ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ – ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Πρόγραμμα Μεταπτυχιακών Σπουδών

«..... Πληροφορική.....»

Μεταπτυχιακή Διατριβή

Τίτλος Διατριβής	**Σχεδίαση και Υλοποίηση Πολυγλωσσικής Διαδικτυακής Πλατφόρμας Ιατρικής Πληροφόρησης με Εξατομικευμένες Προτάσεις** **Design and Implementation of a Personalized Multilingual Medical Information Web Platform**
Ονοματεπώνυμο Φοιτητή	**Συμεωνίδης Διονύσιος**
Πατρώνυμο	**Χριστόφορος**
Αριθμός Μητρώου	**ΜΠΠΛ2329**
Επιβλέπων	**Δρ. Αλέπης Ευθύμιος, Καθηγητής**

Ημερομηνία Παράδοσης **ΜΑΙΟΣ 2026**

Τριμελής Εξεταστική Επιτροπή

Ευθύμιος Αλέπης Καθηγητής	Μαρία Βίρβου Καθηγήτρια	Διονύσιος Σωτηρόπουλος Αν. Καθηγητής

Ευχαριστίες

Θα ήθελα να εκφράσω τις ειλικρινείς μου ευχαριστίες σε όσους συνέβαλαν στην ολοκλήρωση της παρούσας μεταπτυχιακής διπλωματικής εργασίας. Πρωτίστως, οφείλω ένα μεγάλο ευχαριστώ στον επιβλέποντα καθηγητή μου, κ.**Αλέπη Ευθύμιο**, για την πολύτιμη καθοδήγηση, την εμπιστοσύνη και τη συνεχή υποστήριξή του σε όλα τα στάδια εκπόνησης της έρευνάς μου. Επίσης, θα ήθελα να ευχαριστήσω θερμά το σύνολο του διδακτικού προσωπικού του μεταπτυχιακού προγράμματος. Οι γνώσεις και τα ακαδημαϊκά εφόδια που μου προσέφεραν κατά τη διάρκεια των σπουδών μου αποτέλεσαν στέρεη βάση για την πορεία μου. Τέλος, ένα από καρδιάς ευχαριστώ ανήκει στην οικογένειά μου και στους φίλους μου, για την αστείρευτη υπομονή, την κατανόηση και την αδιάκοπη ψυχολογική στήριξη σε όλη αυτή τη διαδρομή.

Περίληψη

Η ραγδαία ανάπτυξη του διαδικτυακού περιεχομένου υγείας έχει βελτιώσει την πρόσβαση του κοινού σε ιατρικές πληροφορίες, αλλά ταυτόχρονα έχει αυξήσει τον κίνδυνο παραπληροφόρησης, μειωμένης χρηστικότητας και μη εξατομικευμένης καθοδήγησης. Η παρούσα διπλωματική εργασία παρουσιάζει τη σχεδίαση και υλοποίηση της διαδικτυακής πλατφόρμας ιατρικής πληροφόρησης που συνδυάζει δομημένη αναζήτηση φαρμακευτικών πληροφοριών με λειτουργικότητα εξατομικευμένων προτάσεων βάσει προφίλ, σε ένα ασφαλές και πολυγλωσσικό περιβάλλον.

Το σύστημα αναπτύσσεται με πολυεπίπεδη αρχιτεκτονική, η οποία διαχωρίζει την παρουσίαση, τη λογική εφαρμογής και την πρόσβαση στα δεδομένα. Στο υποσύστημα backend χρησιμοποιείται το πλαίσιο Java Spring Boot για την υλοποίηση των ελεγκτών (controllers), των υπηρεσιών (services) και των αποθετηρίων (repositories), ενώ το frontend αποδίδεται μέσω server-side rendered προτύπων για ευέλικτες και συντηρήσιμες ροές αλληλεπίδρασης. Η πλατφόρμα περιλαμβάνει διαχείριση προφίλ χρήστη, αναζήτηση φαρμάκων και σελίδες λεπτομερειών, εξατομικευμένες προτάσεις, καθώς και βασικούς ελέγχους προσανατολισμένους στην ασφάλεια, όπως προειδοποιήσεις ειδικές για κάθε χρήστη. Η ασφάλεια υλοποιείται μέσω μηχανισμών αυθεντικοποίησης και προστασίας κωδικών πρόσβασης, ενώ η τοπικοποίηση υποστηρίζεται μέσω πόρων γλώσσας στα Αγγλικά και στα Ελληνικά.

Λέξεις-κλειδιά: ιατρική πληροφορική, διαδικτυακή εφαρμογή, εξατομικευμένες προτάσεις υγείας, Spring Boot, συστήματα ιατρικής πληροφόρησης

Abstract

The rapid growth of online health content has improved public access to medical information, but it has also increased the risk of misinformation, poor usability, and non-personalized guidance. This thesis presents the design and implementation this web-based medical information platform that combines structured drug information retrieval with profile-aware recommendation functionality in a secure and multilingual environment.

The system is developed using a layered architecture that separates presentation, business logic, and data access concerns. On the backend, a Java Spring Boot framework is used to implement controllers, services, and repositories, while the frontend is delivered through server-side rendered templates for responsive and maintainable interaction flows. The platform includes user profile management, drug search and detail pages, personalized recommendations, and basic safety-oriented checks such as user-specific warnings. Security is implemented through authentication and password protection mechanisms, and localization is supported through English and Greek language resources.

Keywords: medical informatics, web application, personalized recommendations, Spring Boot, health information systems

Εισαγωγή

Ο ψηφιακός μετασχηματισμός στον χώρο της υγείας έχει αλλάξει σημαντικά τον τρόπο με τον οποίο οι άνθρωποι αναζητούν και καταναλώνουν ιατρική πληροφορία. Οι ασθενείς δεν βασίζονται πλέον αποκλειστικά σε δια ζώσης ιατρικές επισκέψεις για πληροφορίες πρώτου επιπέδου· αντίθετα, συχνά ξεκινούν από διαδικτυακές πλατφόρμες για να κατανοήσουν συμπτώματα, θεραπευτικές επιλογές, χρήση φαρμάκων και πρακτικές πρόληψης. Παρότι αυτή η μετάβαση έχει βελτιώσει την προσβασιμότητα, έχει ταυτόχρονα δημιουργήσει σημαντικές προκλήσεις. Η διαδικτυακή πληροφορία υγείας είναι συχνά κατακερματισμένη, δύσκολα ερμηνεύσιμη από μη ειδικούς και όχι πάντα προσαρμοσμένη στο προφίλ, στο ιατρικό υπόβαθρο ή στις γλωσσικές προτιμήσεις κάθε χρήστη. Ως αποτέλεσμα, οι χρήστες δυσκολεύονται να εντοπίσουν αξιόπιστο και συναφές περιεχόμενο, ιδιαίτερα όταν οι αποφάσεις αφορούν φαρμακευτική αγωγή και πιθανούς κινδύνους ασφάλειας.

Σε αυτό το πλαίσιο, τα εξατομικευμένα και δομημένα συστήματα ιατρικής πληροφόρησης αποκτούν ολοένα μεγαλύτερη αξία. Μια πλατφόρμα που συνδυάζει αξιόπιστη οργάνωση δεδομένων, σαφή παρουσίαση και λογική προτάσεων προσαρμοσμένη στον χρήστη μπορεί να βελτιώσει τόσο τη χρηστικότητα όσο και την πρακτική υποστήριξη λήψης αποφάσεων. Παράλληλα, μια τέτοια πλατφόρμα οφείλει να ικανοποιεί βασικές μηχανικές απαιτήσεις: ασφαλή διαχείριση χρηστών, συντηρήσιμη αρχιτεκτονική, κλιμακώσιμη διαχείριση δεδομένων και υποστήριξη πολυγλωσσικής αλληλεπίδρασης. Οι περιορισμοί αυτοί καθιστούν την ανάπτυξη λογισμικού προσανατολισμένου στην υγεία όχι μόνο πρόκληση πεδίου εφαρμογής, αλλά και πρόκληση μηχανικής λογισμικού.

Η παρούσα διπλωματική εργασία απαντά σε αυτή την ανάγκη μέσω της σχεδίασης και υλοποίησης μιας διαδικτυακής πλατφόρμας ιατρικής πληροφόρησης με έμφαση σε τρεις κεντρικές δυνατότητες:

- οργανωμένη πρόσβαση σε πληροφορίες σχετικές με φάρμακα,
- διαχείριση προφίλ χρήστη για σκοπούς εξατομίκευσης, και
- παραγωγή προτάσεων και προειδοποιήσεων προσαρμοσμένων στο προφίλ.

Η πλατφόρμα αναπτύσσεται ως full-stack εφαρμογή βασισμένη σε πολυεπίπεδη αρχιτεκτονική, όπου η παρουσίαση, η επιχειρησιακή λογική και η επιμονή δεδομένων διαχωρίζονται με σαφή τρόπο. Η προσέγγιση αυτή ενισχύει τη συντηρησιμότητα και τη μελλοντική επεκτασιμότητα, ενώ επιτρέπει καθαρή υλοποίηση λειτουργιών που απευθύνονται στον τελικό χρήστη, όπως η αναζήτηση, η επεξεργασία προφίλ και η προβολή εξατομικευμένων προτάσεων. Επιπλέον, η ασφάλεια και η τοπικοποίηση αντιμετωπίζονται ως θεμελιώδεις άξονες, ώστε το σύστημα να είναι αφενός ασφαλές ως προς τη διαχείριση δεδομένων χρήστη και αφετέρου προσβάσιμο σε πολλαπλές γλώσσες.

Πίνακας περιεχομένων

1 Αντικείμενο και στόχοι της εργασίας 1
1.1 Τεχνικοί στόχοι 1
1.2 Χαρακτηριστικά της πλατφόρμας 2
1.3 Μεθοδολογία υλοποίησης 3
1.4 Ανασκόπηση Υφιστάμενων Συστημάτων 4
2 Τεχνολογικό υπόβαθρο 5
2.1 Frontend 5
2.2 Backend 6
2.3 Ασφάλεια 6
2.4 Δεδομένα 7
2.5 Τοπικοποίηση (Localization) 8
3 Υλοποίηση συστήματος 9
3.1 Ανάλυση Απαιτήσεων και Περιπτώσεις Χρήσης 9
3.2 Υλοποίηση Frontend 10
3.2.1 Αρχιτεκτονική διεπαφής και προσέγγιση απόδοσης 10
3.2.2 Επαναχρησιμοποίηση layout και κοινά UI τμήματα 10
3.2.3 Model binding φορμών με Thymeleaf 11
3.2.4 Υποστήριξη διεπαφών βάσει λιστών και δυναμικών επιλογών 11
3.2.5 Ενσωμάτωση τοπικοποίησης στη διεπαφή 12
3.2.6 Σύνοψη 12
3.3 Υλοποίηση Backend 13
3.3.1 Πολυεπίπεδη δομή backend 13
3.3.2 Controller layer: routing και model preparation 13
3.3.3 Service layer: επιχειρησιακή λογική και ενορχήστρωση 14
3.3.4 Υλοποίηση recommendation logic στο service layer 15
3.3.5 Πρόσβαση στα δεδομένα από υπηρεσίες 15
3.3.6 End-to-end backend flow (παράδειγμα) 16
3.3.7 Σύνοψη 17
3.4 Υλοποίηση Ασφάλειας 17
3.4.1 Αρχιτεκτονική ασφάλειας και κεντρική παραμετροποίηση 17
3.4.2 Έλεγχος πρόσβασης σε endpoints (authorization policy) 18

3.4.3 Ροή αυθεντικοποίησης 18
3.4.4 Ροή αποσύνδεσης 19
3.4.5 Διαχείριση credentials και AuthenticationManager 19
3.4.6 Πρακτικό παράδειγμα ασφαλούς ροής 19
3.4.7 Σύνοψη 20
3.5 Υλοποίηση δεδομένων 20
3.5.1 Αρχιτεκτονική δεδομένων 20
3.5.2 Μοντελοποίηση οντοτήτων και σχέσεων 21
3.5.3 Επίπεδο repositories (αφαίρεση πρόσβασης δεδομένων) 22
3.5.4 Λογική δεδομένων στο επίπεδο υπηρεσιών και συναλλαγές .. 22
3.5.5 Πρακτική ροή δεδομένων από άκρο σε άκρο 23
3.5.6 Σύνοψη 23
3.6 Υλοποίηση τοπικοποίησης (Localization) 24
3.6.1 Στόχος και προσέγγιση i18n 24
3.6.2 Message bundles και οργάνωση μεταφράσεων 24
3.6.3 Locale resolution και αλλαγή γλώσσας μέσω παραμέτρου 25
3.6.4 Ενσωμάτωση μεταφράσεων στα templates (Thymeleaf) 25
3.6.5 Τοπικοποίηση δυναμικών πεδίων (παράδειγμα κατηγοριών) . 26
3.6.6 Σύνοψη 26
4 Παρουσίαση πλατφόρμας και λειτουργικότητα 28
4.1 Αρχική σελίδα 28
4.2 Σελίδα προφίλ χρήστη 31
4.3 Αναζήτηση φαρμάκων 36
4.4 Εξατομικευμένες προτάσεις 40
5 Βελτιώσεις και μελλοντικές επεκτάσεις 42
Βιβλιογραφία 43
Αναφορές 44

1 Αντικείμενο και στόχοι της εργασίας

Σκοπός της παρούσας διπλωματικής εργασίας είναι η ανάπτυξη και αξιολόγηση μιας πρακτικής διαδικτυακής πλατφόρμας ιατρικής πληροφόρησης, η οποία βελτιώνει την πρόσβαση των χρηστών σε γνώση σχετική με φαρμακευτική αγωγή, ενσωματώνοντας ταυτόχρονα εξατομίκευση βάσει προφίλ, ασφαλή διαχείριση λογαριασμών και πολυγλωσσική χρηστικότητα.

Το έργο τοποθετείται ως εφαρμοσμένη προσπάθεια μηχανικής λογισμικού στον χώρο της ιατρικής πληροφορικής, όπου ο κύριος στόχος δεν είναι μόνο η θεωρητική μοντελοποίηση, αλλά η παράδοση ενός λειτουργικού συστήματος με σαφώς ορισμένη λειτουργικότητα και επεκτάσιμη αρχιτεκτονική.

Σε υψηλό επίπεδο, η εργασία επιδιώκει να καταδείξει ότι μια ελαφριά full-stack αρχιτεκτονική μπορεί να υποστηρίξει ουσιαστικές ροές ιατρικής πληροφόρησης, όταν συνδυάζει:

- δομημένη πρόσβαση σε περιεχόμενο,
- λογική εξατομίκευσης βασισμένη σε δεδομένα προφίλ χρήστη, και
- βασικές απαιτήσεις ποιότητας, όπως ασφάλεια και τοπικοποίηση.

