



**ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΩΝ
ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΕΠΙΚΟΙΝΩΝΙΩΝ ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ**

Πτυχιακή Εργασία

Τίτλος Πτυχιακής Εργασίας	Armoire: Διαδικτυακή Εφαρμογή Καταλογοποίησης Γκαρνταρόμπας και Προτάσεων Ρουχισμού και Εμφάνισεων (Αγγλικά) Armoire: A Web Application for Wardrobe Cataloging and Outfit Planning
Ονοματεπώνυμο Φοιτητή	Σταματίνα Σαουάχ
Πατρώνυμο	Καμίλ-Γεώργιος
Αριθμός Μητρώου	Π21149
Επιβλέπων	Κωνσταντίνα Χρυσafiάδη, Επίκουρος Καθηγήτρια

Ημερομηνία Παράδοσης

Σεπτέμβριος 2026

Copyright ©

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.

Οι απόψεις και τα συμπεράσματα που περιέχονται σε αυτό το έγγραφο εκφράζουν αποκλειστικά τον συγγραφέα και δεν αντιπροσωπεύουν τις επίσημες θέσεις του Πανεπιστημίου Πειραιώς.

Ως συγγραφέας της παρούσας εργασίας δηλώνω πως η παρούσα εργασία δεν αποτελεί προϊόν λογοκλοπής και δεν περιέχει υλικό από μη αναφερόμενες πηγές.

Ευχαριστίες

Θα ήθελα να ευχαριστήσω την επιβλέπουσα καθηγήτριά μου, Κωνσταντίνα Χρυσafiάδη, για την καθοδήγηση και τη στήριξή της σε όλη τη διάρκεια εκπόνησης αυτής της πτυχιακής εργασίας. Ευχαριστώ επίσης τους διδάσκοντες του Τμήματος Πληροφορικής του Πανεπιστημίου Πειραιώς για τις γνώσεις που μου έδωσαν στη διάρκεια των σπουδών μου. Τέλος, ευχαριστώ τους φίλους μου για την υπομονή και τη στήριξή τους, καθώς και για τη συμβολή τους στη βελτίωση της εφαρμογής.

Περίληψη

Η παρούσα πτυχιακή εργασία παρουσιάζει τον σχεδιασμό και την υλοποίηση της εφαρμογής Armoire, μιας διαδικτυακής εφαρμογής για την καταλογοποίηση ρούχων και τον σχεδιασμό εμφανίσεων (outfits). Απευθύνεται σε χρήστες που θέλουν να οργανώσουν ψηφιακά τη γκαρνταρόμπα τους, να φτιάχνουν εμφανίσεις για συγκεκριμένες περιστάσεις, και να λαμβάνουν προτάσεις εμφανίσεων.

Το σύστημα αποτελείται από τρία επίπεδα: ένα frontend σε React και TypeScript, ένα REST API σε Spring Boot (Java 21) και μια βάση δεδομένων MySQL. Ο μηχανισμός προτάσεων λειτουργεί σε τρία βήματα: πρώτα φιλτράρει τα ρούχα ανάλογα με την περίπτωση, μετά ελέγχει αν οι συνδυασμοί ταιριάζουν χρωματικά, και τέλος βαθμολογεί τους έγκυρους συνδυασμούς με βάση τις προτιμήσεις που έχει μάθει από τις προηγούμενες επιλογές του χρήστη. Η εφαρμογή εμφανίζει επίσης υπενθυμίσεις για ρούχα που δεν έχουν φορεθεί και προτείνει νέα ρούχα για να καλυφθούν κενά στη γκαρνταρόμπα.

Για την εκπόνηση της εργασίας έγινε βιβλιογραφική ανασκόπηση, καταγράφηκαν οι απαιτήσεις του συστήματος, σχεδιάστηκε η αρχιτεκτονική και η βάση δεδομένων, και τεκμηριώνεται η λειτουργία της εφαρμογής με παραδείγματα χρήσης και εγχειρίδιο χρήστη. Στο τέλος συζητούνται τα οφέλη της λύσης, οι περιορισμοί της και οι μελλοντικές επεκτάσεις, με έμφαση στην αναγνώριση ρούχων από εικόνα.

Επιστημονική περιοχή: Ανάπτυξη Λογισμικού / Διαδικτυακές Εφαρμογές / Συστήματα Συστάσεων.

Λέξεις Κλειδιά: γκαρνταρόμπα, σχεδιασμός εμφανίσεων, σύστημα συστάσεων, χρωματική αρμονία, διαδικτυακή εφαρμογή

Abstract

This dissertation presents the design and implementation of Armoire, a web application for cataloging clothing items and planning outfits. The application is aimed at users who want to digitally organize their wardrobe, assemble outfits for specific occasions, and receive outfit suggestions.

The system was built as a three-tier application, with a React and TypeScript client, a Spring Boot (Java 21) REST API, and a MySQL database. The suggestion engine combines strict occasion-based filtering, a color-harmony hard filter, and score boosts derived from cosine similarity and Jaccard similarity over the history of saved and worn outfits. In addition, the application displays reminders for unworn items and purchase suggestions to close gaps in the user's wardrobe.

The dissertation includes a literature review of related scientific work, a specification of system requirements, the design of the architecture and database, and documentation of the application's operation through usage examples and a user manual. Finally, the expected benefits, limitations, and directions for future work are discussed, with emphasis on image-based clothing recognition.

Scientific area: Software Development / Web Applications / Recommender Systems.

Key Words: wardrobe management, outfit planning, recommender system, color harmony, web application

Πίνακας Περιεχομένων

Copyright ©	ii
Ευχαριστίες.....	iii
Περίληψη	iv
Abstract.....	iv
Πίνακας Περιεχομένων	v
Κατάλογος Εικόνων	ix
Κατάλογος Πινάκων	xii
1. Εισαγωγή	1
1.1 Κίνητρο και Πλαίσιο Προβλήματος	1
1.2 Συνεισφορά	1
1.3 Μεθοδολογία και Τεχνική Προσέγγιση	1
1.4 Δομή της Πτυχιακής	1
2. Στόχοι της Πτυχιακής	2
2.1 Γενικός Στόχος.....	2
2.2 Ειδικοί Στόχοι	2
2.2.1 Λειτουργικοί Στόχοι.....	2
2.2.2 Τεχνικοί Στόχοι	2
2.3 Περιορισμοί – Παραδοχές (Scope)	2
3. Βιβλιογραφική Ανασκόπηση.....	3
3.1 Ολοκληρωμένα Συστήματα	3
3.2 Διαδικτυακές Εφαρμογές (Web-Based Applications)	3
3.3 Έξυπνες Εφαρμογές/Συστήματα (Smart Application/System).....	4
3.4 Έξυπνες Εφαρμογές για την Οργάνωση Ντουλάπας και Ρουχισμού	4
3.5 Ευφυείς Εφαρμογές και Τεχνητή Νοημοσύνη (Intelligent Systems & AI)	6
3.6 Εμπειρία Χρήστη (User Experience)	8
3.7 Προσαρμοστικότητα – Προσαρμοστικά Συστήματα (Adaptivity / Adaptive Systems).....	9
3.8 Αλληλεπίδραση Ανθρώπου-Υπολογιστή / Διεπαφές Χρήστη (HCI/UX)	13
3.9 Σύνοψη Κεφαλαίου.....	15
4. Περιγραφή Προβλήματος	16
5. Αναφορά σε Παρόμοιες Εφαρμογές.....	17
6. Ανάλυση Απαιτήσεων	22
6.1 Ρόλοι Χρηστών και Λειτουργίες (Use-Case Diagram).....	22
6.2 Λειτουργικές και Μη Λειτουργικές Απαιτήσεις.....	23
6.3 Περιορισμοί και Τεχνικές Προϋποθέσεις Εγκατάστασης/Χρήσης.....	24

