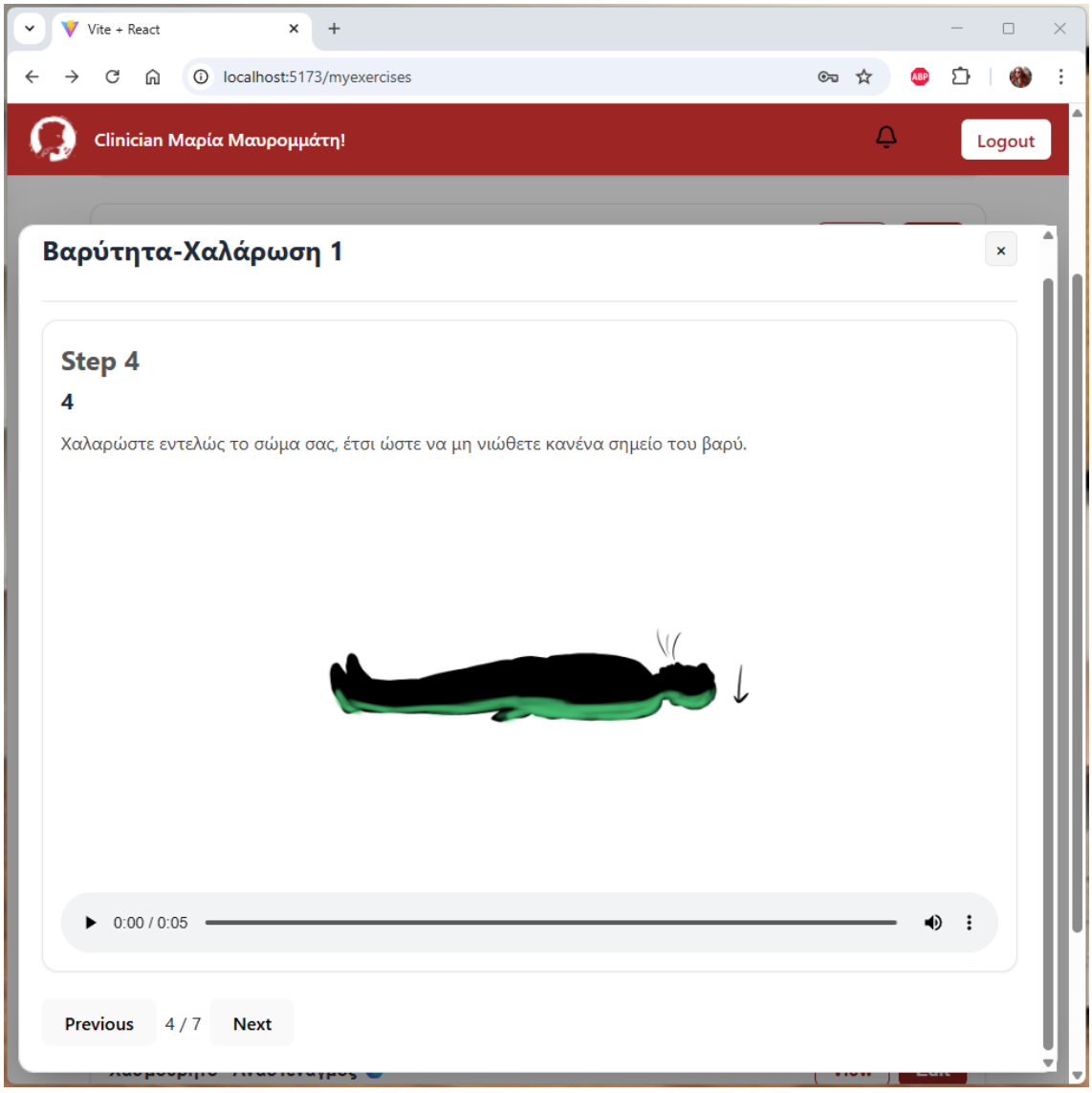
Σχήμα 5.10: Δημιουργία νέου σετ ασκήσεων με δυναμικά βήματα.

Επεξεργασία σετ. Ο κλινικός μπορεί να επεξεργαστεί ένα υπάρχον σετ, τροποποιώντας τίτλους, περιγραφές και αρχεία, καθώς και να προσθέσει ή αφαιρέσει βήματα.



Σχήμα 5.11: Επεξεργασία σετ ασκήσεων.