1.1 Τεχνικοί στόχοι

Οι τεχνικοί στόχοι της εργασίας ορίζονται ως συγκεκριμένοι στόχοι υλοποίησης:

1. Σχεδίαση και υλοποίηση πολυεπίπεδης αρχιτεκτονικής web

   Ανάπτυξη ενός συντηρήσιμου συστήματος με διακριτό διαχωρισμό σε επίπεδα παρουσίασης, επιχειρησιακής λογικής και επιμονής δεδομένων, ώστε να υποστηρίζονται η αναγνωσιμότητα, η αρθρωτότητα και η μελλοντική επέκταση.

2. Υλοποίηση αξιόπιστης διαχείρισης προφίλ χρήστη

   Παροχή μηχανισμών για δημιουργία, επεξεργασία και αποθήκευση δεδομένων προφίλ που απαιτούνται από τις ροές εξατομικευμένων προτάσεων.

3. Ανάπτυξη λειτουργικότητας προτάσεων βάσει προφίλ

   Παραγωγή εξατομικευμένων αποτελεσμάτων προτάσεων και προειδοποιήσεων σχετικών με την ασφάλεια, μέσω επεξεργασίας χαρακτηριστικών προφίλ και διαθέσιμων δεδομένων της πλατφόρμας.

4. Ενεργοποίηση αναζητήσιμης πρόσβασης σε πληροφορίες φαρμάκων

Υποστήριξη αποδοτικής αλληλεπίδρασης του χρήστη με καταχωρίσεις φαρμάκων μέσω λειτουργιών αναζήτησης και σελίδων λεπτομερειών.

5. Ενσωμάτωση μηχανισμών ασφάλειας για προστατευμένες ροές χρήστη

   Εφαρμογή ασφαλούς αυθεντικοποίησης, προστασίας κωδικών πρόσβασης και ελεγχόμενης πρόσβασης σε λειτουργίες που εξαρτώνται από το προφίλ.

6. Υλοποίηση πολυγλωσσικής υποστήριξης

   Παροχή τουλάχιστον δίγλωσσης λειτουργίας (Αγγλικά και Ελληνικά) με συνεπή μετάφραση διεπαφής στις βασικές σελίδες.

7. Εξασφάλιση επεκτασιμότητας της υλοποίησης

   Οργάνωση κώδικα και συνιστωσών έτσι ώστε μελλοντικά χαρακτηριστικά (προηγμένοι κανόνες προτάσεων, APIs, αναλυτικά στοιχεία, mobile clients) να μπορούν να προστεθούν χωρίς εκτεταμένο ανασχεδιασμό.

1.2 Χαρακτηρηστικά της πλατφόρμας

Για την επίτευξη των παραπάνω στόχων, η υλοποιημένη πλατφόρμα περιλαμβάνει τα ακόλουθα βασικά χαρακτηριστικά:

- Διεπαφή αρχικής σελίδας και πλοήγησης για είσοδο σε όλες τις κύριες ενότητες της πλατφόρμας.
- Ροή αυθεντικοποίησης χρήστη για έλεγχο πρόσβασης βάσει λογαριασμού.
- Σελίδα προφίλ και λειτουργίες επεξεργασίας για καταγραφή και διατήρηση δεδομένων που σχετίζονται με την εξατομίκευση.
- Δυνατότητα αναζήτησης φαρμάκων για εντοπισμό καταχωρίσεων μέσω ερωτημάτων χρήστη.
- Σελίδες λεπτομερειών φαρμάκων που παρουσιάζουν δομημένη πληροφορία σε αναγνώσιμη μορφή.
- Μονάδα εξατομικευμένων προτάσεων που παράγει αποτελέσματα προσαρμοσμένα στον χρήστη.
- Επίπεδο προειδοποιήσεων προσανατολισμένο στην ασφάλεια που αναδεικνύει ζητήματα σχετιζόμενα με το προφίλ (όπου εφαρμόζεται).
- Υποστήριξη τοπικοποίησης για εναλλαγή γλώσσας διεπαφής μεταξύ Αγγλικών και Ελληνικών.
- Υποστηρικτικές ενημερωτικές σελίδες (π.χ. νομικά/σχετικά/υποστήριξη) που ενισχύουν τη χρηστικότητα και τη διαφάνεια της πλατφόρμας.

Τα χαρακτηριστικά αυτά επιλέχθηκαν ώστε να αποτυπώνουν μια ολοκληρωμένη διαδρομή χρήστη, από την πρώτη πρόσβαση στην πλατφόρμα έως την κατανάλωση εξατομικευμένων αποτελεσμάτων βάσει προφίλ.

1.3 Μεθοδολογία υλοποίησης

Η υλοποίηση ακολουθεί επαναληπτική και προσαυξητική (iterative and incremental) μεθοδολογία, η οποία επιλέχθηκε για τη μείωση ρίσκου και τη συνεχή βελτίωση της λειτουργικότητας.

Η αναπτυξιακή διαδικασία οργανώθηκε στις ακόλουθες φάσεις:

1. Καθορισμός απαιτήσεων

   Προσδιορισμός λειτουργικών απαιτήσεων (αναζήτηση, προφίλ, προτάσεις, τοπικοποίηση) και μη λειτουργικών απαιτήσεων (ασφάλεια, συντηρησιμότητα, χρηστικότητα).

2. Σχεδίαση συστήματος

   Επιλογή αρχιτεκτονικού προτύπου, σχεδιασμός μοντέλου δεδομένων, ορισμός ροών endpoints/σελίδων και διαμόρφωση στρατηγικής τοπικοποίησης και ασφάλειας.

3. Προσαυξητική υλοποίηση

   Ανάπτυξη ανά ενότητα (αυθεντικοποίηση, προφίλ, αναζήτηση, προτάσεις, τοπικοποίηση), με ενσωμάτωση και επικύρωση κάθε ενότητας πριν τη μετάβαση στην επόμενη.

4. Λειτουργική επαλήθευση και βελτίωση

   Εκτέλεση ελέγχων βάσει σεναρίων (τυπικές ροές χρήστη), διόρθωση σφαλμάτων και προσαρμογές σε επίπεδο χρηστικότητας για πρότυπα σελίδων και πλοήγηση.

5. Τεκμηρίωση και αξιολόγηση

   Καταγραφή τεχνικών αποφάσεων υλοποίησης, σχεδιαστικών συμβιβασμών, επιτευχθέντων στόχων και πεδίων μελλοντικής βελτίωσης.

Η συγκεκριμένη μεθοδολογία είναι συμβατή με το εύρος της εργασίας, καθώς επιτρέπει ελεγχόμενη πρόοδο από τη βασική υποδομή προς τη λειτουργικότητα που είναι ορατή στον χρήστη, διατηρώντας ταυτόχρονα το έργο ευέλικτο σε προσαρμογές απαιτήσεων που προκύπτουν κατά την πορεία υλοποίησης.

1.4 Ανασκόπηση Υφιστάμενων Συστημάτων

Η παροχή ιατρικής και φαρμακευτικής πληροφορίας μέσω του διαδικτύου δεν αποτελεί νέα πρακτική. Σήμερα, υπάρχουν πολυάριθμα συστήματα και πύλες (portals) υγείας που εξυπηρετούν εκατομμύρια χρήστες παγκοσμίως. Συστήματα όπως το WebMD, το Drugs.com, το MedlinePlus, αλλά και εθνικές πλατφόρμες όπως ο Εθνικός Οργανισμός Φαρμάκων (ΕΟΦ) ή ο ΕΟΠΥΥ στην Ελλάδα, προσφέρουν τεράστιο όγκο δεδομένων σχετικά με φαρμακευτικά σκευάσματα, δραστικές ουσίες, παρενέργειες και ιατρικές παθήσεις.

Ωστόσο, αναλύοντας τη σύγχρονη πραγματικότητα (state-of-the-art), παρατηρούνται συγκεκριμένες αδυναμίες στις υπάρχουσες προσεγγίσεις. Η συντριπτική πλειονότητα αυτών των συστημάτων υιοθετεί μια «στατική» προσέγγιση προβολής δεδομένων (one-size-fits-all). Ο χρήστης καλείται να αναζητήσει μια δραστική ουσία και να διαβάσει ένα εκτενές και συχνά δυσνόητο κείμενο πολλών σελίδων, προκειμένου να εντοπίσει αν το φάρμακο αντενδείκνυται για την περίπτωσή του. Η απουσία ενός δυναμικού, εξατομικευμένου φίλτρου που να λαμβάνει υπόψη το προσωπικό ιατρικό ιστορικό, τις τρέχουσες αγωγές και τις αλλεργίες του χρήστη, αφήνει το βάρος της κλινικής κρίσης αποκλειστικά στον ίδιο τον ασθενή, αυξάνοντας τον κίνδυνο ιατρικών σφαλμάτων από παρανόηση.

Επιπλέον, πολλές εμπορικές πλατφόρμες ενσωματώνουν διαφημιστικό περιεχόμενο ή δεν διαχωρίζουν επαρκώς τα τεκμηριωμένα ιατρικά δεδομένα από τα χορηγούμενα άρθρα, προκαλώντας σύγχυση. Στον αντίποδα, τα αμιγώς επιστημονικά συστήματα (όπως το PubMed ή το Galinos) απευθύνονται κυρίως σε επαγγελματίες υγείας, χρησιμοποιώντας βαριά ιατρική ορολογία που είναι δυσπρόσιτη για τον μέσο χρήστη.

Η πλατφόρμα που αναπτύσσεται στην παρούσα διπλωματική εργασία έρχεται να γεφυρώσει αυτό το χάσμα. Προτείνει ένα μοντέλο όπου η έγκυρη φαρμακευτική πληροφορία συναντά την προσωποποιημένη πληροφορική. Μέσω του μηχανισμού προφίλ (user profiling) και του συστήματος προτάσεων (recommendation engine), το σύστημα δεν απαντά απλώς στο ερώτημα «τι είναι αυτό το φάρμακο», αλλά στο πολύ πιο σύνθετο ερώτημα «τι σημαίνει αυτό το φάρμακο για εμένα, με βάση το ιστορικό μου». Αυτή η μετάβαση από τη γενική πληροφορία στην εξατομικευμένη γνώση αποτελεί την κύρια καινοτομία και προστιθέμενη αξία της υλοποίησης.

2 Τεχνολογικό υπόβαθρο

Το τεχνολογικό υπόβαθρο της πλατφόρμας επιλέχθηκε ώστε να καλύπτει πέντε βασικούς περιορισμούς της εργασίας: ταχεία ανάπτυξη, αρχιτεκτονική σαφήνεια, ασφαλή διαχείριση χρηστών, υποστήριξη δομημένων επίμονων δεδομένων και πολυγλωσσική εμπειρία χρήστη.

Αντί να δοθεί έμφαση στη μέγιστη τεχνολογική «καινοτομία», η επιλογή προτεραιοποιεί την αξιοπιστία, τη συντηρησιμότητα και τη συμβατότητα μεταξύ των επιμέρους συνιστωσών, ώστε η υλοποίηση να μπορεί να εξηγηθεί, να αξιολογηθεί και να επεκταθεί σε ελεγχόμενο ακαδημαϊκό πλαίσιο.

2.1 Frontend

Το frontend υλοποιείται με server-side rendered πρότυπα web (επίπεδο προβολής βασισμένο σε Thymeleaf), σε συνδυασμό με τυπικά HTML/CSS και προαιρετικές ελαφριές αλληλεπιδράσεις JavaScript όπου απαιτείται.

Η επιλογή αυτή έγινε για τους εξής λόγους:

- Άμεση ενσωμάτωση με τη ροή MVC του backend: τα server-rendered πρότυπα συνεργάζονται φυσικά με τις προβολές που επιστρέφονται από Spring MVC controllers.
- Μειωμένη αρχιτεκτονική πολυπλοκότητα: αποφεύγεται η εισαγωγή ξεχωριστού SPA build pipeline για το εύρος της παρούσας εργασίας.
- Γρήγορη πρωτοτυποποίηση και συντηρησιμότητα: οι σελίδες μπορούν να αναπτυχθούν και να ενημερωθούν γρήγορα, διατηρώντας αναγνώσιμη δομή προτύπων.
- Καλή προσαρμογή σε ροές βασισμένες σε φόρμες: η επεξεργασία προφίλ, η εισαγωγή αναζήτησης και η προβολή προτάσεων υλοποιούνται αποτελεσματικά σε server-rendered μοτίβα.
- Απλή σύνδεση με τοπικοποίηση: τα message keys εισάγονται απευθείας στα πρότυπα, μειώνοντας την πολυπλοκότητα i18n στο frontend.
- Για την κλίμακα του συγκεκριμένου έργου, η server-side απόδοση αποτελεί ρεαλιστική και πρακτική επιλογή, διατηρώντας την εστίαση στη λειτουργικότητα πεδίου (ιατρική πληροφορία + εξατομίκευση) αντί για πρόσθετο κόστος από βαριά client frameworks.

2.2 Backend

Το backend αναπτύχθηκε σε Java με χρήση Spring Boot, ακολουθώντας πολυεπίπεδη δομή (controllers, services, repositories/entities).

Η επιλογή αυτής της στοίβας τεχνολογιών βασίστηκε στους εξής λόγους:

- Ώριμο οικοσύστημα: το Spring Boot παρέχει πρότυπα και εργαλεία παραγωγικού επιπέδου με ισχυρή υποστήριξη κοινότητας.
- Dependency injection και αρθρωτός σχεδιασμός: βελτιώνουν τον διαχωρισμό ευθυνών και τη δυνατότητα ελέγχου (testability).
- Γρήγορη εκκίνηση web εφαρμογών: η αυτόματη παραμετροποίηση μειώνει τον boilerplate κώδικα και επιταχύνει την υλοποίηση.
- Συμβατότητα MVC με απόδοση προτύπων: εξασφαλίζεται καθαρή end-to-end ροή από το route, στη λογική εφαρμογής και τελικά στην προβολή.
- Κλιμακωσιμότητα δομής κώδικα: τα επίπεδα services και repositories μπορούν να επεκταθούν χωρίς ανασχεδιασμό ολόκληρης της εφαρμογής.
- Η συγκεκριμένη στοίβα υποστηρίζει τόσο τις απαιτήσεις της εργασίας όσο και μελλοντικές κατευθύνσεις επέκτασης (REST APIs, εξωτερικές διασυνδέσεις, πλουσιότερες υπηρεσίες προτάσεων).

2.3 Ασφάλεια

Η ασφάλεια υλοποιείται με Spring Security και πρακτικές κωδικοποίησης/κατακερματισμού κωδικών πρόσβασης για τα διαπιστευτήρια χρηστών.