7. Σχεδιασμός και Υλοποίηση	26
7.1 Αρχιτεκτονική του Συστήματος.....	26
7.2 Σχεδιασμός Βάσης Δεδομένων	27
7.2.1 Πίνακας users	27
7.2.2 Πίνακας clothing_items και πίνακες σύνδεσης.....	28
7.2.3 Πίνακες outfits και outfit_items	29
7.2.4 Διάγραμμα Οντοτήτων-Σχέσεων	30
7.3 Αντιστοίχιση Οντοτήτων με JPA/Hibernate	30
7.4 Αυθεντικοποίηση και Ασφάλεια.....	32
7.4.1 Κατακερματισμός Κωδικού στον Πελάτη	32
7.4.2 Επαλήθευση στον Διακομιστή	33
7.4.3 Έκδοση και Επαλήθευση JWT.....	33
7.4.4 Διαμόρφωση Spring Security.....	34
7.5 Διαχείριση Εικόνων	34
7.5.1 Αποθήκευση με Μοναδικά Ονόματα	34
7.5.2 Επικύρωση Τύπου Εικόνας	35
7.5.3 Προστασία Path Traversal.....	36
7.6 Επικύρωση Ρούχων κατά την Εισαγωγή	36
7.7 Κανόνες Σύνθεσης Συνόλου (Outfit Composition)	37
7.8 Αλυσιδωτή Καταγραφή Χρήσης (Cascading Wear Recording)	39
7.9 Μηχανή Πρότασης Συνόλων — Pipeline Πέντε Φάσεων	40
7.10 Κανόνες Χρωματικής Αρμονίας.....	41
7.11 Σύστημα Βαθμολόγησης Συνδυασμών.....	42
7.12 Εκμάθηση Προτιμήσεων Χρήστη.....	44
7.12.1 Feature Vectors	45
7.13 Cosine Similarity — Ταίριασμα Στυλ	46
7.14 Jaccard Similarity — Ομοιότητα με Αποθηκευμένα Σύνολα.....	47
7.15 Υπενθυμίσεις και Αξιολογήσεις Ντουλάπας (Wardrobe Insights).....	47
7.16 Διαγνωστικό Πλαίσιο Αδύνατων Προτάσεων (Suggestion Context).....	48
7.17 Προτάσεις Αγορών (Purchase Suggestions)	49
7.18 Διαχείριση Ειδικών Περιπτώσεων (Edge Case Handling)	50
7.19 Εμφάνιση Αποτελεσμάτων στον Χρήστη — Κάρτα Πρότασης.....	51
7.20 Εμφάνιση Αδύνατων Προτάσεων — Διαγνωστικό Περιβάλλον.....	51
7.21 Dashboard Αναλυτικών Στοιχείων	51
7.22 Δυναμικές Συμβουλές και Προσωπικό Εισαγωγικό Μήνυμα	52
7.23 Κατάλογος REST API Endpoints	53

7.24 Δομή Frontend (React + TypeScript)	54
8. Σενάρια Χρήσης και Ανάδραση Συστήματος.....	55
8.1 Σενάριο Α: Η Μαρία — Εργαζόμενη με πλούσια ντουλάπα	55
8.1.1 Προφίλ και Ντουλάπα	55
8.1.2 Αίτημα Πρότασης για Δουλειά (WORK)	57
8.1.3 Αποθήκευση και Καταγραφή Χρήσης	59
8.1.4 Dashboard της Μαρίας.....	60
8.1.5 Υπενθυμίσεις Ντουλάπας.....	61
8.1.6 Αίτημα Πρότασης για Casual — Διαφοροποίηση	62
8.2 Σενάριο Β: Ο Γιάννης — Νέος χρήστης με μικρή ντουλάπα	63
8.2.1 Προφίλ και Ντουλάπα	63
8.2.2 Αίτημα Πρότασης για Casual (CASUAL)	65
8.2.3 Αίτημα Πρότασης για Work — Χωρίς Ρούχα για Work	66
8.2.4 Dashboard του Γιάννη.....	68
8.3 Σύγκριση Σεναρίων.....	69
9. Εγχειρίδιο Χρήστη.....	71
9.1 Εισαγωγή.....	71
9.2 Αρχική Σελίδα.....	71
9.3 Εγγραφή Νέου Χρήστη.....	72
9.4 Σύνδεση.....	74
9.5 Dashboard (Πίνακας Ελέγχου)	75
9.6 Γκαρνταρόμπα — Προβολή Ρούχων	78
9.7 Προσθήκη Νέου Ρούχου	79
9.8 Επεξεργασία Ρούχου	82
9.9 Δημιουργία Συνόλου (Outfit) Χειροκίνητα	84
9.10 Λήψη Προτάσεων Εμφάνισης	87
9.11 Καταγραφή Χρήσης (Wear Tracking)	89
9.12 Υπενθυμίσεις και Αρχειοθέτηση μέσω Dashboard	90
9.13 Προτάσεις Αγορών (Purchase Suggestions)	90
9.14 Ρυθμίσεις Λογαριασμού	92
9.15 Αποσύνδεση.....	94
9.16 Γραμμή Πλοήγησης	94
10. Οφέλη που Αναμένουμε από την Προτεινόμενη Λύση	95
11. Μελλοντικές Επεκτάσεις και Βελτιώσεις.....	96
12. Συμπεράσματα.....	97
Πίνακας Ορολογίας	98

Πίνακας Συντμήσεων – Αρκτικόλεξων – Ακρωνυμίων	99
Βιβλιογραφία	100
Παραρτήματα	105
Παράρτημα Α — Scripts Μετάπτωσης Βάσης Δεδομένων (SQL)	106
Παράρτημα Β — Ενοποιημένη Αναφορά REST API	108

Κατάλογος Εικόνων

Εικόνα 3.1. Βρόχος ανατροφοδότησης: καταγραφή φοράς, μάθηση προτιμήσεων και υπενθυμίσεις στο Armoire.....	11
Εικόνα 5.1. Στιγμιότυπο της εφαρμογής Stylebook	17
Εικόνα 5.2. Στιγμιότυπο της εφαρμογής Cladwell.....	18
Εικόνα 5.3. Στιγμιότυπο της εφαρμογής Whering	19
Εικόνα 5.4. Στιγμιότυπο της εφαρμογής Smart Closet	20
Εικόνα 6.1. Διάγραμμα περιπτώσεων χρήσης (use-case) του Armoire	23
Εικόνα 7.1. Αρχιτεκτονική του συστήματος Armoire (τρία επίπεδα).....	27
Εικόνα 7.2. Κώδικας SQL δημιουργίας του πίνακα users	28
Εικόνα 7.3. Κώδικας SQL δημιουργίας του πίνακα clothing_items	28
Εικόνα 7.4. Πίνακες σύνδεσης εποχών και περιστάσεων	29
Εικόνα 7.5. Πίνακες outfits και outfit_items	29
Εικόνα 7.6. Διάγραμμα Οντοτήτων-Συσχετίσεων (ERD) της βάσης δεδομένων του Armoire.....	30
Εικόνα 7.7. JPA οντότητα ClothingItem: αντιστοίχιση @ElementCollection.....	31
Εικόνα 7.8. Διάγραμμα ακολουθίας πιστοποίησης (σύνδεση με JWT)	32
Εικόνα 7.9. Κατακερματισμός SHA-256 του συνδυασμού email και κωδικού (CredentialHasher).....	33
Εικόνα 7.10. Μέθοδος passwordMatches: επαλήθευση κωδικού στον διακομιστή	33
Εικόνα 7.11. Μέθοδος generateAccessToken (JwtService)	34
Εικόνα 7.12. SecurityConfig: κανόνες πρόσβασης	34
Εικόνα 7.13. Μέθοδος store: αποθήκευση εικόνας με όνομα UUID	35
Εικόνα 7.14. Επικύρωση τύπου εικόνας (validateImage)	35
Εικόνα 7.15. Προστασία από path traversal (resolvePath).....	36
Εικόνα 7.16. Επικύρωση στοιχείων ρούχου κατά την εισαγωγή (validateCreateRequest)	36
Εικόνα 7.17. Κανόνες σύνθεσης συνόλου (validateOutfitComposition)	38
Εικόνα 7.18. Διάγραμμα ροής επικύρωσης συνόλου	39
Εικόνα 7.19. Αλυσιδωτή καταγραφή χρήσης (recordWear)	40
Εικόνα 7.20. Μηχανή πρότασης: τμήμα του pipeline πέντε φάσεων (suggest)	41
Εικόνα 7.21. Κανόνες χρωματικής αρμονίας (ColorHarmonyRules)	42
Εικόνα 7.22. Μέθοδος scoreCombination: υπολογισμός βαθμολογίας	44
Εικόνα 7.23. Κατασκευή προφίλ προτιμήσεων (buildProfile).....	45
Εικόνα 7.24. Δημιουργία διανύσματος χαρακτηριστικών ανά ρούχο (itemVector)	46
Εικόνα 7.25. Υπολογισμός ομοιότητας συνημιτόνου (cosine similarity)	46