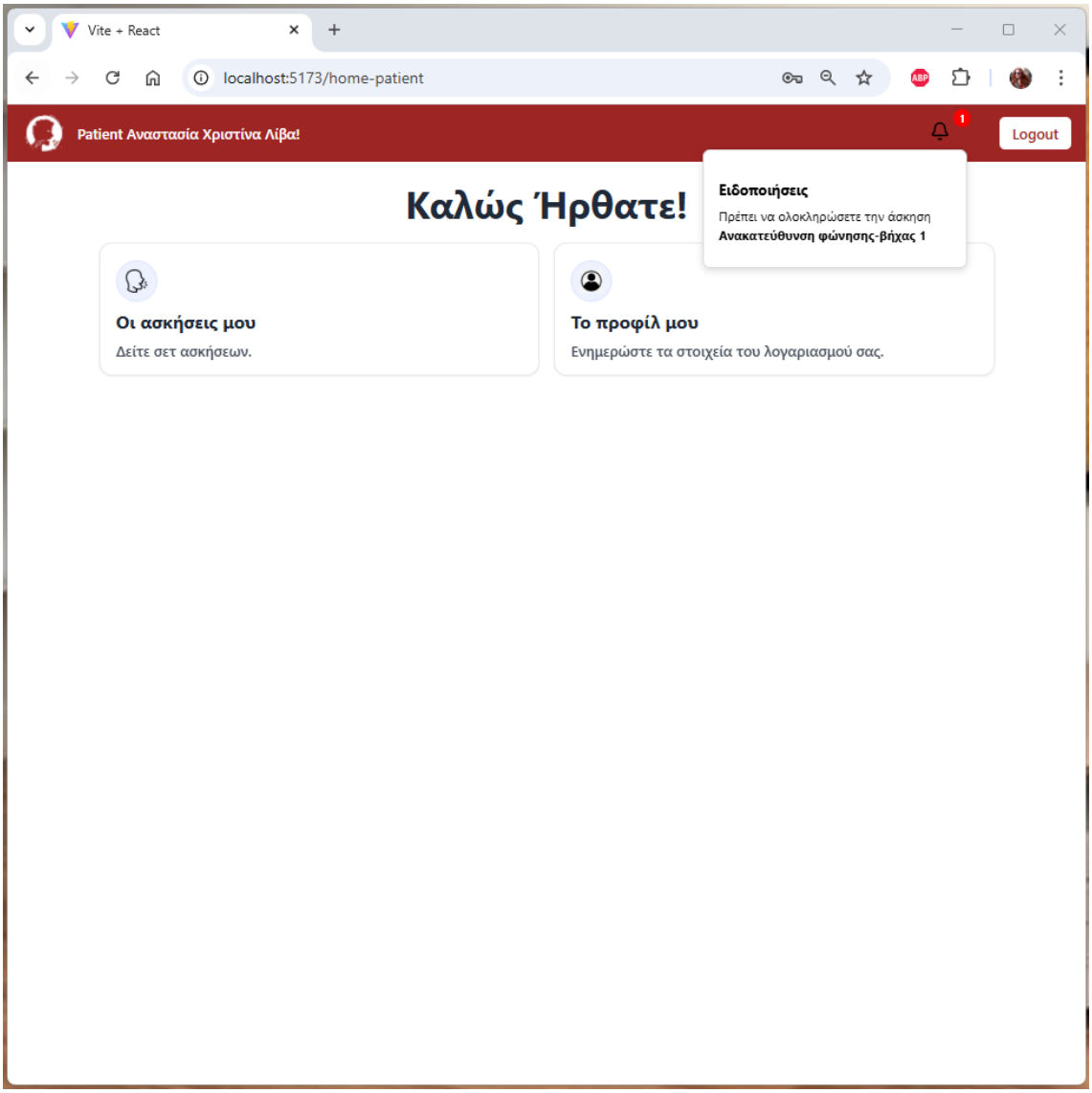
Προβολή σετ. Η λειτουργία προβολής επιτρέπει στον κλινικό να δει ένα σετ βήμα-βήμα, με τον ίδιο τρόπο που θα το εκτελέσει ο ασθενής, περιλαμβάνοντας πολυμέσα.



Σχήμα 5.12: Προβολή σετ ασκήσεων από την πλευρά του κλινικού.