Αιτιολόγηση επιλογής:

- Ασφαλείς προεπιλογές: το Spring Security παρέχει ισχυρή βασική προστασία έναντι συνήθων ευπαθειών σε ροές αυθεντικοποίησης web εφαρμογών.
- Υποστήριξη αυθεντικοποίησης/εξουσιοδότησης: επιτρέπει ελεγχόμενη πρόσβαση σε ενότητες προφίλ και εξατομίκευσης.
- Επεκτάσιμο μοντέλο πολιτικών: μπορούν να προστεθούν προοδευτικά περιορισμοί βάσει ρόλων ή συγκεκριμένων endpoints.
- Ενσωμάτωση password hashing: η αποθήκευση κωδικών σε κωδικοποιημένη μορφή ευθυγραμμίζεται με σύγχρονες απαιτήσεις ασφάλειας.
- Για μια πλατφόρμα ιατρικής πληροφόρησης που διαχειρίζεται δεδομένα προφίλ χρηστών, η ασφάλεια αποτελεί θεμελιώδη απαίτηση. Η επιλεγμένη προσέγγιση διασφαλίζει ότι η ασφάλεια ενσωματώνεται εξαρχής στην αρχιτεκτονική και δεν προστίθεται εκ των υστέρων.

2.4 Δεδομένα

Το επίπεδο δεδομένων της πλατφόρμας υλοποιείται με σχεσιακό μοντέλο επιμονής, χρησιμοποιώντας PostgreSQL ως σύστημα διαχείρισης βάσης δεδομένων, σε συνδυασμό με Spring Data JPA και Hibernate για object-relational mapping και πρόσβαση στα δεδομένα μέσω repositories. Ο συνδυασμός αυτός επιλέχθηκε ώστε να εξασφαλίζει συνέπεια, συντηρησιμότητα και αποδοτική διαχείριση δομημένων οντοτήτων πεδίου.

Η επιλογή του PostgreSQL βασίστηκε στα εξής:

- Ισχυρή συναλλακτική αξιοπιστία (ACID compliance), ιδιαίτερα σημαντική για ενημερώσεις προφίλ και συνεπή συμπεριφορά ανάγνωσης/εγγραφής.
- Ώριμες σχεσιακές δυνατότητες, κατάλληλες για οντότητες με σαφείς σχέσεις (χρήστες, προφίλ, φάρμακα, προτάσεις).
- Υψηλή σταθερότητα και ευρεία υποστήριξη οικοσυστήματος, που την καθιστούν κατάλληλη τόσο για πρωτότυπη υλοποίηση όσο και για μελλοντική παραγωγική εξέλιξη.

Πάνω από το PostgreSQL, η πλατφόρμα αξιοποιεί JPA/Hibernate για χαρτογράφηση Java οντοτήτων σε σχεσιακούς πίνακες. Η προσέγγιση αυτή μειώνει τον boilerplate SQL κώδικα για συνήθεις λειτουργίες και διατηρεί τη λογική πεδίου κοντά στο μοντέλο της εφαρμογής. Τα repository interfaces αφαιρούν πολυπλοκότητα από τις βασικές CRUD και query λειτουργίες, επιτρέποντας στο επίπεδο υπηρεσιών να εστιάζει στη συμπεριφορά της επιχειρησιακής λογικής και όχι στους χαμηλού επιπέδου μηχανισμούς πρόσβασης δεδομένων.

Το μοντέλο δεδομένων οργανώνεται γύρω από τους βασικούς άξονες της πλατφόρμας:

- οντότητες σχετικές με χρήστη και αυθεντικοποίηση,
- χαρακτηριστικά προφίλ χρήστη που αξιοποιούνται για εξατομίκευση,
- δομές σχετικές με φαρμακευτική πληροφορία και προτάσεις.

Ο σχεδιασμός αυτός υποστηρίζει καθαρό διαχωρισμό ευθυνών: οι οντότητες ορίζουν το μοντέλο πεδίου, τα repositories διαχειρίζονται τα μοτίβα πρόσβασης και οι υπηρεσίες ενορχηστρώνουν τους επιχειρησιακούς κανόνες. Ως αποτέλεσμα, το επίπεδο δεδομένων παραμένει επεκτάσιμο και μπορεί να υποστηρίξει μελλοντικές προσθήκες, όπως πλουσιότερο ιστορικό προτάσεων, προηγμένα φίλτρα, μηχανισμούς audit trail ή σχήματα εξαγωγής για διασυνδέσεις.

Συνολικά, η επιλεγμένη στοίβα δεδομένων (PostgreSQL + Spring Data JPA + Hibernate) προσφέρει ισορροπημένη λύση για την παρούσα εργασία: σχεσιακή σαφήνεια, αξιόπιστη συνέπεια και αποδοτικότητα υλοποίησης, σε ευθυγράμμιση με τους λειτουργικούς και αρχιτεκτονικούς στόχους της πλατφόρμας.

2.5 Τοπικοποίηση (Localization)

Η τοπικοποίηση υλοποιείται μέσω message resource bundles (π.χ. αρχεία ιδιοτήτων για Αγγλικά και Ελληνικά), σε συνδυασμό με ρύθμιση locale στο επίπεδο της εφαρμογής.

Η επιλογή αυτή βασίστηκε στους εξής λόγους:

- Εγγενής υποστήριξη από το framework: ο μηχανισμός επίλυσης μηνυμάτων του Spring ενσωματώνεται άμεσα με Thymeleaf και controllers.
- Διαχωρισμός περιεχομένου από λογική: τα μεταφράσιμα κείμενα παραμένουν εξωτερικοποιημένα, βελτιώνοντας τη συντηρησιμότητα.
- Κλιμακώσιμη πολυγλωσσική πορεία: η προσθήκη νέων γλωσσών μπορεί να γίνει με ελάχιστες αλλαγές στον κώδικα.
- Συνεπής εμπειρία χρήστη σε όλες τις σελίδες: κοινές ετικέτες, φόρμες και μηνύματα μεταφράζονται ομοιόμορφα.
- Με βάση το πλαίσιο χρηστών-στόχου, η δίγλωσση υποστήριξη αποτελεί λειτουργική απαίτηση και ο συγκεκριμένος μηχανισμός προσφέρει καθαρή και επεκτάσιμη υλοποίηση i18n.

Συνολικά, το επιλεγμένο τεχνολογικό υπόβαθρο είναι συνειδητά συνεκτικό: κάθε επίπεδο (frontend, backend, ασφάλεια, δεδομένα, τοπικοποίηση) είναι συμβατό με τα υπόλοιπα και ευθυγραμμισμένο με τους στόχους της εργασίας για την ανάπτυξη μιας ασφαλούς, εξατομικευμένης και συντηρήσιμης πλατφόρμας ιατρικής πληροφόρησης.

3 Υλοποίηση συστήματος

Το σύστημα ακολουθεί πολυεπίπεδη αρχιτεκτονική, όπου οι controllers διαχειρίζονται τα HTTP αιτήματα, τα services εφαρμόζουν την επιχειρησιακή λογική, τα repositories αναλαμβάνουν την επιμονή δεδομένων και τα templates αποδίδουν τις σελίδες που αλληλεπιδρούν με τον τελικό χρήστη.

3.1 Ανάλυση Απαιτήσεων και Περιπτώσεις Χρήσης

Πριν από την υλοποίηση της πολυεπίπεδης αρχιτεκτονικής, πραγματοποιήθηκε εκτενής ανάλυση απαιτήσεων για να καθοριστούν οι αλληλεπιδράσεις μεταξύ των χρηστών (actors) και του συστήματος. Η ανάλυση αυτή οδήγησε στον καθορισμό των Λειτουργικών (Functional) και Μη Λειτουργικών (Non-Functional) απαιτήσεων, οι οποίες μοντελοποιήθηκαν μέσω Περιπτώσεων Χρήσης (Use Cases).

Οντότητες - Χρήστες (Actors):

Στο σύστημα αναγνωρίζονται δύο βασικές κατηγορίες χρηστών:

- Απλός Επισκέπτης (Guest User): Μη ταυτοποιημένος χρήστης. Έχει τη δυνατότητα να πλοηγηθεί στις δημόσιες σελίδες της πλατφόρμας, να χρησιμοποιήσει τη μηχανή αναζήτησης για την εύρεση φαρμάκων και να διαβάσει τις γενικές πληροφορίες των σκευασμάτων. Μπορεί επίσης να εγγραφεί στο σύστημα δημιουργώντας νέο λογαριασμό.

- Ταυτοποιημένος Χρήστης (Authenticated User): Χρήστης που έχει συνδεθεί επιτυχώς στην πλατφόρμα. Κληρονομεί όλα τα δικαιώματα του επισκέπτη και αποκτά επιπλέον πρόσβαση στον προσωπικό του χώρο (dashboard). Μπορεί να επεξεργαστεί το ιατρικό του προφίλ, να καταχωρήσει παθήσεις και αλλεργίες, και να δει τις εξατομικευμένες προτάσεις και προειδοποιήσεις που παράγει το σύστημα.

Βασικές Περιπτώσεις Χρήσης (Core Use Cases):

- UC-01: Αναζήτηση και Φιλτράρισμα Φαρμάκων: Ο χρήστης εισάγει έναν όρο αναζήτησης (π.χ., δραστική ουσία, εμπορική ονομασία) ή επιλέγει ένα φίλτρο (π.χ., Μη Συνταγογραφούμενα Φάρμακα). Το σύστημα εκτελεί το ερώτημα στη βάση δεδομένων και επιστρέφει σελιδοποιημένα (paginated) αποτελέσματα.

- UC-02: Διαχείριση Ιατρικού Προφίλ: Ο ταυτοποιημένος χρήστης μεταβαίνει στη σελίδα επεξεργασίας προφίλ. Το σύστημα αντλεί τα υφιστάμενα δεδομένα του (βιομετρικά, παθήσεις, αλλεργίες) και τα προβάλλει στη φόρμα. Ο χρήστης τροποποιεί τα δεδομένα και το σύστημα επικυρώνει τις καταχωρήσεις πριν τις αποθηκεύσει με ασφάλεια (transactional save).

- UC-03: Παραγωγή Εξατομικευμένων Προτάσεων: Αυτή η περίπτωση χρήσης εκτελείται αυτόματα στο παρασκήνιο (backend) όταν ο ταυτοποιημένος χρήστης επισκέπτεται τη σελίδα των προτάσεων. Το σύστημα αναλύει το αποθηκευμένο προφίλ του (UC-02), το διασταυρώνει με τους κανόνες συσχέτισης φαρμάκων/παθήσεων της βάσης δεδομένων και επιστρέφει μια λίστα με κατάλληλα σκευάσματα, συμπληρώματα, καθώς και κρίσιμες προειδοποιήσεις αλληλεπιδράσεων (interaction warnings).

- UC-04: Τοπικοποίηση Διεπαφής (Localization): Ο χρήστης επιλέγει γλώσσα (Αγγλικά ή Ελληνικά) από το κεντρικό μενού. Το σύστημα ανανεώνει τη συνεδρία (session locale) και επαναφορτώνει τη σελίδα εξυπηρετώντας τα αντίστοιχα message bundles, χωρίς να διακόπτεται η τρέχουσα ροή εργασίας του χρήστη.

Η υιοθέτηση αυτών των περιπτώσεων χρήσης εξασφάλισε ότι η ανάπτυξη της εφαρμογής παρέμεινε προσηλωμένη στην παραγωγή αξίας για τον τελικό χρήστη, καθοδηγώντας τον σχεδιασμό των controllers, των υπηρεσιών και του μοντέλου δεδομένων που αναλύονται στις επόμενες ενότητες.

3.2 Υλοποίηση Frontend

3.2.1 Αρχιτεκτονική διεπαφής και προσέγγιση απόδοσης

Το frontend της πλατφόρμας υλοποιείται με server-side rendered πρότυπα Thymeleaf.

Η προσέγγιση αυτή επιτρέπει άμεση σύνδεση της διεπαφής με το model του backend, χωρίς την επιπλέον πολυπλοκότητα ενός ξεχωριστού SPA frontend.

Κάθε αίτημα χρήστη ακολουθεί ροή:

1. route σε controller,
2. προετοιμασία model attributes,
3. απόδοση template με Thymeleaf.

Το μοντέλο αυτό είναι κατάλληλο για φόρμες, φίλτρα αναζήτησης και προβολή εξατομικευμένων αποτελεσμάτων.

3.2.2 Επαναχρησιμοποίηση layout και κοινά UI τμήματα

Για να αποφεύγεται διπλοεγγραφή κώδικα UI, χρησιμοποιούνται κοινά fragments (π.χ. navbar, footer, scripts) από το βασικό layout.

Έτσι εξασφαλίζεται οπτική συνέπεια σε όλες τις σελίδες και γρήγορη συντήρηση.

```
<div th:replace="~{layout/base :: navbar}"></div>
...
<div th:replace="~{layout/base :: footer}"></div>
<div th:replace="~{layout/base :: scripts}"></div>
```

Με αυτή τη δομή, αλλαγές σε κοινά στοιχεία εφαρμόζονται κεντρικά και διαχέονται σε όλη την εφαρμογή.

3.2.3 Model binding φορμών με Thymeleaf

Στη σελίδα profile/edit, τα πεδία φόρμας συνδέονται απευθείας με το αντικείμενο UserProfile μέσω th:object και th:field.

Αυτό μειώνει τη χειροκίνητη διαχείριση τιμών και ενισχύει τη συνέπεια frontend-backend.

```
<form th:action="@{/profile/update}" method="post" th:object="${profile}">
  <label for="age" class="form-label" th:text="#{profile.basic_info.age}">Age</label>
  <input type="number" class="form-control" id="age" th:field="*{age}" min="0" max="150">
  <label for="gender" class="form-label" th:text="#{profile.basic_info.gender}">Gender</label>
  <select class="form-control" id="gender" th:field="*{gender}">
    <option value="" th:text="#{form.select.placeholder}">Select...</option>
    <option th:each="g : ${genders}" th:value="${g}" th:text="${g}"></option>
  </select>
  <button type="submit" class="btn btn-primary" th:text="#{profile.edit.save_basic}">
    Save Basic Information
  </button>
</form>
```

Η παραπάνω λογική επιτρέπει αξιόπιστη μεταφορά δεδομένων χρήστη στο backend για αποθήκευση.

3.2.4 Υποστήριξη διεπαφών βάσει λιστών και δυναμικών επιλογών

Το frontend χρησιμοποιεί δυναμικά δεδομένα από το model για λίστες (π.χ. genders, bloodTypes, drugs, commonConditions) με th:each.