Εικόνα 7.26. Υπολογισμός ομοιότητας Jaccard (savedOutfitJaccard).....	47
Εικόνα 7.27. Κατηγοριοποίηση υπενθυμίσεων (classifyNudge)	48
Εικόνα 7.28. Κατασκευή διαγνωστικού πλαισίου (SuggestionContextBuilder).....	49
Εικόνα 7.29. Αντιστοίχιση περιστασης-υποκατηγορίας (subcategoryFor).....	50
Εικόνα 7.30. Υπολογισμός αναλυτικών στοιχείων (buildWardrobeInsights).....	52
Εικόνα 8.1. Η γκαρνταρόμπα της Μαρίας με κάποια από τα 18 ρούχα.....	56
Εικόνα 8.2. Στατιστικά, κάλυψη κατηγοριών και δυναμικές συμβουλές της Μαρίας...	57
Εικόνα 8.3. Οι τρεις προτάσεις WORK για τη Μαρία	58
Εικόνα 8.4. Κουμπί «Save outfit» στην κάρτα πρότασης	60
Εικόνα 8.5. Dashboard της Μαρίας: εισαγωγικό μήνυμα και στατιστικά	61
Εικόνα 8.6. Οι τρεις προτάσεις CASUAL για τη Μαρία	62
Εικόνα 8.7. Η γκαρνταρόμπα του Γιάννη (4 ρούχα)	64
Εικόνα 8.8. Στατιστικά, κάλυψη κατηγοριών και συμβουλές του Γιάννη.....	64
Εικόνα 8.9. Οι δύο προτάσεις CASUAL για τον Γιάννη	65
Εικόνα 8.10. Μήνυμα «No work pieces in your wardrobe» και προτάσεις αγορών	67
Εικόνα 8.11. Dashboard του Γιάννη.....	68
«Get started» και «Log in»	71
Εικόνα 9.1. Αρχική σελίδα: ενότητα hero με τα κουμπιά «Get started» και «Log in» ..	71
Εικόνα 9.2. Αρχική σελίδα: ενότητα χαρακτηριστικών	72
Εικόνα 9.3. Αρχική σελίδα: υποσέλιδο (footer)	72
Εικόνα 9.4. Φόρμα εγγραφής (κενή)	73
Εικόνα 9.5. Φόρμα εγγραφής (συμπληρωμένη)	74
Εικόνα 9.6. Φόρμα σύνδεσης	75
Εικόνα 9.7. Μήνυμα σφάλματος σύνδεσης.....	75
Εικόνα 9.8. Dashboard νέου χρήστη (κενή γκαρνταρόμπα)	76
Εικόνα 9.9. Dashboard χρήστη με πλούσια γκαρνταρόμπα	77
Εικόνα 9.10. Ενότητα υπενθυμίσεων στο Dashboard	78
Εικόνα 9.11. Η γκαρνταρόμπα σε μορφή πλέγματος με κάρτες ρούχων	79
Εικόνα 9.12. Φόρμα προσθήκης ρούχου (κενή).....	80
Εικόνα 9.13. Φόρμα προσθήκης ρούχου (συμπληρωμένη, με φωτογραφία).....	81
Εικόνα 9.14. Φόρμα προσθήκης ρούχου με μήνυμα σφάλματος.....	82
Εικόνα 9.15. Κάρτα ρούχου με κουμπί «Edit».....	83
Εικόνα 9.16. Φόρμα επεξεργασίας με προσυμπληρωμένα πεδία.....	84
Εικόνα 9.17. Φόρμα δημιουργίας συνόλου: επιλογή ρούχων	85
Εικόνα 9.18. Μήνυμα σφάλματος για αρχειοθετημένο ρούχο	86
Εικόνα 9.19. Λίστα αποθηκευμένων συνόλων.....	87

Εικόνα 9.20. Επιλογή περίπτωσης για προτάσεις.....	88
Εικόνα 9.21. Κάρτες προτάσεων με βαθμολογία και επεξήγηση.....	88
Εικόνα 9.22. Κενή κατάσταση: μήνυμα «No work pieces in your wardrobe» με προτάσεις αγορών.....	89
Εικόνα 9.23. Κουμπί «Wore this today» σε κάρτα συνόλου.....	90
Εικόνα 9.24. Υπενθυμίσεις TRY_TODAY και SHELVE στο Dashboard	90
Εικόνα 9.25. Προτάσεις αγορών (wardrobe gap και retagging).....	92
Εικόνα 9.26. Ρυθμίσεις: στοιχεία λογαριασμού	93
Εικόνα 9.27. Ρυθμίσεις: αλλαγή κωδικού	93
Εικόνα 9.28. Ρυθμίσεις: διαγραφή λογαριασμού	94
Εικόνα 9.29. Γραμμή πλοήγησης	94

Κατάλογος Πινάκων

Πίνακας 3.1. Ρητό και σιωπηρό μοντέλο χρήστη: σύγκριση προσεγγίσεων και εφαρμογή στο Armoire	9
Πίνακας 3.2. Ευρετικές του Nielsen και εφαρμογή τους στο Armoire	14
Πίνακας 5.1. Σύγκριση του Armoire με παρόμοιες εφαρμογές γκαρνταρόμπας	20
Πίνακας 6.1. Ρόλοι χρηστών και λειτουργίες	22
Πίνακας 6.2. Λειτουργικές απαιτήσεις	23
Πίνακας 7.1. Αντιστοίχιση κατηγοριών και υποκατηγοριών ρούχων	36
Πίνακας 7.2. Χρωματικές ασυμβατότητες	42
Πίνακας 7.3. Ανάλυση συστήματος βαθμολόγησης	Error! Bookmark not defined.
Πίνακας 7.4. Κατάλογος REST API endpoints	53
Πίνακας 8.1. Ανάλυση βαθμολογίας προτάσεων WORK για τη Μαρία	58
Πίνακας 8.2. Ανάλυση βαθμολογίας προτάσεων CASUAL για τη Μαρία	63
Πίνακας 8.3. Ανάλυση βαθμολογίας προτάσεων CASUAL για τον Γιάννη	66
Πίνακας 8.4. Σύγκριση σεναρίων Μαρίας και Γιάννη	69

1. Εισαγωγή

1.1 Κίνητρο και Πλαίσιο Προβλήματος

Οι περισσότεροι άνθρωποι έχουν σήμερα περισσότερα ρούχα από όσα φοράνε πραγματικά. Το πρόβλημα δεν είναι η ποσότητα, αλλά η οργάνωση: οι περισσότεροι δεν έχουν έναν εύκολο τρόπο να δουν τι έχουν, να θυμηθούν συνδυασμούς που τους άρεσαν, ή να αποφασίσουν γρήγορα τι θα φορέσουν για μια συγκεκριμένη περίπτωση. Το αποτέλεσμα είναι ότι ξοδεύουν χρόνο ψάχνοντας στην ντουλάπα, ξεχνούν ρούχα που έχουν αγοράσει, και καταλήγουν να φοράνε πάντα τους ίδιους συνδυασμούς.