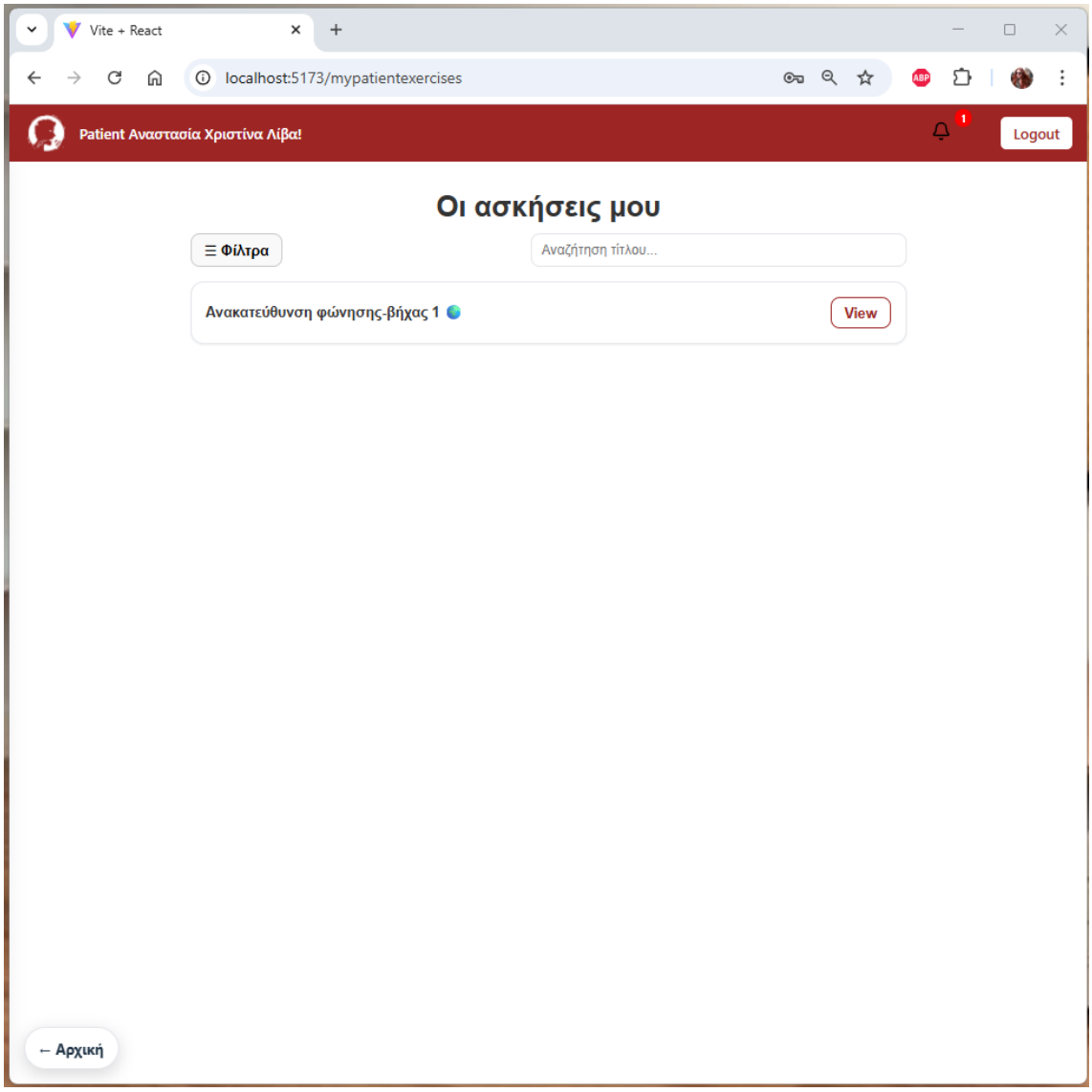
5.1.3 Οθόνες Ασθενή

Homepage Patient. Ο ασθενής βλέπει τις βασικές επιλογές: οι ασκήσεις του και το προφίλ του. Εμφανίζονται επίσης ειδοποιήσεις για εκκρεμείς ασκήσεις.



Σχήμα 5.13: Αρχική σελίδα ασθενή με ειδοποίηση άσκησης.