Αυτό επιτρέπει ανανεώσιμες επιλογές χωρίς hardcoded τιμές στο template.

```
<select class="form-select" name="drugId">
```

```
    <option value="" th:text="#{profile.medications.select_medication}">Select medication...</option>
    <option th:each="drug : ${drugs}"
        th:value="${drug.id}"
        th:text="${drug.strength != null ? drug.name + ' (' + drug.strength + ')' : drug.name}">
    </option>
</select>
```

Έτσι το UI παραμένει συγχρονισμένο με τα πραγματικά δεδομένα της βάσης.

3.2.5 Ενσωμάτωση τοπικοποίησης στη διεπαφή

Η τοπικοποίηση εφαρμόζεται απευθείας στα templates μέσω message keys (#{...}), ώστε η ίδια σελίδα να αποδίδεται σε διαφορετικές γλώσσες χωρίς αλλαγή markup ή business logic.

```
<title th:text="#{profile.edit.title}">Edit Profile - MedInfo</title>
<h5 th:text="#{profile.conditions.title}">Medical Conditions</h5>
<small class="text-muted" th:text="#{form.multiselect.hint}">
  Hold Ctrl (Cmd on Mac) to select multiple.
</small>
```

Η πρακτική αυτή κρατά ξεχωριστά το περιεχόμενο από τη δομή και υποστηρίζει εύκολη επέκταση σε επιπλέον γλώσσες.

3.2.6 Σύνοψη

Η υλοποίηση frontend της πλατφόρμας βασίζεται σε:

server-side rendering με Thymeleaf,
επαναχρησιμοποιήσιμα layout fragments,
άμεσο model-to-form binding,
δυναμική απόδοση λιστών από backend δεδομένα,
εγγενή υποστήριξη τοπικοποίησης.

Το αποτέλεσμα είναι ένα frontend καθαρό, συντηρήσιμο και κατάλληλο για ροές ιατρικής πληροφόρησης που βασίζονται σε φόρμες και εξατομίκευση.

3.3 Υλοποίηση Backend

Το backend υλοποιείται με Spring Boot, ακολουθώντας πολυεπίπεδο σχεδιασμό:

- Επίπεδο Controller: διαχείριση διαδρομών (routes) και προετοιμασία του model.
- Επίπεδο Service: εφαρμογή επιχειρησιακής λογικής και ενορχήστρωση ροών.
- Επίπεδο Repository: επιμονή οντοτήτων και πρόσβαση στα δεδομένα.

Αντιπροσωπευτικό παράδειγμα αποτελεί η ροή ενημέρωσης προφίλ. Ο controller ανακτά τον τρέχοντα αυθεντικοποιημένο χρήστη, φορτώνει τα δεδομένα προφίλ, εφαρμόζει τις αλλαγές που προέρχονται από το form model και αναθέτει την αποθήκευση στο service layer.

3.3.1 Πολυεπίπεδη δομή backend

Το backend της πλατφόρμας υλοποιείται με Spring Boot και ακολουθεί πολυεπίπεδη αρχιτεκτονική:

- Controller layer για διαχείριση αιτημάτων HTTP και απόδοση views,
- Service layer για επιχειρησιακή λογική και ενορχήστρωση,
- Repository layer για πρόσβαση και επιμονή δεδομένων.

Η δομή αυτή μειώνει τη σύζευξη μεταξύ επιπέδων και επιτρέπει πιο εύκολη συντήρηση/επέκταση.

3.3.2 Controller layer: routing και model preparation

Οι controllers ορίζουν endpoints και μεταφέρουν στο frontend τα απαραίτητα model attributes.

Παράδειγμα από το UserProfileController:

```
@GetMapping("/edit")
public String editProfile(Model model) {
   Long userId = getCurrentUserId();
   if (userId == null) {
      return "redirect:/login";
   }
   UserProfile profile = userProfileService.getProfileByUserId(userId)
      .orElseGet(() -> {
         User user = userService.findById(userId).orElse(null);
         if (user != null) {
```

```
                return userProfileService.createProfile(user);
            }
            return null;
        });
    model.addAttribute("profile", profile);
    model.addAttribute("commonConditions", getCommonConditions());
    model.addAttribute("genders", Arrays.asList("Male", "Female"));
    model.addAttribute("bloodTypes", Arrays.asList("A+", "A-", "B+", "B-", "AB+", "AB-", "O+", "O-
"));
    model.addAttribute("drugs", drugService.getAllDrugs());
    return "profile/edit";
}
```

Ο controller παραμένει εστιασμένος στη ροή αιτήματος/απόκρισης, χωρίς να περιέχει βαριά επιχειρησιακή λογική.

3.3.3 Service layer: επιχειρησιακή λογική και ενορχήστρωση

Το service layer υλοποιεί τους κανόνες του domain και συντονίζει τα repositories.

Παράδειγμα ενημέρωσης προφίλ:

```
public UserProfile updateProfile(Long profileId, UserProfile updatedProfile) {
    UserProfile profile = userProfileRepository.findById(profileId)
        .orElseThrow(() -> new RuntimeException("Profile not found"));
    profile.setAge(updatedProfile.getAge());
    profile.setGender(updatedProfile.getGender());
    profile.setWeight(updatedProfile.getWeight());
    profile.setHeight(updatedProfile.getHeight());
    profile.setBloodType(updatedProfile.getBloodType());
    return userProfileRepository.save(profile);
}
```

Με αυτό το μοτίβο, οι controllers καλούν μεθόδους υψηλού επιπέδου αντί να χειρίζονται απευθείας persistence/κανόνες.

3.3.4 Υλοποίηση recommendation logic στο service layer

Η παραγωγή εξατομικευμένων προτάσεων υλοποιείται στο RecommendationService, όπου γίνεται μετασχηματισμός δεδομένων προφίλ σε αποτελέσματα προτάσεων και προειδοποιήσεων.

```
public PersonalizedRecommendations getRecommendations(Long userId) {
    UserProfile profile = userProfileService.getProfileByUserId(userId)
        .orElseThrow(() -> new RuntimeException("User profile not found"));
    PersonalizedRecommendations recommendations = new PersonalizedRecommendations();
    List<Drug> recommendedDrugs = getDrugsForConditions(profile.getMedicalConditions());
    recommendations.setRecommendedDrugs(recommendedDrugs);
    List<Supplement> recommendedSupplements =
getSupplementsForConditions(profile.getMedicalConditions());
    recommendations.setRecommendedSupplements(recommendedSupplements);
    List<PersonalizedRecommendations.InteractionWarning> interactionWarnings =
        checkInteractions(profile.getCurrentMedications(), recommendedDrugs);
    recommendations.setInteractionWarnings(interactionWarnings);
    List<PersonalizedRecommendations.AllergyWarning> allergyWarnings =
        checkAllergies(profile.getAllergies(), recommendedDrugs, recommendedSupplements);
    recommendations.setAllergyWarnings(allergyWarnings);
    return recommendations;
}
```

Η τοποθέτηση αυτής της λογικής στο service layer βελτιώνει testability και επαναχρησιμοποίηση.

3.3.5 Πρόσβαση στα δεδομένα από υπηρεσίες

Το επίπεδο persistence στη πλατφόρμα υλοποιείται μέσω repositories που επεκτείνουν το JpaRepository.

Στην πράξη, τα repositories λειτουργούν ως «γέφυρα» ανάμεσα στις υπηρεσίες (service layer) και τη βάση δεδομένων: οι υπηρεσίες καλούν μεθόδους όπως findById, findAll, save και deleteById, χωρίς να γράφουν SQL.

Με αυτόν τον τρόπο, η πρόσβαση στα δεδομένα γίνεται δηλωτικά (μέσω έτοιμων ή derived μεθόδων), ενώ η λογική της εφαρμογής παραμένει συγκεντρωμένη στο service layer και όχι σε χαμηλού επιπέδου κώδικα πρόσβασης δεδομένων.

```
public interface DrugRepository extends JpaRepository<Drug, Long> {
    List<Drug> findByNameContainingIgnoreCase(String name);
```

```
    @Query("SELECT d FROM Drug d WHERE " +
        "LOWER(d.name) LIKE LOWER(CONCAT('%', :searchTerm, '%')) OR " +
        "LOWER(d.activeIngredient) LIKE LOWER(CONCAT('%', :searchTerm, '%'))")
    Page<Drug> searchDrugs(@Param("searchTerm") String searchTerm, Pageable pageable);
}
```

Στη συνέχεια το service layer αξιοποιεί τις μεθόδους αυτές για φιλτράρισμα, αναζήτηση και pagination χωρίς χειροκίνητη SQL.

3.3.6 End-to-end backend flow (παράδειγμα)

Τυπική ροή ενημέρωσης προφίλ:

1. Ο χρήστης υποβάλλει φόρμα στο /profile/update.
2. Ο controller λαμβάνει το model (@ModelAttribute UserProfile).
3. Το service φορτώνει την υπάρχουσα οντότητα και εφαρμόζει αλλαγές.
4. Το repository εκτελεί save(...) μέσω Hibernate/JPA.
5. Ο controller κάνει redirect στη σελίδα προφίλ.

```
@PostMapping("/update")
public String updateProfile(@ModelAttribute UserProfile updatedProfile) {
    Long userId = getCurrentUserId();
    if (userId == null) {
        return "redirect:/login";
    }
    UserProfile profile = userProfileService.getProfileByUserId(userId)
        .orElseThrow(() -> new RuntimeException("Profile not found"));
    profile.setAge(updatedProfile.getAge());
    profile.setGender(updatedProfile.getGender());
    profile.setWeight(updatedProfile.getWeight());
    profile.setHeight(updatedProfile.getHeight());
    profile.setBloodType(updatedProfile.getBloodType());
    userProfileService.saveProfile(profile);
    return "redirect:/profile";
}
```

3.3.7 Σύνοψη

Η backend υλοποίηση της πλατφόρμας βασίζεται σε καθαρό διαχωρισμό επιπέδων:

- controllers για HTTP ροές και model rendering,
- services για domain logic και orchestration,
- repositories για persistence abstraction.

Έτσι επιτυγχάνονται καθαρότητα κώδικα, καλύτερη δυνατότητα ελέγχου και ασφαλής βάση για μελλοντικές επεκτάσεις (νέοι κανόνες προτάσεων, REST APIs, εξωτερικές διασυνδέσεις).

3.4 Υλοποίηση Ασφάλειας

Η ασφάλεια διαμορφώνεται κεντρικά μέσω του Spring Security. Οι δημόσιες διαδρομές δηλώνονται ρητά ως προσβάσιμες, ενώ οι διαδρομές προφίλ και προτάσεων απαιτούν αυθεντικοποίηση.

Η υλοποίηση χρησιμοποιεί επίσης form-based login και ελεγχόμενη ροή αποσύνδεσης (logout).

3.4.1 Αρχιτεκτονική ασφάλειας και κεντρική παραμετροποίηση

Η ασφάλεια της πλατφόρμας υλοποιείται κεντρικά μέσω Spring Security.

Αντί για διάσπαρτους ελέγχους σε controllers, εφαρμόζεται ενιαία πολιτική πρόσβασης σε επίπεδο SecurityFilterChain, ώστε οι κανόνες να είναι σαφείς, συνεπείς και εύκολα επεκτάσιμοι.

```
@Configuration
@EnableWebSecurity
public class SecurityConfig {
   @Bean
   public SecurityFilterChain securityFilterChain(HttpSecurity http) throws Exception {
      http
         .csrf(csrf -> csrf.disable())
         .authorizeHttpRequests(auth -> auth
            // ...
         );
      return http.build();
   }
}
```

Η προσέγγιση αυτή ενσωματώνει την ασφάλεια στην αρχιτεκτονική του συστήματος ως βασικό δομικό στοιχείο.

3.4.2 Έλεγχος πρόσβασης σε endpoints (authorization policy)

Στην πολιτική πρόσβασης, οι δημόσιες σελίδες δηλώνονται ρητά ως permitAll, ενώ οι ενότητες που βασίζονται σε προσωπικά δεδομένα απαιτούν αυθεντικοποίηση.

```
.authorizeHttpRequests(auth -> auth
   .requestMatchers("/", "/drugs/**", "/supplements/**", "/about/**",
            "/support/**", "/legal/**", "/register", "/login",
            "/css/**", "/js/**", "/images/**", "/error").permitAll()
   .requestMatchers("/profile/**", "/recommendations/**").authenticated()
   .anyRequest().authenticated()
)
```

Με αυτό το μοντέλο:

- η πληροφοριακή πλοήγηση παραμένει ανοιχτή,
- οι εξατομικευμένες λειτουργίες προστατεύονται.

3.4.3 Ροή αυθεντικοποίησης

Η αυθεντικοποίηση γίνεται με form-based login, με προσαρμοσμένη σελίδα εισόδου και σαφή ροή επιτυχίας/αποτυχίας.

```
.formLogin(form -> form
   .loginPage("/login")
   .defaultSuccessUrl("/profile", true)
   .failureUrl("/login?error=true")
   .permitAll()
)
```

Η ρύθμιση αυτή:

- οδηγεί τον επιτυχή χρήστη στο προσωπικό περιβάλλον (/profile),

- επιστρέφει πληροφορία αποτυχίας για λανθασμένα διαπιστευτήρια,
- διατηρεί ελεγχόμενη και κατανοητή εμπειρία σύνδεσης.

3.4.4 Ροή αποσύνδεσης

Η αποσύνδεση υλοποιείται επίσης μέσω Spring Security με καθορισμένο redirect μετά το logout.

```
.logout(logout -> logout
   .logoutSuccessUrl("/login?logout=true")
   .permitAll()
)
```

Έτσι ολοκληρώνεται η ροή συνεδρίας με προβλέψιμο τρόπο, χωρίς custom ad-hoc λογική σε controllers.

3.4.5 Διαχείριση credentials και AuthenticationManager

Ο AuthenticationManager εκτίθεται ως bean και συνεργάζεται με τον μηχανισμό αυθεντικοποίησης του Spring.

Παράλληλα, η προστασία κωδικών υλοποιείται με password encoding (hashed αποθήκευση, όχι plain text).

```
@Bean
public AuthenticationManager authenticationManager(AuthenticationConfiguration config) throws Exception {
   return config.getAuthenticationManager();
}
```

Η χρήση του ενσωματωμένου authentication stack του framework μειώνει τον κίνδυνο λανθασμένης custom υλοποίησης.

3.4.6 Πρακτικό παράδειγμα ασφαλούς ροής

Τυπική ασφαλής ροή στη πλατφόρμα:

- Μη αυθεντικοποιημένος χρήστης ζητά /profile.
- Η πολιτική authenticated() μπλοκάρει την πρόσβαση.
- Ο χρήστης ανακατευθύνεται σε /login.