Η παρούσα πτυχιακή εργασία προτείνει μια λύση σε αυτό το πρόβλημα: την εφαρμογή Armoire, μια διαδικτυακή εφαρμογή διαχείρισης γκαρνταρόμπας.

1.2 Συνεισφορά

Η κύρια συνεισφορά της εργασίας είναι ένας μηχανισμός πρότασης εμφανίσεων που είναι εύκολα κατανοητός: εξηγεί γιατί προτείνει κάθε εμφάνιση, αντί να λειτουργεί σαν ένα «μαύρο κουτί». Ο μηχανισμός συνδυάζει απλούς κανόνες (π.χ. ποια χρώματα ταιριάζουν μεταξύ τους) με μάθηση από τις προηγούμενες επιλογές του χρήστη.

1.3 Μεθοδολογία και Τεχνική Προσέγγιση

Η εργασία ακολούθησε τα εξής βήματα: πρώτα, βιβλιογραφική έρευνα σε σχετικά θέματα. Έπειτα, καταγραφή των απαιτήσεων και των ρόλων των χρηστών. Στη συνέχεια, σχεδιασμός της αρχιτεκτονικής και της βάσης δεδομένων. Μετά, υλοποίηση του μηχανισμού πρότασης εμφανίσεων. Τέλος, τεκμηρίωση της λειτουργίας του συστήματος με παραδείγματα και εγχειρίδιο χρήσης.

Στην υλοποίηση χρησιμοποιήθηκαν οι εξής τεχνολογίες: React και TypeScript για το frontend, Spring Boot (Java 21) για το backend, και MySQL για τη βάση δεδομένων.

1.4 Δομή της Πτυχιακής

Το Κεφάλαιο 2 παρουσιάζει τους στόχους της εργασίας. Το Κεφάλαιο 3 περιέχει τη βιβλιογραφική ανασκόπηση. Το Κεφάλαιο 4 περιγράφει το πρόβλημα και το Κεφάλαιο 5 εξετάζει παρόμοιες εφαρμογές και τις διαφοροποιήσεις του Armoire από αυτές. Το Κεφάλαιο 6 αναλύει τις απαιτήσεις του συστήματος. Το Κεφάλαιο 7 παρουσιάζει τον σχεδιασμό και την υλοποίηση: αρχιτεκτονική, βάση δεδομένων, ασφάλεια και μηχανισμό προτάσεων. Το Κεφάλαιο 8 παρουσιάζει σενάρια χρήσης με στιγμιότυπα οθόνης. Το Κεφάλαιο 9 αποτελεί το εγχειρίδιο χρήστη. Το Κεφάλαιο 10 παρουσιάζει τα αναμενόμενα οφέλη της λύσης και το Κεφάλαιο 11 τις μελλοντικές επεκτάσεις. Το Κεφάλαιο 12 συνοψίζει τα συμπεράσματα. Ακολουθούν ο Πίνακας Ορολογίας, ο Πίνακας Συντμήσεων, η Βιβλιογραφία και τα Παραρτήματα.

2. Στόχοι της Πτυχιακής

2.1 Γενικός Στόχος

Ο γενικός στόχος της εργασίας είναι να φτιαχτεί μια διαδικτυακή εφαρμογή που να βοηθάει τους χρήστες να οργανώνουν τη γκαρνταρόμπα τους και να παίρνουν κατανοητές προτάσεις εμφανίσεων, χωρίς να χρειάζεται ειδικός εξοπλισμός ή συνδρομή.

2.2 Ειδικοί Στόχοι

2.2.1 Λειτουργικοί Στόχοι

- Δημιουργία ψηφιακού καταλόγου ρούχων με φωτογραφία, κατηγορία, χρώμα, εποχή και περίσταση.
- Χειροκίνητη δημιουργία εμφανίσεων, με έλεγχο αν τα χρώματα ταιριάζουν.
- Αυτόματη πρόταση εμφανίσεων ανά περίσταση, με εξήγηση του γιατί προτείνεται η καθεμία.
- Καταγραφή του πότε φοριέται μια εμφάνιση, ώστε το σύστημα να μαθαίνει τις προτιμήσεις του χρήστη.
- Υπενθυμίσεις για ρούχα που δεν έχουν φορεθεί, και προτάσεις για νέα ρούχα που λείπουν από τη γκαρνταρόμπα.

2.2.2 Τεχνικοί Στόχοι

- Αρχιτεκτονική τριών επιπέδων, με ξεκάθαρο διαχωρισμό frontend, API και βάσης δεδομένων.
- Ασφαλής σύνδεση χρηστών με JWT, χωρίς να στέλνεται ποτέ ο κωδικός σε απλό κείμενο.
- Βάση δεδομένων που υποστηρίζει πολλαπλές ετικέτες ανά ρούχο (π.χ. πολλές εποχές, πολλές περιστάσεις).

2.3 Περιορισμοί – Παραδοχές (Scope)

Η τρέχουσα έκδοση του Armoire καλύπτει: κατάλογο ρούχων, χειροκίνητη και αυτόματη δημιουργία εμφανίσεων, καταγραφή φοράς και διαχείριση λογαριασμού. Δεν υλοποιεί ακόμα την αυτόματη αναγνώριση ρούχων από φωτογραφία — αυτό συζητείται ως μελλοντική επέκταση στο Κεφάλαιο 11. Σημαντικό: το σύστημα βασίζεται εξ ολοκλήρου στις ετικέτες που βάζει ο ίδιος ο χρήστης όταν προσθέτει ένα ρούχο. Αν ένα ρούχο δεν επισημανθεί σωστά, το σύστημα δεν θα μπορέσει να το προτείνει.

3. Βιβλιογραφική Ανασκόπηση

Αυτό το κεφάλαιο παρουσιάζει τη σχετική βιβλιογραφία που στηρίζει τον σχεδιασμό του Armoire. Χωρίζεται σε οκτώ θεματικές ενότητες και μια σύνοψη: ολοκληρωμένα συστήματα, διαδικτυακές εφαρμογές, έξυπνες εφαρμογές και συστήματα, έξυπνες εφαρμογές οργάνωσης ντουλάπας, ευφυείς εφαρμογές και τεχνητή νοημοσύνη, εμπειρία χρήστη, προσαρμοστικότητα, και αλληλεπίδραση ανθρώπου-υπολογιστή.

3.1 Ολοκληρωμένα Συστήματα

Το Armoire είναι ένα ολοκληρωμένο σύστημα: ένα frontend, ένα API και μια βάση δεδομένων λειτουργούν μαζί σαν ένα ενιαίο προϊόν. Οι Land et al. [1] μελετούν πώς ενοποιούνται ξεχωριστά συστήματα σε ένα και περιγράφουν διαφορετικά επίπεδα ενοποίησης. Το Armoire ακολουθεί την προσέγγιση όπου το frontend, το backend και η βάση δεδομένων μοιράζονται την ίδια αναπαράσταση για ένα ρούχο και μια εμφάνιση. Οι Shibl et al. [2] εξηγούν τις συνηθισμένες δυσκολίες όταν ενοποιούνται διαφορετικά κομμάτια λογισμικού, όπως η ασυμφωνία δεδομένων. Για αυτόν τον λόγο, το Armoire χρησιμοποιεί ένα ενιαίο API (/api/v1) και δεν επιτρέπει στο frontend να έχει άμεση πρόσβαση στη βάση δεδομένων.

Οι Gupta et al. [3] περιγράφουν ένα σύστημα που συνδέει συσκευές καταγραφής δεδομένων με ένα κεντρικό σύστημα επεξεργασίας. Κάτι παρόμοιο γίνεται στο Armoire: οι φωτογραφίες που ανεβάζει ο χρήστης συνδέονται με ένα κεντρικό σύστημα που παράγει στατιστικά στο dashboard. Δύο ακόμη μελέτες, των Chukwumuanya et al. [4] και των Tanaman et al. [5], περιγράφουν συστήματα διαχείρισης αποθεμάτων για μικρές επιχειρήσεις που συνδυάζουν καταχώρηση προϊόντων, παρακολούθηση και αναφορές σε μία εφαρμογή. Αν και αυτά τα συστήματα αφορούν εμπορικό απόθεμα και όχι προσωπική γκαρνταρόμπα, ακολουθούν την ίδια λογική με το Armoire: κατάλογος, παρακολούθηση και αναφορές σε ένα ενιαίο εργαλείο. Τέλος, ο Desai [6] εξετάζει πώς μπορεί μια εφαρμογή να λειτουργεί και offline, κάτι που συζητείται στο Κεφάλαιο 5 σχετικά με την επιλογή να είναι το Armoire διαδικτυακή εφαρμογή και όχι εγκατεστημένη εφαρμογή.