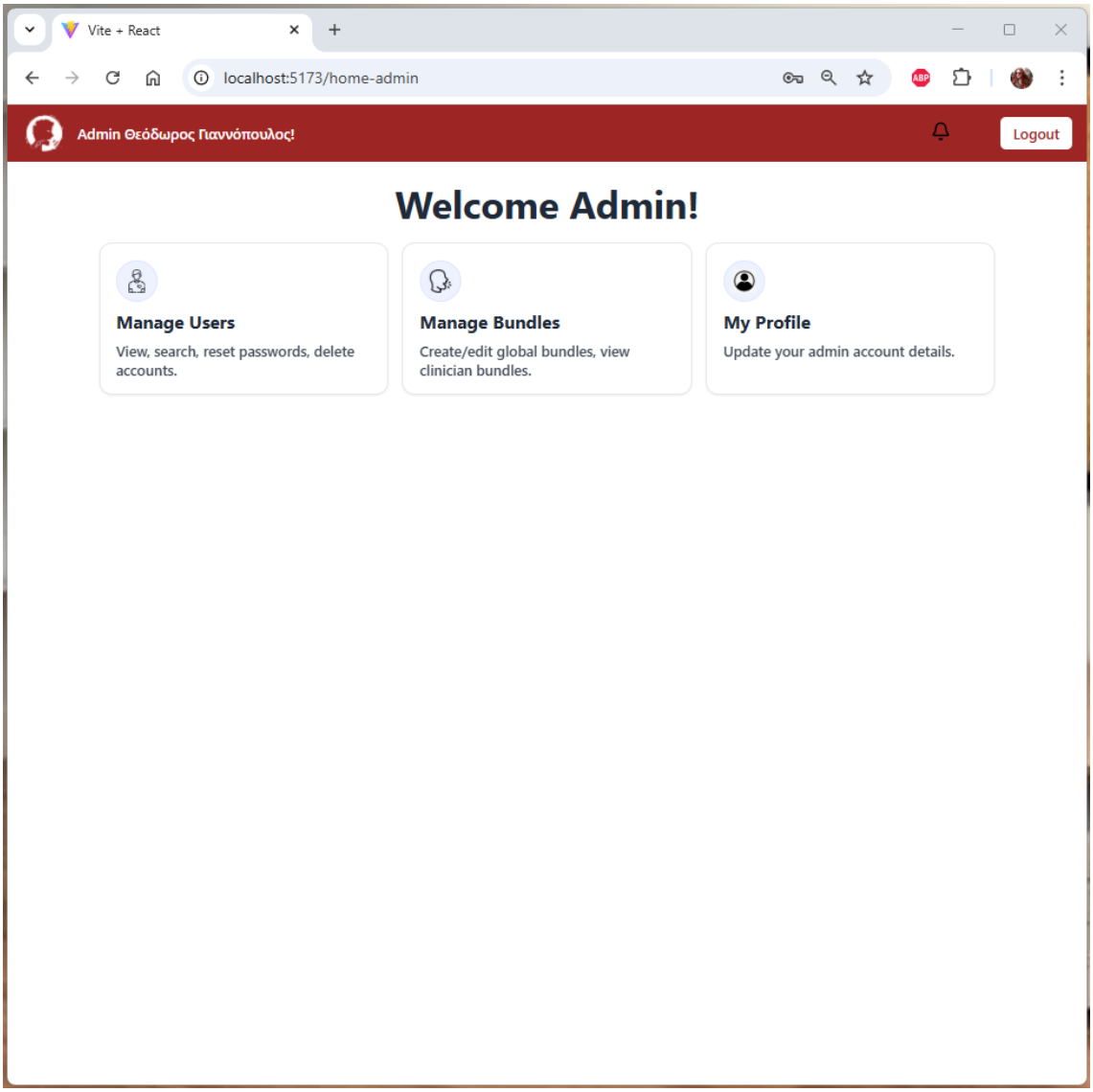
My Exercises. Ο ασθενής έχει πρόσβαση στη λίστα των ασκήσεων που του έχουν ανατεθεί από τον χλινικό.



Σχήμα 5.14: Σελίδα «Οι ασκήσεις μου» για τον ασθενή.