- Με επιτυχή είσοδο, μεταφέρεται στο /profile.
- Με logout, επιστρέφει στο /login?logout=true.

Αυτό αποδεικνύει ότι η πρόσβαση σε προφίλ/προτάσεις γίνεται μόνο εντός έγκυρης συνεδρίας χρήστη.

3.4.7 Σύνοψη

Η υλοποίηση ασφάλειας της πλατφόρμας βασίζεται σε:

- κεντρική πολιτική endpoint authorization,
- form-based authentication με ελεγχόμενη ροή,
- προστατευμένες διαδρομές για εξατομικευμένες λειτουργίες,
- ενσωμάτωση μηχανισμών Spring Security για συνεπή διαχείριση συνεδρίας.

Το αποτέλεσμα είναι ένα ασφαλές πλαίσιο λειτουργίας που υποστηρίζει την επεξεργασία προσωπικών δεδομένων και τη διάθεση profile-aware υπηρεσιών με ελεγχόμενη πρόσβαση.

3.5 Υλοποίηση δεδομένων

3.5.1 Αρχιτεκτονική δεδομένων

Η πλατφόρμα υλοποιεί σχεσιακή αρχιτεκτονική επιμονής με βάση το PostgreSQL, χρησιμοποιώντας Spring Data JPA και Hibernate ως επίπεδο ORM/διαχείρισης επιμονής.

Το PostgreSQL παρέχει συναλλακτική συνέπεια, ενώ το Hibernate αναλαμβάνει τη χαρτογράφηση των Java οντοτήτων σε σχεσιακούς πίνακες και τη δημιουργία/εκτέλεση SQL πράξεων.

```
# Database Configuration (PostgreSQL Server)
spring.datasource.url=jdbc:postgresql://<host>:5432/postgres
spring.datasource.driver-class-name=org.postgresql.Driver
# JPA/Hibernate Configuration
spring.jpa.hibernate.ddl-auto=update
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.PostgreSQLDialect
```

```
spring.jpa.show-sql=true
```

Η παραπάνω ρύθμιση επιτρέπει αξιόπιστη σχεσιακή αποθήκευση και ταχύτερη ανάπτυξη με ελάχιστο χειροκίνητο SQL κώδικα.

3.5.2 Μοντελοποίηση οντοτήτων και σχέσεων

Οι οντότητες του πεδίου μοντελοποιούνται μέσω JPA annotations.

Κεντρικό παράδειγμα αποτελεί η οντότητα UserProfile, η οποία αποθηκεύει χαρακτηριστικά προφίλ και συνδέεται με ιατρικές καταστάσεις, τρέχουσα φαρμακευτική αγωγή και αλλεργίες μέσω χαρτογραφημένων συσχετίσεων.

```
@Entity
@Table(name = "user_profiles")
public class UserProfile {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    @OneToOne
    @JoinColumn(name = "user_id", nullable = false, unique = true)
    private User user;
    @OneToMany(mappedBy = "userProfile", cascade = CascadeType.ALL, orphanRemoval = true, fetch = FetchType.EAGER)
    private List<UserMedicalCondition> medicalConditions = new ArrayList<>();
    @OneToMany(mappedBy = "userProfile", cascade = CascadeType.ALL, orphanRemoval = true, fetch = FetchType.EAGER)
    private List<UserCurrentMedication> currentMedications = new ArrayList<>();
    @OneToMany(mappedBy = "userProfile", cascade = CascadeType.ALL, orphanRemoval = true, fetch = FetchType.EAGER)
    private List<UserAllergy> allergies = new ArrayList<>();
}
```

Το μοντέλο αυτό υποστηρίζει δομημένη αναπαράσταση του ιατρικού πλαισίου κάθε χρήστη, το οποίο αξιοποιείται στις ροές εξατομίκευσης.

3.5.3 Επίπεδο repositories (αφαίρεση πρόσβασης δεδομένων)

Η πρόσβαση στα δεδομένα υλοποιείται μέσω repository interfaces που επεκτείνουν το JpaRepository.

Τα interfaces αυτά προσφέρουν έτοιμες CRUD λειτουργίες και derived queries, χωρίς την ανάγκη χαμηλού επιπέδου DAO boilerplate.

```
@Repository
public interface UserProfileRepository extends JpaRepository<UserProfile, Long> {
   Optional<UserProfile> findByUserId(Long userId);
   Optional<UserProfile> findByUserUsername(String username);
}
```

Για την αναζήτηση φαρμάκων, τα repositories περιλαμβάνουν επίσης custom queries και υποστήριξη σελιδοποίησης:

```
@Repository
public interface DrugRepository extends JpaRepository<Drug, Long> {
   List<Drug> findByNameContainingIgnoreCase(String name);
   @Query("SELECT d FROM Drug d WHERE " +
        "LOWER(d.name) LIKE LOWER(CONCAT('%', :searchTerm, '%')) OR " +
        "LOWER(d.activeIngredient) LIKE LOWER(CONCAT('%', :searchTerm, '%'))")
   Page<Drug> searchDrugs(@Param("searchTerm") String searchTerm, Pageable pageable);
}
```

3.5.4 Λογική δεδομένων στο επίπεδο υπηρεσιών και συναλλαγές

Η λογική διαχείρισης δεδομένων υλοποιείται στο επίπεδο υπηρεσιών, όπου τα repositories ενορχηστρώνονται μέσα σε transactional όρια (@Transactional).

Έτσι εξασφαλίζεται συνέπεια όταν ενημερώνονται προφίλ και συσχετισμένες συλλογές οντοτήτων.

```
@Service
@Transactional
public class UserProfileService {
   @Autowired
   private UserProfileRepository userProfileRepository;
   public UserProfile createProfile(User user) {
      UserProfile profile = new UserProfile(user);
      return userProfileRepository.save(profile);
   }
```

```
    public UserProfile saveProfile(UserProfile profile) {
        return userProfileRepository.save(profile);
    }
    public void addCurrentMedication(Long profileId, UserCurrentMedication medication) {
        UserProfile profile = userProfileRepository.findById(profileId)
            .orElseThrow(() -> new RuntimeException("Profile not found"));
        profile.addCurrentMedication(medication);
        userProfileRepository.save(profile);
    }
}
```

Ο σχεδιασμός αυτός διατηρεί τους controllers «λεπτούς» και συγκεντρώνει τη λογική συνέπειας δεδομένων στο service layer.

3.5.5 Πρακτική ροή δεδομένων από άκρο σε άκρο

Μια τυπική ροή ενημέρωσης προφίλ περιλαμβάνει:

Ο controller λαμβάνει αίτημα ενημέρωσης προφίλ.
Το service φορτώνει την υπάρχουσα οντότητα UserProfile.
Εφαρμόζονται οι νέες τιμές στα πεδία της οντότητας.
Το repository save(...) επιμένει τις αλλαγές μέσω Hibernate.
Τα ενημερωμένα δεδομένα αξιοποιούνται από υπηρεσίες προτάσεων.
Παράδειγμα λειτουργίας επιμονής από την ενότητα φαρμάκων:

```
public Drug saveDrug(Drug drug) {
    drug.setUpdatedAt(LocalDate.now());
    return drugRepository.save(drug);
}
```

Η παραπάνω πρακτική δείχνει ενιαία και συνεπή ροή εγγραφής δεδομένων σε όλα τα υποσυστήματα.

3.5.6 Σύνοψη

Το επίπεδο δεδομένων συνδυάζει:

- σχεσιακή αξιοπιστία (PostgreSQL),
- χαρτογράφηση ORM και παραγωγή SQL (Hibernate),
- πρόσβαση μέσω repositories (Spring Data JPA),
- ασφαλή συναλλακτική ενορχήστρωση επιχειρησιακής λογικής (service layer).

Ως αποτέλεσμα, το σύστημα επιτυγχάνει συνέπεια, συντηρησιμότητα και επεκτασιμότητα για ροές ιατρικής πληροφόρησης προσαρμοσμένες στο προφίλ χρήστη.

3.6 Υλοποίηση τοπικοποίησης (Localization)

3.6.1 Στόχος και προσέγγιση i18n

Η τοπικοποίηση στη πλατφόρμα υλοποιείται με τον κλασικό μηχανισμό i18n του Spring (message bundles) σε συνδυασμό με Thymeleaf templates.

Ο στόχος είναι η ίδια λειτουργικότητα και δομή σελίδων να αποδίδεται σε διαφορετική γλώσσα, χωρίς αλλαγές στη λογική εφαρμογής.

3.6.2 Message bundles και οργάνωση μεταφράσεων

Οι μεταφράσεις εξωτερικοποιούνται σε αρχεία properties, ώστε το κείμενο της διεπαφής να μην είναι hardcoded στα templates ή στον κώδικα.

Η εφαρμογή ρυθμίζει ως basename το messages, οπότε γίνεται αυτόματη επίλυση π.χ. messages.properties και messages_el.properties.

```
# Internationalization Configuration
spring.messages.basename=messages
spring.messages.encoding=UTF-8
spring.web.locale=en
```

Με αυτό το μοντέλο:

- η συντήρηση μεταφράσεων γίνεται κεντρικά,

- η προσθήκη νέας γλώσσας γίνεται κυρίως με νέο bundle.

3.6.3 Locale resolution και αλλαγή γλώσσας μέσω παραμέτρου

Η ενεργή γλώσσα καθορίζεται από LocaleResolver και μπορεί να αλλάζει μέσω LocaleChangeInterceptor με παράμετρο lang.

Η ρύθμιση γίνεται σε επίπεδο MVC config.

```
@Bean
public LocaleResolver localeResolver() {
   SessionLocaleResolver localeResolver = new SessionLocaleResolver();
   localeResolver.setDefaultLocale(Locale.ENGLISH);
   return localeResolver;
}
@Bean
public LocaleChangeInterceptor localeChangeInterceptor() {
   LocaleChangeInterceptor interceptor = new LocaleChangeInterceptor();
   interceptor.setParamName("lang");
   interceptor.setIgnoreInvalidLocale(true);
   return interceptor;
}
@Override
public void addInterceptors(InterceptorRegistry registry) {
   registry.addInterceptor(localeChangeInterceptor()).addPathPatterns("/**");
}
```

Έτσι, ένα αίτημα όπως ?lang=el ενεργοποιεί την Ελληνική γλώσσα για τη συνεδρία.

3.6.4 Ενσωμάτωση μεταφράσεων στα templates (Thymeleaf)

Στα templates, το UI κείμενο αποδίδεται μέσω message keys με σύνταξη #{...}.

Αυτό διατηρεί το HTML σταθερό και μεταβάλλει μόνο το περιεχόμενο ανά locale.

```
<title th:text="#{profile.title}">My Profile - MedInfo</title>
```

```
<h5 th:text="#{profile.basic_info.title}">Basic Information</h5>
<label class="form-label" th:text="#{profile.basic_info.age}">Age</label>
```

Με τον τρόπο αυτό, οι ίδιες σελίδες λειτουργούν δίγλωσσα χωρίς διπλά templates.

3.6.5 Τοπικοποίηση δυναμικών πεδίων (παράδειγμα κατηγοριών)

Πέρα από στατικά labels, το σύστημα υποστηρίζει και μεταφράσεις δυναμικών τιμών (π.χ. κατηγορίες φαρμάκων), μετατρέποντάς τες σε message keys και επιλύοντάς τες μέσω MessageSource.

```
private String translateCategory(String category) {
    if (category == null || category.trim().isEmpty()) {
        return category;
    }
    String messageKey = "category." + category.toLowerCase().replace(" ", "_");
    Locale locale = LocaleContextHolder.getLocale();
    try {
        return messageSource.getMessage(messageKey, null, category, locale);
    } catch (Exception e) {
        return category;
    }
}
```

Αυτό επιτρέπει συνεπή εμφάνιση κατηγοριών σε κάθε γλώσσα, χωρίς αλλαγή της αποθηκευμένης τιμής στη βάση.

3.6.6 Σύνοψη

Η τοπικοποίηση στη πλατφόρμα βασίζεται σε:

- message bundles (messages*.properties) με εξωτερικοποιημένες μεταφράσεις,
- locale resolution σε επίπεδο εφαρμογής (session-based),
- αλλαγή γλώσσας μέσω παραμέτρου lang,
- χρήση message keys στα Thymeleaf templates,
- υποστήριξη μετάφρασης και για δυναμικές τιμές (όπου απαιτείται).

Το αποτέλεσμα είναι μια καθαρή και επεκτάσιμη i18n υλοποίηση, κατάλληλη για δίγλωσση διεπαφή και μελλοντική προσθήκη επιπλέον γλωσσών.

4 Παρουσίαση πλατφόρμας και λειτουργικότητα

Το κεφάλαιο αυτό παρουσιάζει τη λειτουργία της πλατφόρμας από την οπτική του τελικού χρήστη, μέσω αντιπροσωπευτικών οθονών της πλατφόρμας. Η παρουσίαση οργανώνεται σε βασικές ροές χρήσης: αρχική πλοήγηση, διαχείριση προφίλ και αναζήτηση φαρμάκων.