3.2 Διαδικτυακές Εφαρμογές (Web-Based Applications)

Το backend του Armoire είναι ένα REST API, και το frontend είναι μια single-page εφαρμογή (SPA) σε React. Οι Golmohammadi et al. [7], στην ανασκόπησή τους για τον έλεγχο (testing) REST APIs, περιγράφουν τις βασικές αρχές ενός REST API: να μη διατηρεί κατάσταση (statelessness), να οργανώνεται γύρω από πόρους, και να έχει ενιαία δομή. Αυτές οι αρχές καθόρισαν τον σχεδιασμό των endpoints του Armoire (/api/v1/*, ενότητα 7.23). Οι Rodríguez et al. [8] εξέτασαν πολλά πραγματικά APIs και βρήκαν ότι λίγα ακολουθούν πλήρως τις αρχές REST — κάτι χρήσιμο για να συγκρίνουμε πόσο «καθαρό» είναι το API του Armoire.

Στο frontend, οι Gavrilă et al. [9] προτείνουν να χωρίζεται μια SPA σε ξεχωριστά επίπεδα (παρουσίαση, κατάσταση, δεδομένα), και οι Mukhiya και Hung [10] προτείνουν παρόμοια δομή με component-based βιβλιοθήκες. Το frontend του Armoire ακολουθεί αυτή τη λογική: τα components είναι χωρισμένα ανά λειτουργία (wardrobe, outfit), με ξεχωριστό επίπεδο για την επικοινωνία με το API. Οι Prayogi et al. [11] περιγράφουν τον σχεδιασμό και την υλοποίηση ενός REST API για ακαδημαϊκό πληροφοριακό σύστημα

— μια εμπειρία παρόμοια με αυτή που οδήγησε στην επιλογή του Spring Boot για το Armoire, λόγω της ασφάλειας και των έτοιμων εργαλείων που προσφέρει.

3.3 Έξυπνες Εφαρμογές/Συστήματα (Smart Application/System)

Η ιδέα της «έξυπνης γκαρνταρόμπας» υπάρχει στη βιβλιογραφία εδώ και χρόνια, συνήθως με χρήση ειδικού εξοπλισμού όπως ετικέτες RFID. Οι Goh et al. [12] παρουσιάζουν ένα από τα πρώτα τέτοια συστήματα, που παρακολουθεί τα ρούχα με RFID και λαμβάνει υπόψη χρώμα και περίσταση. Οι Indrawan et al. [13] δείχνουν ότι μια γκαρνταρόμπα μπορεί να μάθει τις συνήθειες του χρήστη απλά παρακολουθώντας ποια ρούχα κινούνται, χωρίς ο χρήστης να καταχωρεί τίποτα με το χέρι. Το Armoire έχει τον ίδιο στόχο — να καταλαβαίνει τι έχει και τι φοράει ο χρήστης — αλλά το πετυχαίνει μέσω φωτογραφιών και ετικετών, χωρίς να χρειάζεται ειδικός εξοπλισμός.

Οι Prakasa et al. [14] προσθέτουν αισθητήρες θερμοκρασίας και υγρασίας σε μια έξυπνη γκαρνταρόμπα, και οι Gaikwad et al. [15] κάνουν μια γενική επισκόπηση συστημάτων smart-home. Μια πρόσφατη μελέτη, των Alfantoukh et al. [16], συνδυάζει RFID και αναγνώριση εικόνας για την καταγραφή ρούχων σε πραγματικό χρόνο. Το Armoire, αντί για ειδικό εξοπλισμό, χρησιμοποιεί ήδη φωτογραφίες που ανεβάζει ο χρήστης — και θα μπορούσε στο μέλλον να χρησιμοποιήσει τεχνικές αναγνώρισης εικόνας, όπως αυτές που περιγράφουν οι Cheng et al. [17] στην ανασκόπηση τους για την όραση υπολογιστών στη μόδα (βλ. Κεφάλαιο 11).

3.4 Έξυπνες Εφαρμογές για την Οργάνωση Ντουλάπας και Ρουχισμού

Η προηγούμενη ενότητα εξέτασε έξυπνα συστήματα ντουλάπας που βασίζονται σε υλικό (αισθητήρες, RFID, IoT). Η παρούσα ενότητα εστιάζει στις λογισμικές εφαρμογές — εφαρμογές κινητού και ιστού — που επιτρέπουν στον χρήστη να οργανώσει ψηφιακά τη γκαρνταρόμπα του, να σχεδιάσει εμφανίσεις, και να λάβει προτάσεις ρουχισμού χωρίς να απαιτείται εξειδικευμένο υλικό.

Οι Jiang και Macintyre [18] πραγματοποιούν εκτενή ανάλυση 27 εφαρμογών διαχείρισης γκαρνταρόμπας, εξετάζοντας 5.953 αξιολογήσεις χρηστών από τα καταστήματα εφαρμογών της Apple και της Google. Η έρευνά τους αναδεικνύει ότι οι εφαρμογές αυτές δεν προσφέρουν μόνο πρακτικά οφέλη οργάνωσης, αλλά και μη αναμενόμενα οφέλη βιωσιμότητας: οι χρήστες αποκτούν επίγνωση του πόσο συχνά φοράνε κάθε ρούχο, κάτι που επηρεάζει την αγοραστική τους συμπεριφορά και τους ενθαρρύνει να αξιοποιούν αυτά που ήδη έχουν αντί να αγοράζουν νέα. Το Armoire ενσωματώνει αυτή τη λογική στον πυρήνα του: η καταγραφή φοράς και το σύστημα υπενθυμίσεων για αφόρετα ρούχα δεν σχεδιάστηκαν ως εργαλεία βιωσιμότητας, αλλά επιτυγχάνουν ακριβώς αυτό το αποτέλεσμα — ο χρήστης βλέπει ποια ρούχα αγνοεί και παροτρύνεται να τα αξιοποιήσει ή να τα αρχειοθετήσει.

Στο πεδίο της βιώσιμης μόδας, η μελέτη περίπτωσης της εφαρμογής Whering [19] δείχνει πώς μια εφαρμογή γκαρνταρόμπας μπορεί να λειτουργήσει ως εργαλείο αλλαγής συμπεριφοράς: η ψηφιοποίηση των ρούχων βοηθά τους χρήστες να «ξαναανακαλύψουν» κομμάτια που είχαν ξεχάσει, μειώνοντας την ανάγκη για νέες αγορές. Το Armoire ακολουθεί παρόμοια στρατηγική μέσω της σελίδας εποπτείας (dashboard), που εμφανίζει αναλυτικά στατιστικά ανά κατηγορία ρούχου, ανά χρώμα,

ανά περίπτωση, και επισημαίνει περιστάσεις για τις οποίες ο χρήστης δεν έχει αρκετά ρούχα — κάτι που αποτελεί εργαλείο εποπτείας, όχι απλής αποθήκευσης.

Από τεχνολογικής πλευράς, οι Alfantoukh et al. [16] παρουσιάζουν το Dithar, μια εφαρμογή που συνδυάζει IoT με ψηφιακή διαχείριση ντουλάπας. Σε αντίθεση με τις αμιγώς hardware λύσεις της ενότητας 3.3, το Dithar ενδιαφέρει εδώ γιατί το λογισμικό του κομμάτι — η ψηφιοποίηση, η αναζήτηση, και η οργάνωση μέσω εφαρμογής — λειτουργεί ανεξάρτητα από τους αισθητήρες. Το Armoire ακολουθεί αυτή τη λογική αλλά χωρίς απαίτηση υλικού: η ψηφιοποίηση γίνεται με χειροκίνητη καταχώρηση και φωτογραφία, κάτι που είναι πιο προσιτό αλλά λιγότερο αυτοματοποιημένο.