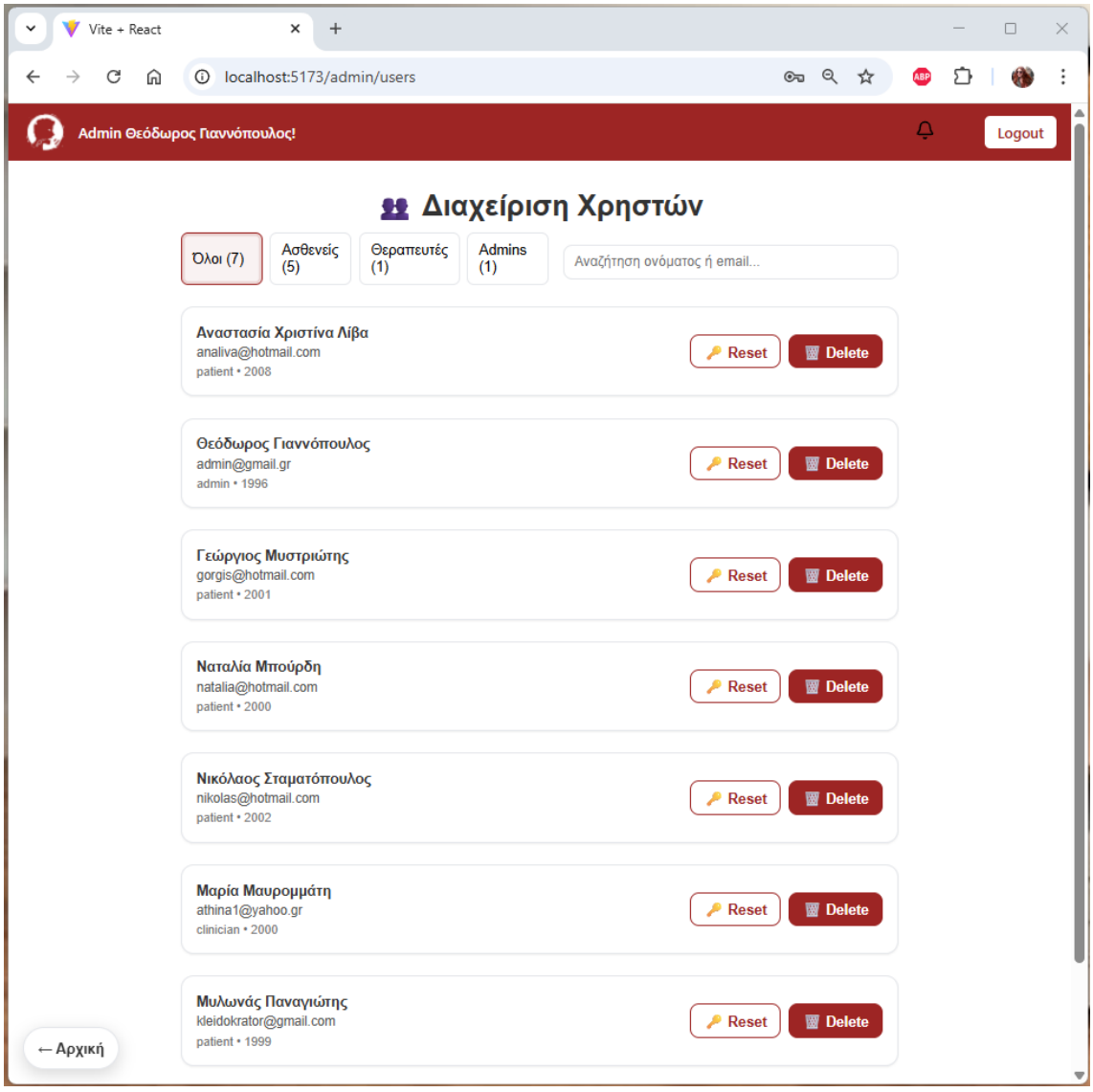
5.1.4 Οθόνες Διαχειριστή

Home Page Admin. Η αρχική σελίδα του διαχειριστή παρέχει πρόσβαση στη διαχείριση χρηστών, bundles και στο προφίλ του.



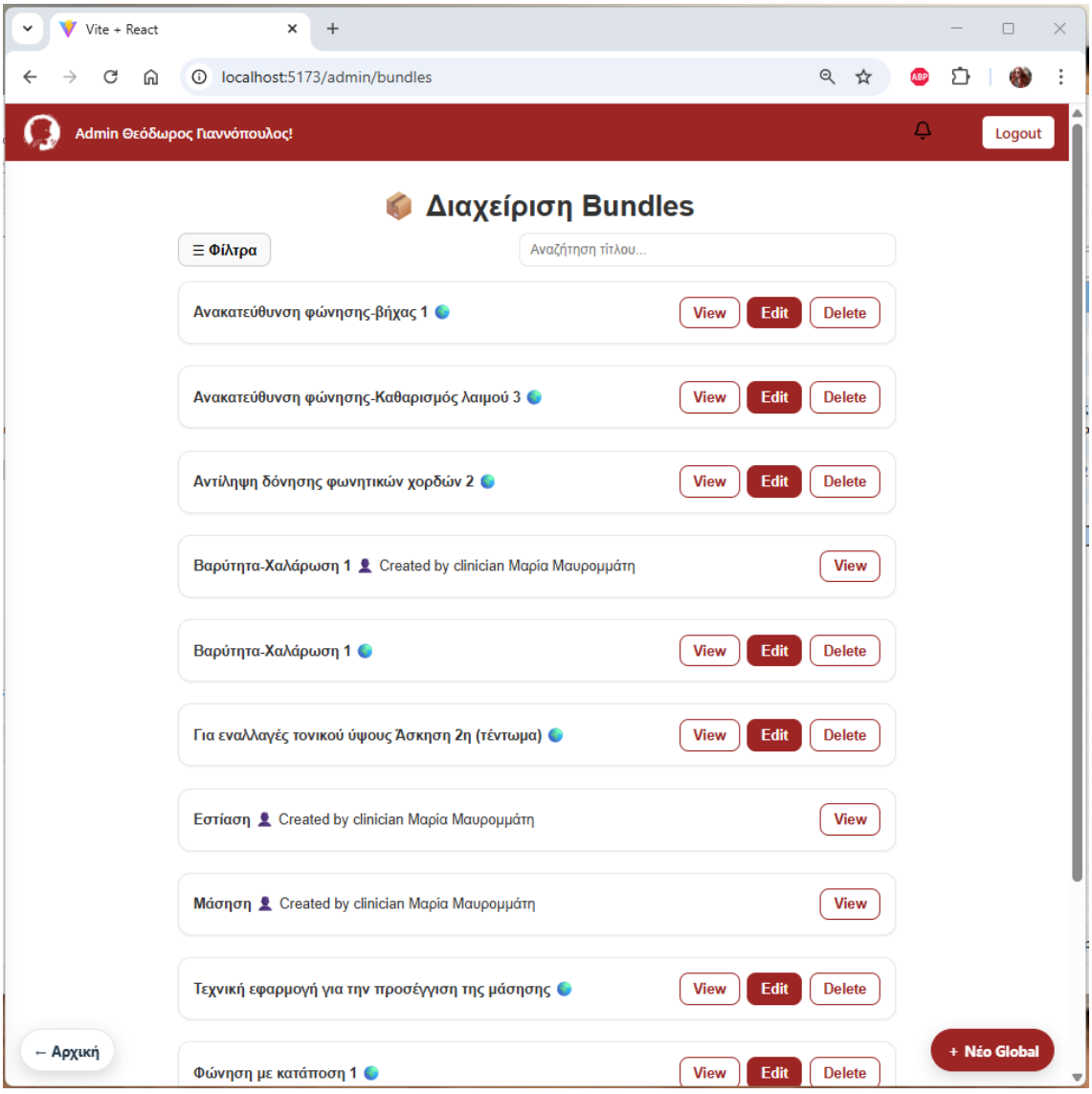
Σχήμα 5.15: Αρχική σελίδα διαχειριστή.