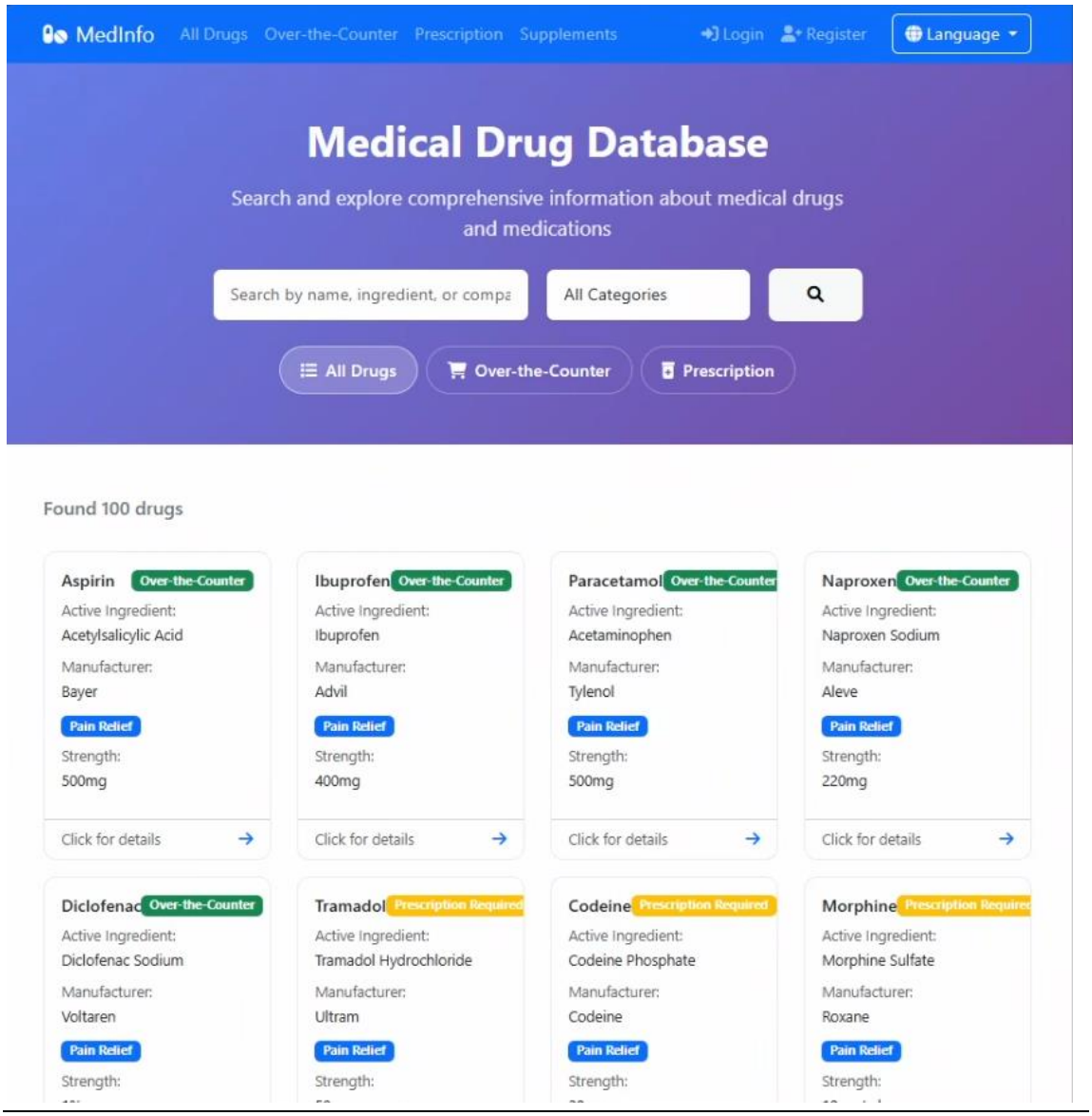
4.1 Αρχική σελίδα

Η αρχική εμπειρία του χρήστη εστιάζει στη γρήγορη πρόσβαση στην κύρια λειτουργικότητα της πλατφόρμας.

Η σελίδα παρέχει:

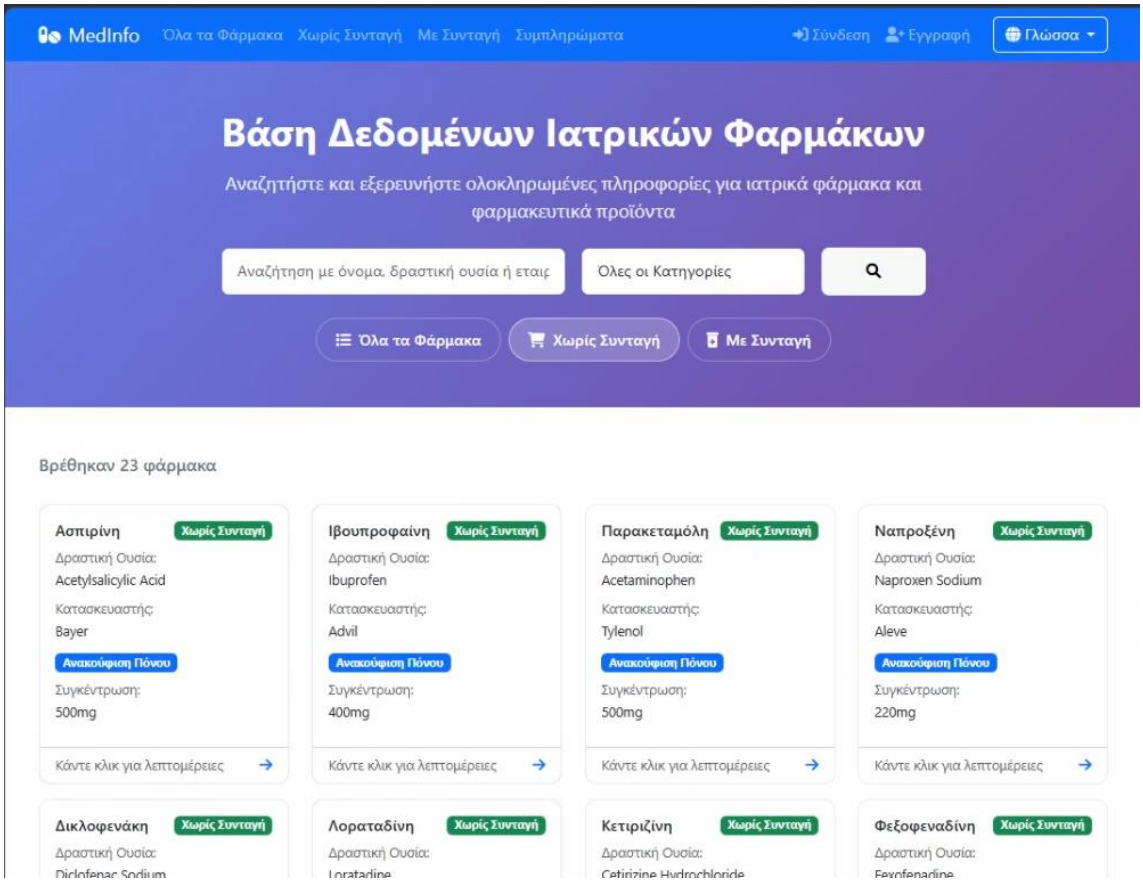
- κεντρική πλοήγηση σε βασικές ενότητες,
- άμεση πρόσβαση στην αναζήτηση,
- σαφή και συνεπή διεπαφή σε υποστηριζόμενες γλώσσες.

Στο σχήμα 4.1 παρουσιάζεται η αρχική σελίδα της πλατφόρμας, η οποία είναι σχεδιασμένη με γνώμονα την ευχρηστία. Στο κέντρο κυριαρχεί η μπάρα αναζήτησης (search bar) για την άμεση εύρεση φαρμακευτικών σκευασμάτων ή δραστικών ουσιών, ενώ στο επάνω μέρος βρίσκεται το μενού πλοήγησης για τη μετάβαση στις επιμέρους λειτουργίες της εφαρμογής.



Σχήμα .4.1: Αρχική προβολή της πλατφόρμας.

Το σχήμα 4.2 απεικονίζει τη βάση δεδομένων έπειτα από την εφαρμογή του φίλτρου για "Μη Συνταγογραφούμενα Φάρμακα" (ΜΗΣΥΦΑ). Το σύστημα απομονώνει και εμφανίζει στον χρήστη μόνο τα σκευάσματα που μπορούν να χορηγηθούν χωρίς την προσκόμιση ιατρικής συνταγής.



Σχήμα 4.2: Εφαρμογή γρήγορου φίλτρου/πλοήγησης από την αρχική οθόνη.

4.2 Σελίδα προφίλ χρήστη

Η σελίδα προφίλ αποτελεί το κέντρο εξατομίκευσης της εφαρμογής. Ο χρήστης μπορεί να δει και να ενημερώσει προσωπικά/ιατρικά δεδομένα που επηρεάζουν τα αποτελέσματα προτάσεων.

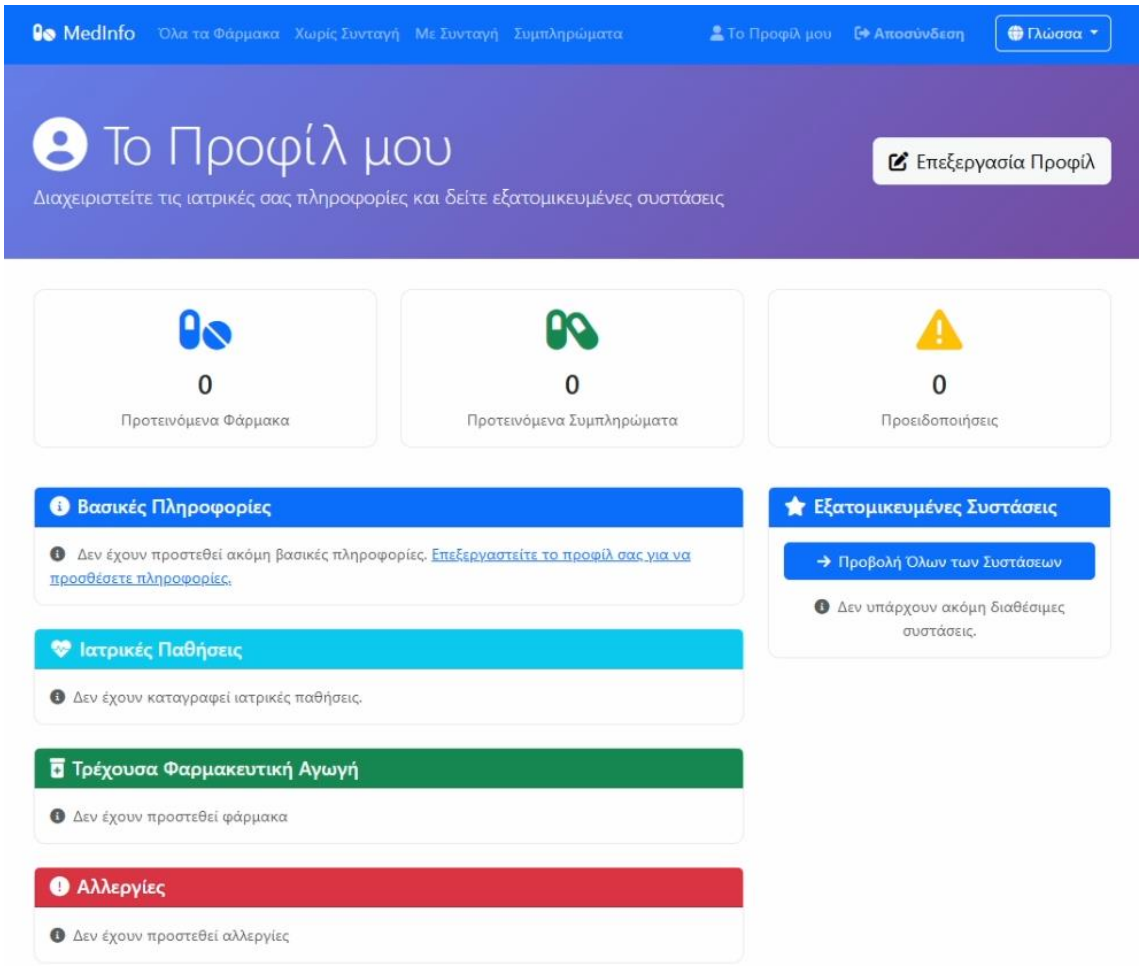
Η ενότητα περιλαμβάνει:

- βασικά στοιχεία προφίλ,
- ιατρικές καταστάσεις,
- τρέχουσα φαρμακευτική αγωγή,
- αλλεργίες,
- συνοπτικούς δείκτες προτεινόμενων αποτελεσμάτων και προειδοποιήσεων.

Στο σχήμα 4.3 παρουσιάζεται η σελίδα σύνδεσης της εφαρμογής. Μέσω αυτής της διεπαφής πραγματοποιείται η ταυτοποίηση των χρηστών. Ο χρήστης καλείται να συμπληρώσει τα διαπιστευτήριά του, δηλαδή το όνομα χρήστη και τον κωδικό πρόσβασης (password) στα αντίστοιχα πεδία, προκειμένου να αποκτήσει εξουσιοδοτημένη πρόσβαση στις λειτουργίες της πλατφόρμας.

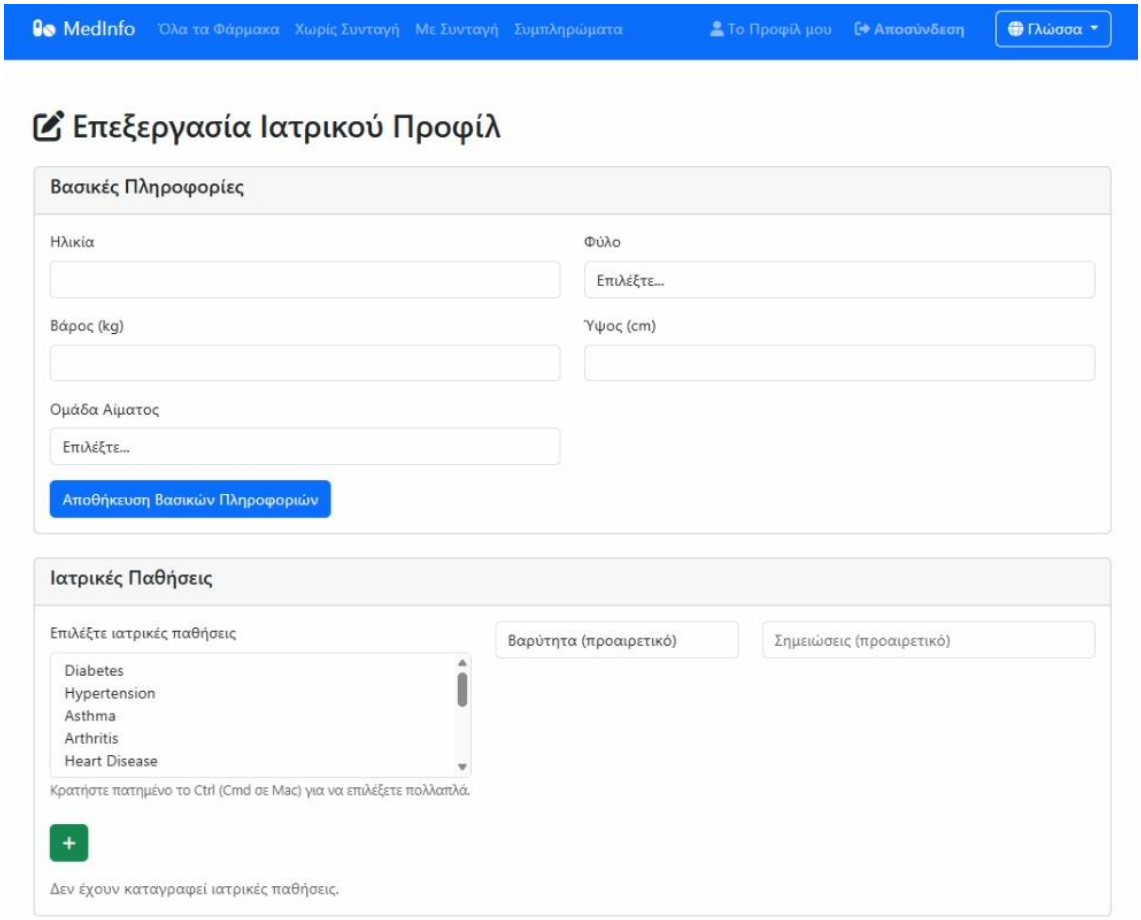
Σχήμα 4.3: Σελίδα σύνδεσης στο προσωπικό προφίλ.