Η σχεδίαση εμφανίσεων αποτελεί βασική λειτουργία πολλών εφαρμογών γκαρνταρόμπας, αλλά ο τρόπος υλοποίησής της διαφέρει σημαντικά. Οι Singh et al. [20] προτείνουν μοντέλο βαθιάς μάθησης για αυτόματη σύσταση εμφανίσεων βασισμένη σε χαρακτηριστικά εικόνας — μια προσέγγιση που απαιτεί μεγάλο όγκο δεδομένων εκπαίδευσης. Αντίθετα, οι Patel et al. [21] ακολουθούν ανθρωποκεντρική προσέγγιση, ενσωματώνοντας αισθητήρες και ΑΙ σε ένα σύστημα όπου ο χρήστης παραμένει στο κέντρο της απόφασης. Το Armoire τοποθετείται μεταξύ αυτών των δύο άκρων: χρησιμοποιεί αλγοριθμικές τεχνικές (ομοιότητα συνημιτόνου, ομοιότητα Jaccard, κανόνες χρωματικής αρμονίας) για να παράγει προτάσεις, αλλά ο χρήστης αποφασίζει ποια εμφάνιση θα αποθηκεύσει και θα φορέσει — οι προτάσεις είναι εργαλείο, όχι εντολή.

Μια εμπορική εφαρμογή που ξεχωρίζει στη βιβλιογραφία είναι το Chérie [22], σχεδιασμένο με τρισδιάστατη εικονική δοκιμή και ΑΙ σύσταση εμφανίσεων. Η προσέγγιση του Chérie υπογραμμίζει μια τάση: οι χρήστες θέλουν να βλέπουν πώς θα μοιάζει μια εμφάνιση πριν την αποφασίσουν. Το Armoire δεν προσφέρει εικονική δοκιμή, αλλά αντιμετωπίζει το ίδιο πρόβλημα με οπτικό τρόπο: κάθε πρόταση εμφάνισης εμφανίζεται ως σύνθεση φωτογραφιών των ρούχων που την αποτελούν, ώστε ο χρήστης να αξιολογεί οπτικά τον συνδυασμό.

Οι Jiang και Macintyre [23] εξετάζουν επίσης τις αντιλήψεις των καταναλωτών σχετικά με λειτουργίες εφαρμογών που σχεδιάστηκαν για να μειώσουν τις περιττές αγορές μόδας. Η έρευνά τους δείχνει ότι πάνω από τους μισούς συμμετέχοντες πιστεύουν ότι η διαχείριση ντουλάπας μπορεί να αλλάξει τη σχέση τους με τα ρούχα που ήδη έχουν. Στο Armoire, αυτή η αλλαγή επιδιώκεται μέσω του συστήματος πρότασης αγορών: αντί να προτρέπει γενικές αγορές, η εφαρμογή εντοπίζει συγκεκριμένα κενά στη γκαρνταρόμπα (π.χ. δεν υπάρχει κατάλληλο ρούχο για μια περίπτωση) και προτείνει στοχευμένες αγορές — μια λογική που μετατρέπει την κατανάλωση από παρορμητική σε σχεδιασμένη.

Συνολικά, η βιβλιογραφία δείχνει ότι οι εφαρμογές διαχείρισης γκαρνταρόμπας βρίσκονται σε φάση ωρίμανσης: από απλές λίστες ρούχων, εξελίσσονται σε συστήματα που συνδυάζουν ψηφιοποίηση, αλγοριθμική σύσταση, και εργαλεία βιωσιμότητας. Το Armoire τοποθετείται σε αυτό ακριβώς το σημείο: δεν είναι απλός κατάλογος ρούχων, αλλά ένα σύστημα που μαθαίνει από τη συμπεριφορά του χρήστη, προτείνει εμφανίσεις με βάση πολλαπλά κριτήρια, υπενθυμίζει αφόρετα ρούχα, και εντοπίζει κενά —

λειτουργίες που, σύμφωνα με τους Jiang και Macintyre [18], αποτελούν τα χαρακτηριστικά που κάνουν αυτές τις εφαρμογές πραγματικά χρήσιμες.

3.5 Ευφυείς Εφαρμογές και Τεχνητή Νοημοσύνη (Intelligent Systems & AI)

Ο πυρήνας του Armoire είναι ένα σύστημα συστάσεων που αποφασίζει ποιες εμφανίσεις να προτείνει σε κάθε χρήστη. Οι Adomavicius και Tuzhilin [24] διακρίνουν τρεις κατηγορίες τέτοιων συστημάτων: με βάση το περιεχόμενο (αξιολόγηση των ιδιοτήτων του αντικειμένου), συνεργατικά (αξιοποίηση προτιμήσεων παρόμοιων χρηστών), και υβριδικά. Το Armoire ακολουθεί υβριδική προσέγγιση: κρίνει κάθε εμφάνιση τόσο με κανόνες μόδας (π.χ. αν τα χρώματα ταιριάζουν, αν τα ρούχα ταιριάζουν στην περίπτωση) όσο και με βάση τι έχει επιλέξει ο χρήστης στο παρελθόν.

Για να γίνει αυτή η σύγκριση, κάθε ρούχο στη ντουλάπα του χρήστη αναπαρίσταται ως ένα διάνυσμα αριθμών — μια τεχνική που περιγράφουν οι Salton και Buckley [25]. Στην πράξη, αυτό σημαίνει ότι ένα μαύρο φόρεμα για βράδυ κωδικοποιείται ως ένα σύνολο αριθμών που αντιστοιχούν στην κατηγορία (φόρεμα), το χρώμα (μαύρο), την περίπτωση (βραδινή) και την εποχή (κατάλληλο για χειμώνα). Μια ολόκληρη εμφάνιση αξιολογείται μέσω των ρούχων που την αποτελούν: το σύστημα υπολογίζει την ομοιότητα κάθε ρούχου με το προφίλ του χρήστη και παίρνει τον μέσο όρο τους. Αυτή η αριθμητική αναπαράσταση είναι αυτή που επιτρέπει στο σύστημα να «καταλαβαίνει» τι περιέχει κάθε εμφάνιση χωρίς να βασίζεται μόνο σε ετικέτες.

Για να αξιολογήσει αν μια εμφάνιση ταιριάζει στον χρήστη, το Armoire χρησιμοποιεί την ομοιότητα συνημιτόνου (cosine similarity), που σύμφωνα με τους Wang και Dong [26] είναι από τις πιο αξιόπιστες μετρικές σύγκρισης διανυσμάτων. Πρακτικά: αν ένας χρήστης επιλέγει συστηματικά casual εμφανίσεις σε γήινους τόνους, το προφίλ προτιμήσεών του θα «μοιάζει» περισσότερο με νέες casual εμφανίσεις παρά με επίσημες — και αυτή ακριβώς τη μαθηματική ομοιότητα μετράει η cosine similarity. Παράλληλα, η ομοιότητα Jaccard — με βελτιωμένες παραλλαγές που προτείνουν οι Bag et al. [27] — χρησιμοποιείται για να εντοπίσει αν μια υποψήφια εμφάνιση μοιράζεται πολλά κοινά κομμάτια με εμφανίσεις που ο χρήστης έχει ήδη αποθηκεύσει, ώστε να ενισχύονται οι συνδυασμοί που μοιάζουν με αυτούς που του αρέσουν. Ο συνδυασμός των δύο μετρικών συνδέει τις προτάσεις τόσο με τα χαρακτηριστικά που προτιμά ο χρήστης όσο και με τα συγκεκριμένα σύνολα που έχει επιλέξει.