Manage Users. Ο διαχειριστής μπορεί να δει όλους τους χρήστες, να επαναφέρει κωδικούς ή να διαγράψει λογαριασμούς.



Σχήμα 5.16: Διαχείριση χρηστών από τον διαχειριστή.

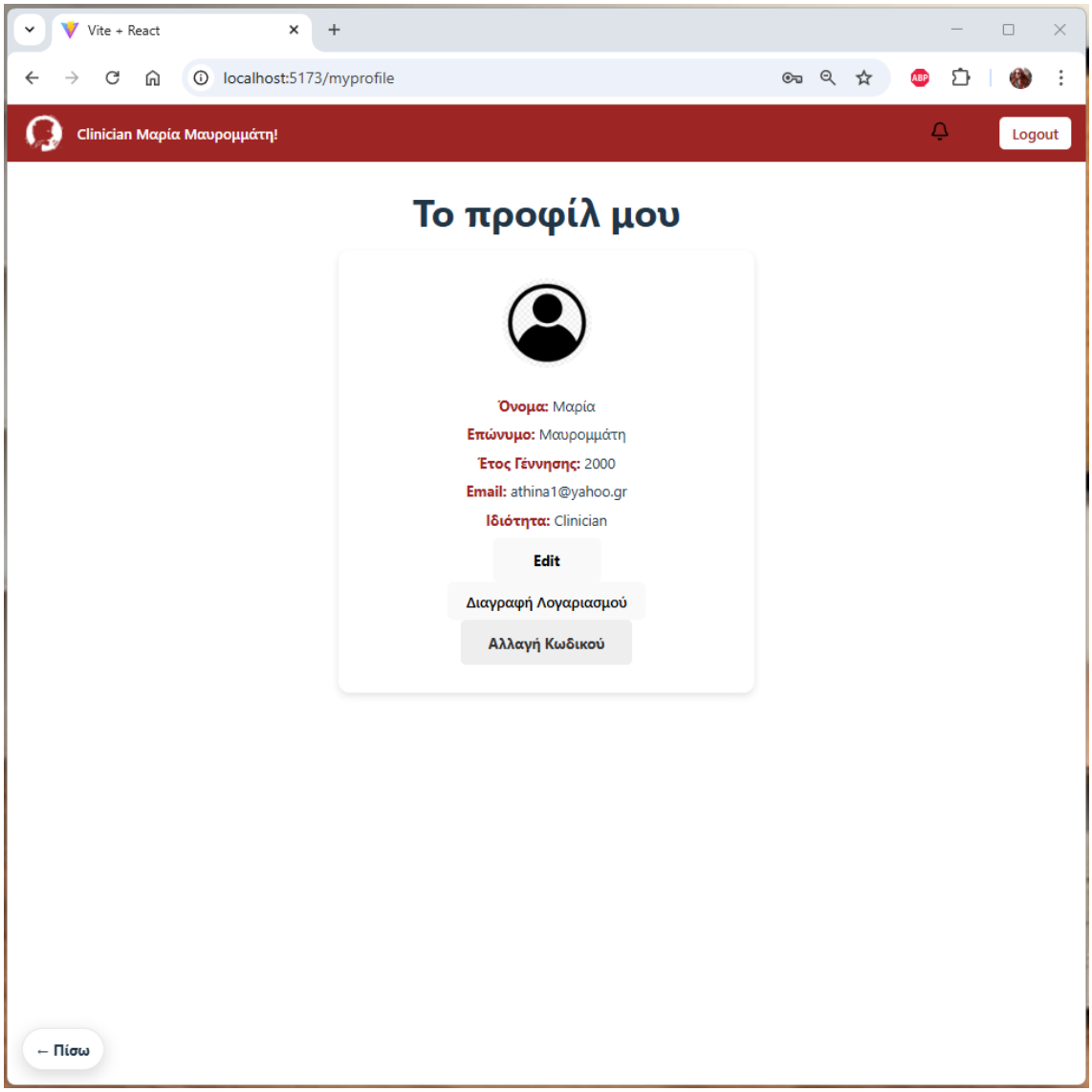
Manage Bundles. Εδώ εμφανίζονται όλα τα διαθέσιμα bundles με δυνατότητα προβολής, επεξεργασίας ή διαγραφής.