Στο σχήμα 4.4 παρουσιάζεται η κεντρική οθόνη του προφίλ, η οποία λειτουργεί ως ένας εξατομικευμένος ιατρικός φάκελος του χρήστη. Στη σελίδα αυτή προβάλλεται αναλυτικά η τρέχουσα φαρμακευτική αγωγή που λαμβάνει ο χρήστης, καθώς και μια λίστα με προτεινόμενα φαρμακευτικά σκευάσματα. Επιπλέον, εμφανίζονται σημαντικές κλινικές πληροφορίες, όπως οι καταγεγραμμένες αλλεργίες και οι παθήσεις του χρήστη, προσφέροντας μια ολοκληρωμένη και άμεσα προσβάσιμη εικόνα του ιατρικού του ιστορικού



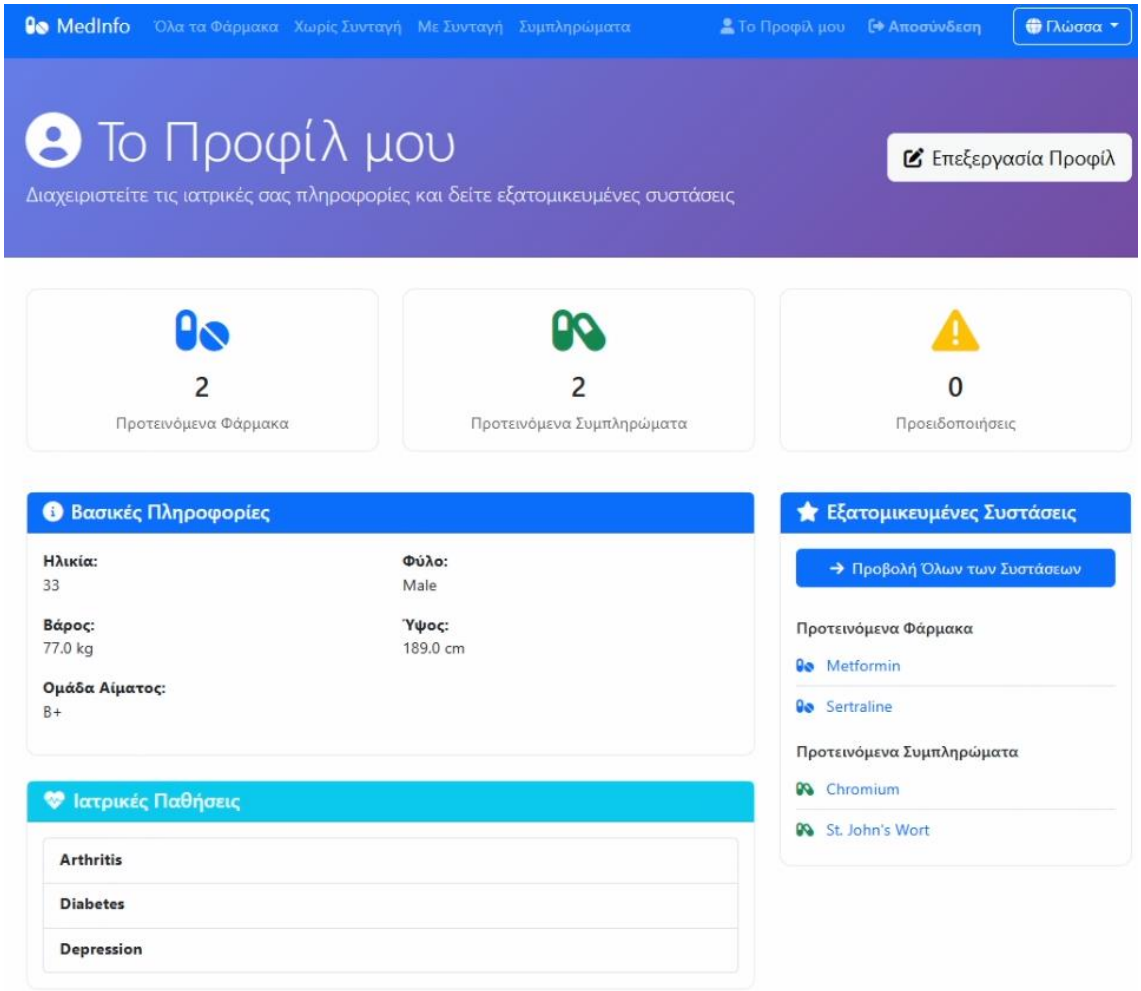
Σχήμα 4.4: Σελίδα προβολής προφίλ.

Το σχήμα 4.5 απεικονίζει τη σελίδα επεξεργασίας του προφίλ. Μέσω αυτής της διεπαφής, ο χρήστης έχει τη δυνατότητα να καταχωρήσει και να επικαιροποιήσει τα προσωπικά και ιατρικά του δεδομένα. Ειδικότερα, παρέχεται η επιλογή προσθήκης ή τροποποίησης των παθήσεων και των αλλεργιών. Παράλληλα, ο χρήστης καλείται να συμπληρώσει βασικά βιομετρικά και δημογραφικά στοιχεία, όπως το ύψος, το βάρος, την ομάδα αίματος και το φύλο του, εξασφαλίζοντας την ακρίβεια των δεδομένων στα οποία βασίζεται το σύστημα για την εξατομίκευση της εμπειρίας του.



Σχήμα 4.5: Σελίδα επεξεργασίας προφίλ.

Στο σχήμα 4.6 παρουσιάζεται η κεντρική σελίδα του προφίλ πλήρως ενημερωμένη με τα πραγματικά δεδομένα του χρήστη. Στη συγκεκριμένη απεικόνιση, διακρίνονται τα καταχωρημένα βιομετρικά στοιχεία, οι αλλεργίες και το ιστορικό των παθήσεων. Αξιοποιώντας αυτά τα εξατομικευμένα ιατρικά δεδομένα, το σύστημα προβάλλει στοχευμένα τη λίστα με τα προτεινόμενα φαρμακευτικά σκευάσματα (suggested medication), αναδεικνύοντας τη δυναμική λειτουργία της πλατφόρμας και την ικανότητά της να προσαρμόζεται στις ειδικές ανάγκες του εκάστοτε χρήστη.



Σχήμα 4.6: Παράδειγμα συμπληρωμένου προφίλ με ενεργά δεδομένα.

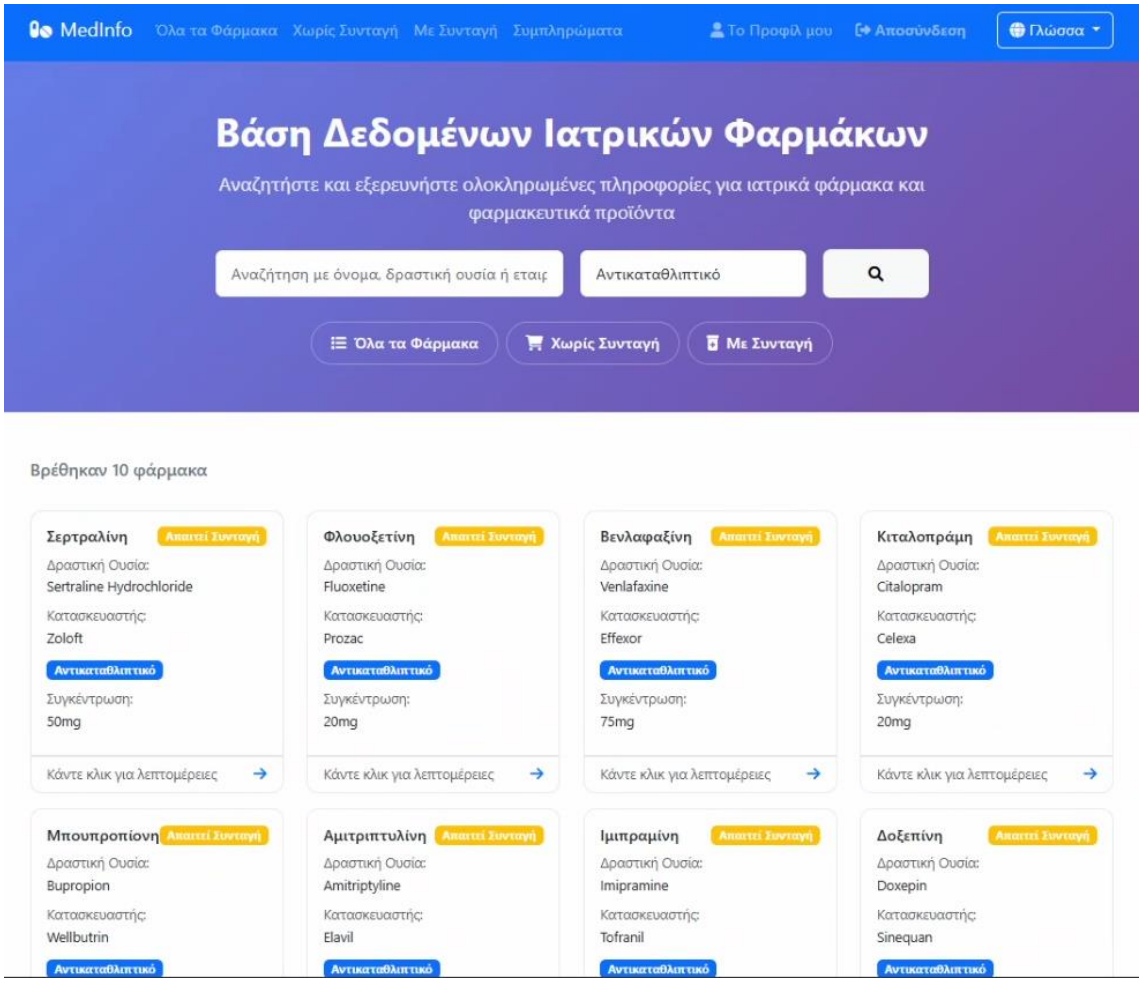
4.3 Αναζήτηση φαρμάκων

Η λειτουργία αναζήτησης επιτρέπει στον χρήστη να εντοπίζει φαρμακευτικές εγγραφές με βάση όρους αναζήτησης και φίλτρα.

Βασικές δυνατότητες:

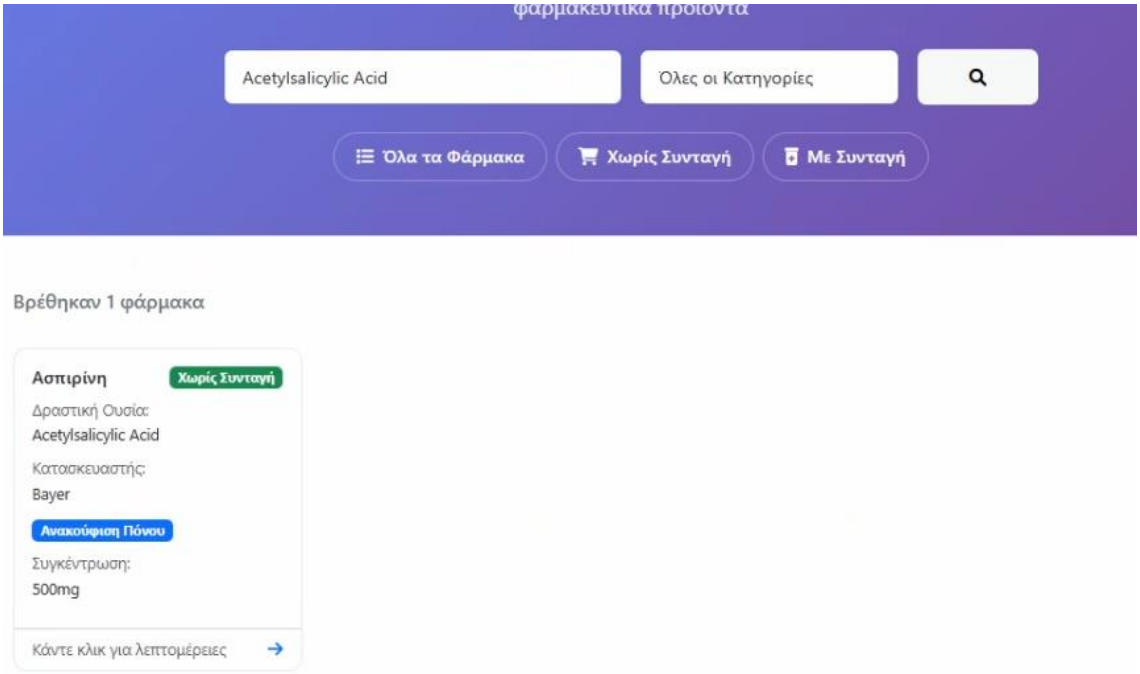
- αναζήτηση με λέξεις-κλειδιά,
- φιλτράρισμα αποτελεσμάτων,
- μετάβαση σε σελίδα λεπτομερειών φαρμάκου.

Στο σχήμα 4.7 παρουσιάζεται η διεπαφή της πλατφόρμας με επιλεγμένο το φίλτρο αναζήτησης για τα συνταγογραφούμενα φάρμακα. Με την εφαρμογή του συγκεκριμένου φίλτρου, η λίστα των αποτελεσμάτων ανανεώνεται δυναμικά, απομονώνοντας και προβάλλοντας αποκλειστικά τα φαρμακευτικά σκευάσματα για τη χορήγηση των οποίων απαιτείται η προσκόμιση ιατρικής συνταγής.