Εκτός από τις στατιστικές μετρικές, το Armoire εφαρμόζει κανόνες χρωματικής αρμονίας: π.χ. αποφεύγει να συνδυάσει δύο κομμάτια με χρώματα που «χτυπούν» σύμφωνα με τον χρωματικό κύκλο. Ωστόσο, οι Li et al. [28] δείχνουν πειραματικά ότι η αντίληψη των ανθρώπων για χρωματικούς συνδυασμούς αποκλίνει συχνά από αυτό που ορίζει η θεωρία — κάποιιοι θεωρητικά «ακατάλληλοι» συνδυασμοί αρέσουν στον κόσμο, και κάποιοι «σωστοί» αφήνουν αδιάφορους. Αυτό σημαίνει ότι οι κανόνες του Armoire λειτουργούν ως χρήσιμο φίλτρο αλλά δεν αντικαθιστούν το προσωπικό γούστο — γι' αυτό και το σύστημα τους χρησιμοποιεί σε συνδυασμό με την ιστορική συμπεριφορά του χρήστη, όχι μεμονωμένα.

Πιο σύνθετες προσεγγίσεις βαθιάς μάθησης έχουν δείξει ισχυρότερα αποτελέσματα. Οι Han et al. [29] εκπαιδεύουν νευρωνικό δίκτυο που μαθαίνει αυτόματα ποια ρούχα ταιριάζουν μεταξύ τους — αντί να βασίζεται σε χειροκίνητους κανόνες, εξάγει μοτίβα από χιλιάδες υπάρχουσες εμφανίσεις. Οι Vasileva et al. [30] πάνε ένα βήμα παραπέρα: κατασκευάζουν έναν κοινό χώρο αναπαράστασης (embedding space) που επιτρέπει τη σύγκριση ρούχων εντελώς διαφορετικού τύπου — π.χ. αν ένα συγκεκριμένο πουκάμισο ταιριάζει με ένα συγκεκριμένο παντελόνι, ακόμα κι αν ποτέ δεν εμφανίστηκαν μαζί. Για το Armoire, αυτές οι τεχνικές δεν είναι εφαρμόσιμες σήμερα γιατί απαιτούν δεκάδες χιλιάδες δείγματα εκπαίδευσης, αλλά αποτελούν φυσική εξέλιξη καθώς η βάση χρηστών μεγαλώνει και συσσωρεύονται δεδομένα.

Μια ακόμη δυνατότητα που θα μπορούσε να προστεθεί στο μέλλον είναι η αναζήτηση παρόμοιων ρούχων από φωτογραφία — π.χ. ο χρήστης βλέπει ένα μπουφάν στο δρόμο, το φωτογραφίζει, και το Armoire βρίσκει παρόμοια κομμάτια στη ντουλάπα του ή στα καταστήματα. Το CLIP [31] μπορεί να μετατρέψει μια εικόνα σε διάνυσμα που «καταλαβαίνει» τι απεικονίζεται, και το FAISS [32] μπορεί να αναζητήσει γρήγορα τα πιο όμοια διανύσματα σε μεγάλη βάση δεδομένων. Ο συνδυασμός τους θα έδινε στο Armoire μια λειτουργία οπτικής αναζήτησης χωρίς να χρειαστεί ο χρήστης να περιγράψει με λόγια τι ψάχνει.

Οι Mohammadi και Kalhor [33] επιβεβαιώνουν μέσα από ευρεία ανασκόπηση ότι η χρήση τεχνητής νοημοσύνης στη μόδα αποτελεί ενεργό ερευνητικό πεδίο, ενώ οι Ding et al. [34] σημειώνουν ότι οι περισσότερες από αυτές τις τεχνικές δεν είναι ακόμη εύκολα εφαρμόσιμες σε μικρές εφαρμογές. Η σχεδιαστική φιλοσοφία του Armoire αντιμετωπίζει αυτή ακριβώς τη δυσκολία: ξεκινάει με τεχνικές που δουλεύουν αξιόπιστα με λίγα δεδομένα (κανόνες + cosine/Jaccard), αλλά η αρθρωτή αρχιτεκτονική του επιτρέπει κάθε στοιχείο — τη χρωματική αρμονία, τη μετρική ομοιότητας, τον τρόπο αναπαράστασης — να αντικατασταθεί ανεξάρτητα με πιο σύνθετο μοντέλο, χωρίς να χρειαστεί να ξαναγραφτεί το υπόλοιπο σύστημα.

Η Chrysafiadi [35] παρουσιάζει πώς η ασαφής λογική μπορεί να ενσωματωθεί σε ευφυή συστήματα λογισμικού, αντιμετωπίζοντας αποφάσεις που δεν είναι δυαδικές αλλά βαθμιαίες. Στο Armoire, αυτή η ανάγκη εμφανίζεται στη βαθμολόγηση εμφανίσεων: το σύστημα υπολογίζει ένα σύνθετο σκορ από πολλαπλά σήματα — χρωματική αρμονία, πληρότητα συνδυασμού, και ομοιότητα με τις μαθημένες προτιμήσεις του χρήστη. Τα βάρη αυτών των σημάτων είναι σήμερα σταθερές, που σημαίνει ότι η σχέση μεταξύ ομοιότητας και τελικού σκορ είναι γραμμική. Η ασαφής λογική θα μπορούσε να αντικαταστήσει αυτή τη γραμμικότητα: αντί ένα σταθερό βάρος να πολλαπλασιάζεται με τον βαθμό ομοιότητας, ένας ασαφής κανόνας θα μπορούσε να λαμβάνει υπόψη πόσο πρόσφατα ή πόσο συχνά εκδηλώθηκε η προτίμηση — μια εμφάνιση που φορέθηκε πολλές φορές τον τελευταίο μήνα να σταθμίζεται διαφορετικά από μία που φορέθηκε μία φορά πριν έξι μήνες, ακόμα κι αν ο συνολικός αριθμός φορών είναι ίδιος. Σε άλλο πεδίο, οι Chrysafiadi et al. [36] εφαρμόζουν ασαφή λογική σε σύστημα πρόληψης πυρκαγιάς στο σπίτι, μοντελοποιώντας ανθρώπινα σφάλματα λόγω απόσπασης προσοχής — μια αβεβαιότητα ανάλογη με αυτήν που αντιμετωπίζει κάθε σύστημα σύστασης ρουχισμού, όπου οι προτιμήσεις δεν είναι ποτέ δυαδικές αλλά εξαρτώνται από πλαίσιο (εποχή, διάθεση, περίσταση). Στο πλαίσιο των υβριδικών ευφυών συστημάτων, οι Panagoulas et al. [37] συνδυάζουν αυστηρούς κανόνες με τεχνητή

νοημοσύνη (GPT) σε εκπαιδευτική εφαρμογή πολιτισμού, δείχνοντας ότι ένα σύστημα που χρησιμοποιεί κανόνες για ακρίβεια και ΑΙ για ευελιξία υπερτερεί κάθε μεμονωμένης προσέγγισης. Το Armoire ακολουθεί ακριβώς αυτή τη φιλοσοφία: εφαρμόζει αυστηρούς κανόνες χρωματικής αρμονίας που απορρίπτουν ασύμβατους συνδυασμούς πριν αξιολογηθούν, και ταυτόχρονα μετρικές ομοιότητας που μαθαίνουν από τη συμπεριφορά του χρήστη (ομοιότητα συνημιτόνου, ομοιότητα Jaccard) για να κατατάξει τις εναπομείνουσες εμφανίσεις με βάση τις προτιμήσεις.