Σχήμα 5.17: Διαχείριση bundles από τον διαχειριστή.

5.1.5 Διαχείριση Προφίλ

Edit Profile. Όλοι οι χρήστες (ανεξαρτήτως ρόλου) μπορούν να επεξεργαστούν το προφίλ τους μέσω αντίστοιχης φόρμας.



Σχήμα 5.18: Επεξεργασία στοιχείων προφίλ.

Κεφάλαιο 6

Επίλογος

6.1 Συμπεράσματα

Η παρούσα διπλωματική εργασία είχε ως στόχο την ανάπτυξη μιας διαδικτυακής εφαρμογής λογοθεραπείας, η οποία επιτρέπει την εξατομικευμένη ανάθεση ασκήσεων και την ευέλικτη διαχείριση ασθενών από τον λογοθεραπευτή. Η υλοποιημένη πλατφόρμα αποδεικνύει πως η αξιοποίηση σύγχρονων τεχνολογιών μπορεί να καλύψει το διαπιστωμένο κενό που υπάρχει στη χρήση ψηφιακών εργαλείων στον τομέα της λογοθεραπείας, προσφέροντας ένα φιλικό και πρακτικό περιβάλλον τόσο στους επαγγελματίες υγείας όσο και στους ασθενείς.

Η εφαρμογή επέτρεψε στους κλινικούς να οργανώνουν το ασκησιολόγιο, να δημιουργούν νέες ασκήσεις, να αναθέτουν προσωποποιημένα προγράμματα και να παρακολουθούν την πρόοδο κάθε ασθενή. Από την πλευρά του, ο ασθενής έχει πλέον τη δυνατότητα να λαμβάνει σαφείς οδηγίες, να εκτελεί τις ασκήσεις με οπτικοακουστική υποστήριξη και να συμμετέχει πιο ενεργά στη θεραπευτική διαδικασία, ακόμη και εξ αποστάσεως.

Συνολικά, η εργασία ανέδειξε ότι η ενσωμάτωση ψηφιακών λύσεων στη λογοθεραπεία μπορεί να βελτιώσει την ποιότητα της παρεχόμενης φροντίδας και να συμβάλλει στη συνεχή υποστήριξη του ασθενή εκτός των κλασικών συνεδριών.

6.2 Μελλοντικές Επεκτάσεις

Η περαιτέρω ανάπτυξη της εφαρμογής μπορεί να επικεντρωθεί στον εμπλουτισμό του ασκησιολογίου με νέους τύπους ασκήσεων, όπως δραστηριότητες που αξιοποιούν τη χρήση μικροφώνου, αυτόματη ανάλυση φωνής και διαδραστικές λειτουργίες βασισμένες σε τεχνολογίες αναγνώρισης ομιλίας. Τέτοιες δυνατότητες θα επιτρέψουν στους ασθενείς να λαμβάνουν άμεση και εξατομικευμένη ανατροφοδότηση για την εκτέλεση των ασκήσεών τους, ενισχύοντας τη συμμετοχή και το αίσθημα προόδου.

Παράλληλα, η δημιουργία ενός συστήματος ανταλλαγής φεεδβαςκ, όπου οι λογοθεραπευτές θα μπορούν να προτείνουν ή να αξιολογούν νέες λειτουργίες και ασκήσεις, αναμένεται να συμβάλει στη διαρκή βελτίωση και προσαρμογή της πλατφόρμας στις πραγματικές ανάγκες της κοινότητας.