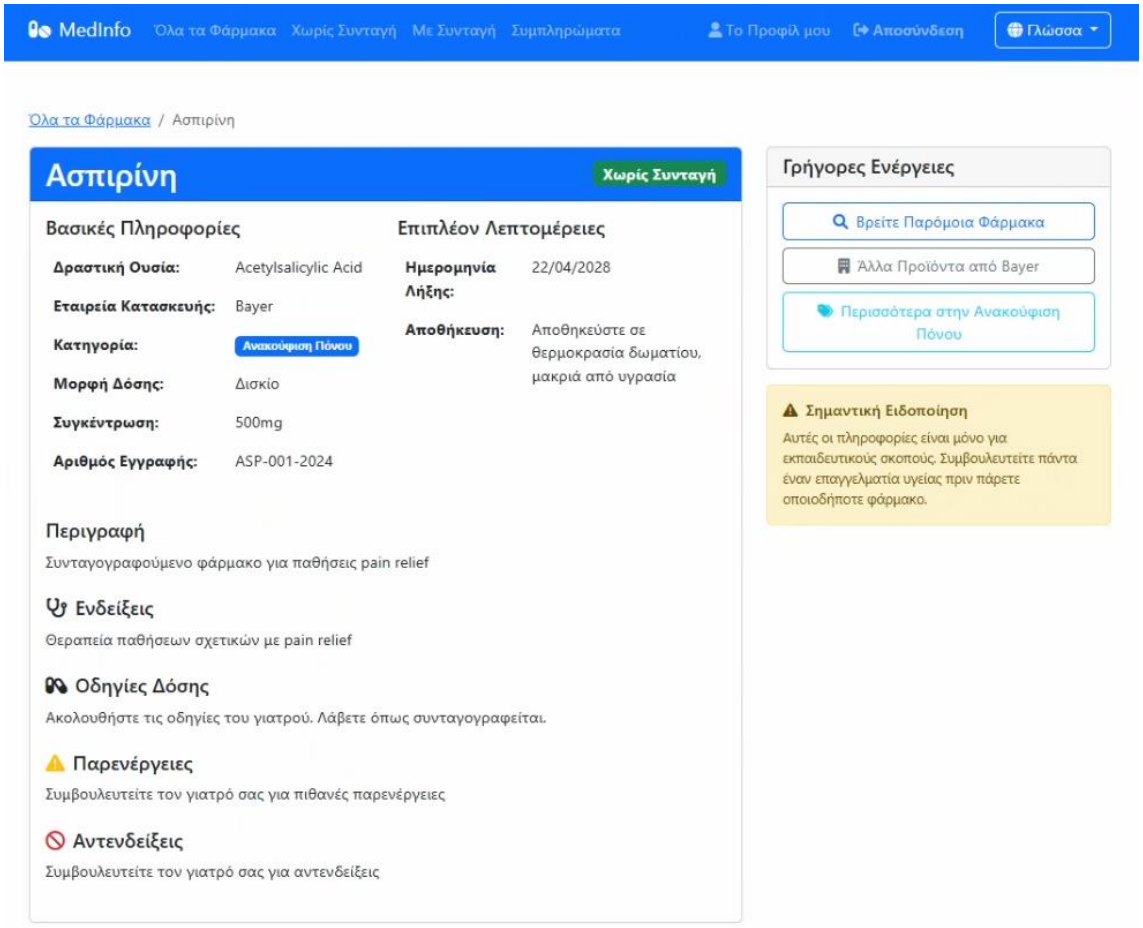
Σχήμα 4.7: Προβολή αναζήτησης με προεπιλεγμένα αποτελέσματα.

Στο σχήμα 4.8 παρουσιάζεται ένα πρακτικό παράδειγμα χρήσης της κεντρικής λειτουργίας αναζήτησης της πλατφόρμας. Στο συγκεκριμένο στιγμιότυπο, ο χρήστης έχει πληκτρολογήσει τον όρο «ακετυλοσαλικυλικό οξύ» (acetylsalicylic acid), τη δραστική ουσία που είναι ευρέως γνωστή στο κοινό ως ασπιρίνη. Το σύστημα επεξεργάζεται το ερώτημα και επιστρέφει άμεσα τα σχετικά αποτελέσματα, επιδεικνύοντας την ικανότητα της βάσης δεδομένων να εντοπίζει σκευάσματα με βάση τη φαρμακολογική τους σύσταση



Σχήμα 4.8: Αποτελέσματα μετά αναζήτηση δραστικής ουσίας..

Στο σχήμα 4.9 παρουσιάζεται η σελίδα αναλυτικών πληροφοριών ενός επιλεγμένου φαρμακευτικού σκευάσματος. Στη συγκεκριμένη διεπαφή, ο χρήστης αποκτά πρόσβαση σε ένα εκτενές σύνολο δεδομένων που αντλούνται απευθείας από τη βάση. Ειδικότερα, προβάλλονται κρίσιμα στοιχεία όπως η δραστική ουσία, η μορφή του φαρμάκου, η συνιστώμενη δοσολογία, οι ενδείξεις και αντενδείξεις, καθώς και οι πιθανές παρενέργειες. Η δομημένη αυτή παρουσίαση εξασφαλίζει την πλήρη, οργανωμένη και ασφαλή ενημέρωση του χρήστη για το εκάστοτε φάρμακο.



Σχήμα 4.9: Σελίδα λεπτομερειών επιλεγμένου φαρμάκου.

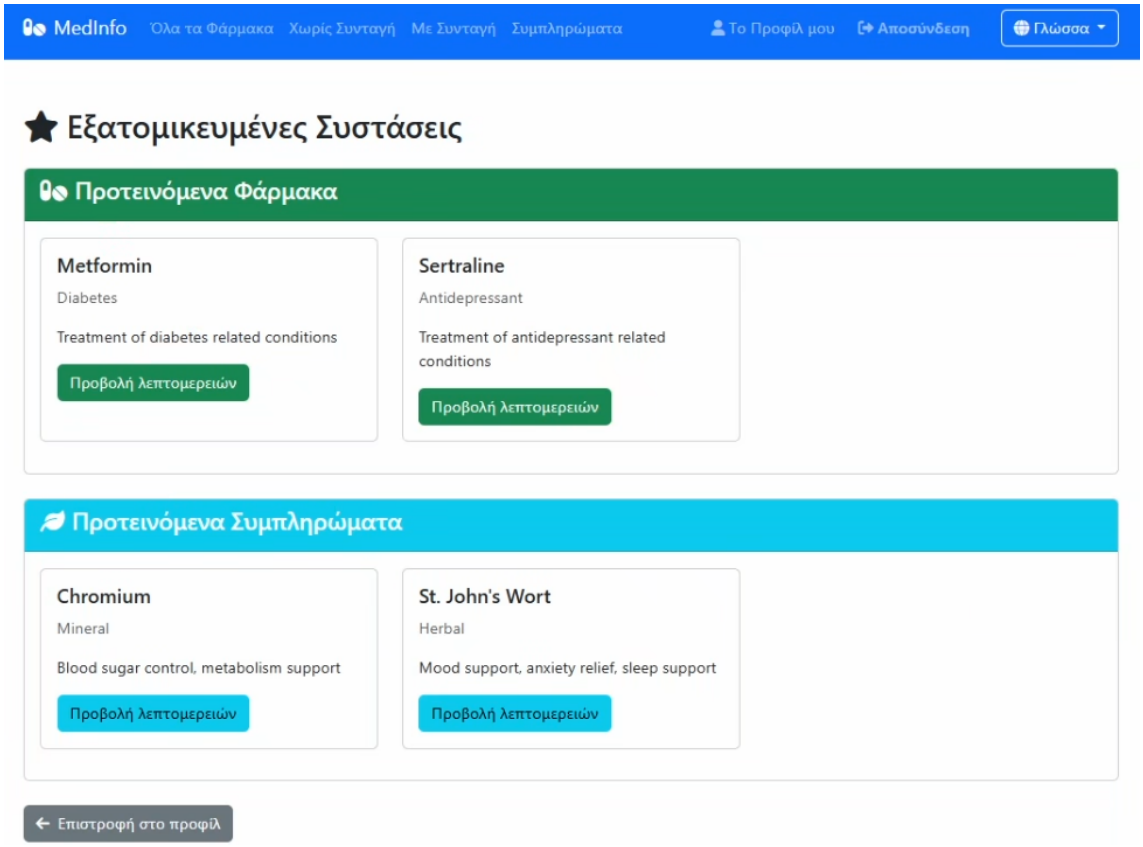
4.4 Εξατομικευμένες προτάσεις

Η ενότητα προτάσεων παρουσιάζει αποτελέσματα που προκύπτουν από τα δεδομένα προφίλ του χρήστη, ενισχύοντας τον πρακτικό χαρακτήρα της πλατφόρμας.

Παρουσιάζονται:

- προτεινόμενα φάρμακα/συμπληρώματα,
- προειδοποιήσεις σχετικές με αλληλεπιδράσεις ή αλλεργίες,
- κατάσταση χωρίς διαθέσιμες προτάσεις (empty state), όταν λείπουν κρίσιμα δεδομένα προφίλ.

Στο σχήμα 4.10 παρουσιάζεται η σελίδα εξατομικευμένων προτάσεων για τον συνδεδεμένο χρήστη. Αξιοποιώντας τα καταχωρημένα δεδομένα του ιατρικού ιστορικού και του προφίλ του, το σύστημα παράγει και προβάλλει στοχευμένες συστάσεις. Στη συγκεκριμένη οθόνη, η πλατφόρμα δεν περιορίζεται μόνο στην υπόδειξη προτεινόμενων φαρμακευτικών σκευασμάτων, αλλά επεκτείνεται και στην προβολή κατάλληλων συμπληρωμάτων διατροφής, προσφέροντας έτσι μια ολοκληρωμένη και προσωποποιημένη προσέγγιση στις ανάγκες του χρήστη.



Σχήμα 4.10: Σελίδα εξατομικευμένων προτάσεων.

5 Βελτιώσεις και μελλοντικές επεκτάσεις

Η πλατφόρμα καλύπτει τους βασικούς στόχους της εργασίας, ωστόσο υπάρχουν συγκεκριμένες κατευθύνσεις που μπορούν να ενισχύσουν τη λειτουργικότητα, την ασφάλεια και την παραγωγική ωριμότητα του συστήματος.

Βελτίωση λογικής προτάσεων

- Εμπλουτισμός της recommendation λογικής με μηχανισμό βαθμολόγησης/κατάταξης (ranking) αντί απλής αντιστοίχισης.
- Παροχή επεξήγησης ανά πρόταση (explainability), ώστε ο χρήστης να γνωρίζει γιατί εμφανίζεται κάθε αποτέλεσμα.
-

Ενίσχυση ελέγχων ασφάλειας

- Επέκταση των ελέγχων αλληλεπιδράσεων φαρμάκων με πιο εξειδικευμένες πηγές/κανόνες.
- Βελτίωση policy-level προστασίας σε authentication endpoints (π.χ. rate limiting και επιπλέον hardening).

Εξέλιξη αρχιτεκτονικής και λειτουργικότητας

- Προσθήκη REST API επιπέδου για υποστήριξη mobile clients ή εξωτερικών συστημάτων.
- Επέκταση του backend ώστε να υποστηρίζει μελλοντικές ενσωματώσεις με τρίτα συστήματα υγείας.

Βελτιώσεις ποιότητας λογισμικού

- Επέκταση αυτοματοποιημένων δοκιμών (integration και end-to-end) για κρίσιμες ροές χρήστη.
- Ενσωμάτωση CI pipeline για συστηματικό build/test validation πριν από κάθε release.

Επέκταση τοπικοποίησης και προσβασιμότητας

- Προσθήκη επιπλέον γλωσσών πέρα από Αγγλικά/Ελληνικά με την ίδια i18n υποδομή.
- Βελτιώσεις προσβασιμότητας (σαφέστερη πλοήγηση με πληκτρολόγιο, semantic labels, βελτιωμένη αναγνωσιμότητα).

Οι παραπάνω επεκτάσεις μπορούν να υλοποιηθούν σταδιακά πάνω στην υφιστάμενη αρχιτεκτονική, διατηρώντας το σύστημα συντηρήσιμο και επεκτάσιμο, ενώ αυξάνουν την πρακτική του αξία για πραγματικά σενάρια χρήσης.

Βιβλιογραφία

[1] Pressman, R. S. and Maxim, B. R., Software Engineering: A Practitioner’s Approach, McGraw-Hill.

[2] Ricci, F., Rokach, L. and Shapira, B., Recommender Systems Handbook, Springer.

[3] Shortliffe, E. H. and Cimino, J. J. , Biomedical Informatics: Computer Applications in Health Care and Biomedicine, Springer.

[4] Stallings, W. and Brown, L. , Computer Security: Principles and Practice, Pearson.

[5] Alepis E., "Software Measures for Common Design Patterns Using Visual Studio Code Metrics," in *DOI:10.1109/IISA.2018.8633694*, July 2018.

[6] Alepis E. and Virvou M., Object-Oriented User Interfaces for Personalized Mobile Learning, Springer, 2014.

[7] I. Sommerville, Software Engineering, Pearson.

[8] Achilleas Papageorgiou, Michael Strigkos, Eugenia A. Politou, Efthimios Alepis, Agusti Solanas and Constantinos Patsakis, "Security and Privacy Analysis of Mobile Health Applications: The Alarming State of Practice," *IEEE Access 6: 9390-9403,* 2018.

[9] Andreas Karavokyris, Efthimios Alepis and Software Measures for Common Design Patterns Using, "Software Measures for Common Design Patterns Using Visual Studio Code Metrics," *IISA 2018: 1-7.*

[10] Dimitrios P. Panagoulias, Maria Virvou and George A. Tsihrintzis, "nuhealthsoft: A Nutritional and Health Data Processing Software Tool from a patient's perspective," *SITIS 2022: 386-393.*

[11] Efthymios Alepis and Maria Virvou, "Object oriented architecture for affective multimodal e-learning interfaces," *Intell. Decis. Technol. 4(3): 171-180,* 2010.

[12] Eugenia A. Politou, Alexandra K. Michota, Efthimios Alepis, Matthias Pocs and Constantinos Patsakis, "Backups and the right to be forgotten in the GDPR: An uneasy relationship," *Comput. Law Secur. Rev. 34(6): 1247-1257 ,* 2018.

[13] Eugenia A. Politou, Efthimios Alepis and Constantinos Patsakis, "Forgetting personal data and revoking consent under the GDPR: Challenges and proposed solutions," *J. Cybersecur. 4(1): tyy001,* 2018.

[14] Ioannis Hatzilygeroudis, George A. Tsihrintzis, Maria Virvou and Isidoros Perikos, "Special issue on information, intelligence, systems and applications," *Neural Comput. Appl. 35(1): 1-2 ,* 2023.

[15] Panagiotis Pavlakis, Efthymios Alepis and Maria Virvou, "Intelligent Mobile Multimedia Application for the Support of the Elderly," *IIH-MSP 2012: 297-300.*

Αναφορές

Java (JDK) – https://www.oracle.com/java/
Spring Boot – https://spring.io/projects/spring-boot
Spring Framework – https://spring.io/projects/spring-framework
Spring Security – https://spring.io/projects/spring-security
Spring Data JPA – https://spring.io/projects/spring-data-jpa
Hibernate ORM – https://hibernate.org/orm/
Thymeleaf – https://www.thymeleaf.org/
PostgreSQL – https://www.postgresql.org/
IntelliJ IDEA – https://www.jetbrains.com/idea/