Επιπλέον, ιδιαίτερης σημασίας είναι η προσθήκη αναλυτικού ιστορικού, ώστε ο κλινικός να έχει πρόσβαση σε παλαιότερες σημειώσεις και αναθέσεις για κάθε ασθενή, διευκολύνοντας έτσι την παρακολούθηση της προόδου και τη λήψη τεκμηριωμένων θεραπευτικών αποφάσεων. Οι παραπάνω κατευθύνσεις, σε συνδυασμό με τη συνεχή συλλογή ανατροφοδότησης από τους ίδιους τους χρήστες, θα διασφαλίσουν τη δυναμική εξέλιξη της εφαρμογής και τη μέγιστη αξιοποίησή της στην πράξη.

Σε επίπεδο λογισμικού, σε περίπτωση σημαντικής αύξησης της ζήτησης και του αριθμού των χρηστών, η εφαρμογή θα μπορούσε να μεταβεί από μονολιθική αρχιτεκτονική σε αρχιτεκτονική μικροϋπηρεσιών (microservices), διασφαλίζοντας έτσι μεγαλύτερη επεκτασιμότητα, ευκολία συντήρησης και ανθεκτικότητα. Επιπλέον, η ενσωμάτωση εναλλακτικών τρόπων ταυτοποίησης και εγγραφής χρηστών, όπως το Google Authentication και άλλες μέθοδοι θα προσφέρουν μεγαλύτερη ευκολία και ασφάλεια κατά τη διαδικασία σύνδεσης. Τέλος, η ανάπτυξη mobile εφαρμογής θα δώσει ευκολότερη πρόσβαση από φορητές συσκευές, καλύπτοντας ακόμη πιο αποτελεσματικά τις ανάγκες των χρηστών σε κάθε περιβάλλον.

Βιβλιογραφία

[1] GeeksForGeeks. «Difference between http and https». Accessed: 2025-09-03. (2024), address: https://www.geeksforgeeks.org/computer-networks/difference-between-http-and-https-2/.

[2] MDN Web Docs (Mozilla). «Http request methods». (2025), address: https://developer.mozilla.org/en-US/docs/Web/HTTP/Reference/Methods (visited on 09/05/2025).

[3] Bunny.net Academy. «What is http (hypertext transfer protocol)?». Accessed: 2025-09-03. (2024), address: https://bunny.net/academy/http/what-is-http-hypertext-transfer-protocol/.

[4] MDN Web Docs. «Http cookies». Accessed: 2025-09-03. (2025), address: https://developer.mozilla.org/en-US/docs/Web/HTTP/Guides/Cookies.

[5] Amazon Web Services. «What is a restful api?». Accessed: 2025-09-04. (2025), address: https://aws.amazon.com/what-is/restful-api/.

[6] M. Papazoglou, *Web Services: Principles and Technology*. Pearson Education, 2008.

[7] D. Lalias, «Δικτυακή εφαρμογή για την οπτικοποίηση εξαρτήσεων μεταξύ endpoints μιας διεπαφής rest», Διπλωματική εργασία, Εθνικό Μετσόβιο Πολυτεχνείο, Αθήνα, 2024.

[8] RestfulAPI.net. «What is rest — representational state transfer?». Accessed: 2025-09-04. (2024), address: https://restfulapi.net/.

[9] Bunny.net Academy. «What is representational state transfer (rest)?». Accessed: 2025-09-04. (2024), address: https://bunny.net/academy/http/what-is-representational-state-transfer-rest/.

[10] GeeksForGeeks. «Node.js rest api introduction». Accessed: 2025-09-04. (2024), address: https://www.geeksforgeeks.org/node-js/rest-api-introduction/.

[11] Βεσκούκης, *Στοιχεία τεχνολογίας λογισμικού*, Greek. Κάλλιπος, Ανοικτές Ακαδημαϊκές Εκδόσεις, 2015, Προπτυχιακό εγχειρίδιο, ISBN: 978-960-603-060-4. DOI: 10.57713/kallipos-720. address: https://dx.doi.org/10.57713/kallipos-720.

[12] PostgreSQL Global Development Group. «Postgresql documentation». (2025), address: https://www.postgresql.org/docs/.

[13] Node.js Foundation. «Node.js documentation». (2025), address: https://nodejs.org/en/docs.

[14] Express.js Team. «Express.js documentation». (2025), address: https://expressjs.com/en/4x/api.html.

[15] Passport.js. «Passport.js documentation». (2025), address: https://www.passportjs.org/docs/.

[16] bcrypt.js. «Bcrypt documentation». (2025), address: https://www.npmjs.com/package/bcrypt.

[17] Multer. «Multer documentation». (2025), address: https://github.com/expressjs/multer.

[18] Meta Platforms, Inc. «React – official documentation». (2025), address: https://react.dev/.
[19] React Router Team. «React router documentation». (2025), address: https://reactrouter.com/en/main.
[20] React Community. «React context api documentation». (2025), address: https://react.dev/reference/react/Context.
[21] Docker Inc. «Docker documentation». (2025), address: https://docs.docker.com/.