Πέρα από τις αλγοριθμικές τεχνικές, η τεχνητή νοημοσύνη εφαρμόζεται και σε ευρύτερα πεδία εξατομίκευσης. Οι Vali-Sarafoğlu και Chrysafiadi [38] αναπτύσσουν σύστημα σύστασης ταινιών βασισμένο στη μέθοδο TOPSIS, η οποία κατατάσσει επιλογές αξιολογώντας ταυτόχρονα πολλαπλά κριτήρια και επιλέγοντας αυτήν που απέχει λιγότερο από την ιδανική λύση. Η αρχιτεκτονική βαθμολόγησης του Armoire ακολουθεί ανάλογη πολυκριτηριακή λογική: κάθε εμφάνιση αξιολογείται ταυτόχρονα ως προς τη χρωματική αρμονία, την πληρότητα, και την ομοιότητα με το προφίλ στυλ του χρήστη — ο τελικός αριθμός είναι ένα σύνθετο σκορ, όχι μεμονωμένο κριτήριο. Μια μελλοντική μετάβαση σε TOPSIS θα μπορούσε να κανονικοποιήσει αυτά τα ετερογενή κριτήρια σε κοινή κλίμακα, αποφεύγοντας τα εμπειρικά βάρη που χρησιμοποιούνται σήμερα. Οι Chrysafiadi et al. [39] εφαρμόζουν ασαφή λογική στον τουρισμό, εξατομικεύοντας τα σημεία ενδιαφέροντος ανάλογα με το προφίλ του επισκέπτη. Η αναλογία με το Armoire είναι δομική: όπως ο τουρίστας έχει ένα προφίλ ενδιαφερόντων (π.χ. ιστορία, γαστρονομία) που κατασκευάζεται από τις αναζητήσεις και τις επισκέψεις του, ο χρήστης του Armoire έχει ένα προφίλ στυλ που κωδικοποιεί τις προτιμήσεις του ως σταθμισμένο διάνυσμα χαρακτηριστικών (κατηγορία ρούχου, χρώμα, περίσταση, εποχή). Στο ίδιο πνεύμα, οι Kontogianni et al. [40] παρουσιάζουν μια αναπαραγωγική διαδικασία ταξινόμησης σημείων ενδιαφέροντος τουρισμού που συνδυάζει τεχνητή νοημοσύνη και ασαφή λογική, τονίζοντας τη σημασία της ερμηνευσιμότητας στα αποτελέσματα — δηλαδή ο τελικός χρήστης να μπορεί να καταλάβει γιατί μια σύσταση εμφανίστηκε ψηλά. Αυτή η αρχή εφαρμόζεται στο Armoire μέσω των επεξηγηματικών κειμένων που συνοδεύουν κάθε πρόταση εμφάνισης, ώστε ο χρήστης να καταλαβαίνει αν η εμφάνιση προτάθηκε λόγω χρωματικής αρμονίας, λόγω ομοιότητας με το στυλ του, ή και για τους δύο λόγους.

3.6 Εμπειρία Χρήστη (User Experience)

Η εμπειρία χρήστη ήταν σημαντικός παράγοντας στον σχεδιασμό του Armoire, ειδικά στις οθόνες που εμφανίζονται όταν ο χρήστης δεν έχει ακόμα ρούχα ή προτάσεις. Ο Kaplan [41] εξηγεί ότι μια «κενή» οθόνη δεν πρέπει να μένει κενή: πρέπει να εξηγεί στον χρήστη τι πρέπει να κάνει και να τον καθοδηγεί στο επόμενο βήμα. Ακριβώς αυτή τη λογική ακολουθεί η οθόνη κενής κατάστασης του Armoire (ενότητα 7.20). Οι Lindgaard et al. [42] δείχνουν ότι ο χρήστης σχηματίζει άποψη για μια ιστοσελίδα μέσα σε μόλις 50 χιλιοστά του δευτερολέπτου, γι' αυτό δόθηκε ιδιαίτερη προσοχή στην αρχική σελίδα και στο dashboard του Armoire.

Η μέτρηση της εμπειρίας χρήστη δεν είναι εύκολη υπόθεση: οι Nakamura et al. [43] δείχνουν ότι διαφορετικά εργαλεία αξιολόγησης μπορούν να δώσουν αντικρουόμενα αποτελέσματα για το ίδιο σύστημα, και οι Perrig et al. [44] παρατηρούν ότι οι περισσότερες μελέτες δεν εξηγούν γιατί επέλεξαν συγκεκριμένο τρόπο μέτρησης. Οι Sulikowski et al. [45] δείχνουν ότι μια καλή αισθητική σε ένα ηλεκτρονικό κατάστημα

ρούχων τραβάει περισσότερο την προσοχή του χρήστη στις προτάσεις, και οι Aayesha et al. [46] προτείνουν έναν τρόπο αξιολόγησης ειδικά για συστήματα συστάσεων ρούχων — και τα δύο σχετικά με το πώς παρουσιάζονται οι προτάσεις στο Armoire.

3.7 Προσαρμοστικότητα – Προσαρμοστικά Συστήματα (Adaptivity / Adaptive Systems)

Ένα σύστημα συστάσεων γίνεται προσαρμοστικό όταν δεν αρκείται στο να εφαρμόζει σταθερούς κανόνες, αλλά χτίζει ένα μοντέλο για κάθε χρήστη και αλλάζει τη συμπεριφορά του ανάλογα. Οι προηγούμενες ενότητες εξέτασαν τις αλγοριθμικές τεχνικές (ομοιότητα, φιλτράρισμα, βαθμολόγηση) — η παρούσα ενότητα εστιάζει στην προσαρμοστικότητα ως *σχεδιαστική αρχή*: πώς ένα σύστημα αποφασίζει *πότε* και *τι* να προσαρμόσει, πώς κατασκευάζει το μοντέλο χρήστη, και πώς αντιμετωπίζει τις περιπτώσεις που δεν έχει ακόμα αρκετά δεδομένα.

Ο Brusilovsky [47] ορίζει ένα προσαρμοστικό σύστημα ως αυτό που κατασκευάζει ένα μοντέλο χρήστη (user model) και το χρησιμοποιεί για να τροποποιεί δύο πράγματα: *τι* παρουσιάζει (adaptive content) και *πώς* το παρουσιάζει (adaptive presentation). Στο Armoire, αυτή η διάκριση εκδηλώνεται συγκεκριμένα: η σειρά κατάταξης των προτεινόμενων εμφανίσεων αλλάζει ανάλογα με το προφίλ προτιμήσεων (adaptive content), ενώ τα highlights που συνοδεύουν κάθε πρόταση — π.χ. «Matches your style preferences» ή η αναφορά σε preference boost — αλλάζουν ανάλογα με τη δύναμη του σήματος ομοιότητας (adaptive presentation). Αυτή η διπλή προσαρμογή εξηγεί γιατί δύο χρήστες με τα ίδια ρούχα αλλά διαφορετικό ιστορικό φοράς βλέπουν διαφορετικές κατατάξεις και διαφορετικές εξηγήσεις.

Οι Miraz et al. [48] κατηγοριοποιούν τα προσαρμοστικά συστήματα ανάλογα με τον *τρόπο κατασκευής* του μοντέλου χρήστη: ρητό (explicit), σιωπηρό (implicit), ή υβριδικό. Το μοντέλο χρήστη του Armoire είναι αποκλειστικά σιωπηρό — κατασκευάζεται αυτόματα από δύο σήματα συμπεριφοράς: ποιες εμφανίσεις αποθηκεύει ο χρήστης (βάρος 1.0) και πόσες φορές τις φοράει (βάρος $2.0 \times \text{wearCount}$). Δεν ζητείται ποτέ από τον χρήστη να δηλώσει τις προτιμήσεις του — αυτό μειώνει το γνωστικό φορτίο κατά την εγγραφή, αλλά δημιουργεί το πρόβλημα της ψυχρής εκκίνησης (cold start), καθώς χωρίς ιστορικό φοράς, το μοντέλο είναι κενό. Στην τρέχουσα έκδοση, η εφαρμογή αντιμετωπίζει αυτό το πρόβλημα με υποβάθμιση (fallback): όταν το προφίλ είναι κενό, η βαθμολόγηση βασίζεται αποκλειστικά στη δομή του συνδυασμού (πληρότητα, χρωματική αρμονία), χωρίς πρόσθετο boost — δηλαδή υποβαθμίζεται σε ένα μη προσαρμοστικό σύστημα κανόνων μέχρι να συγκεντρωθούν αρκετά δεδομένα.

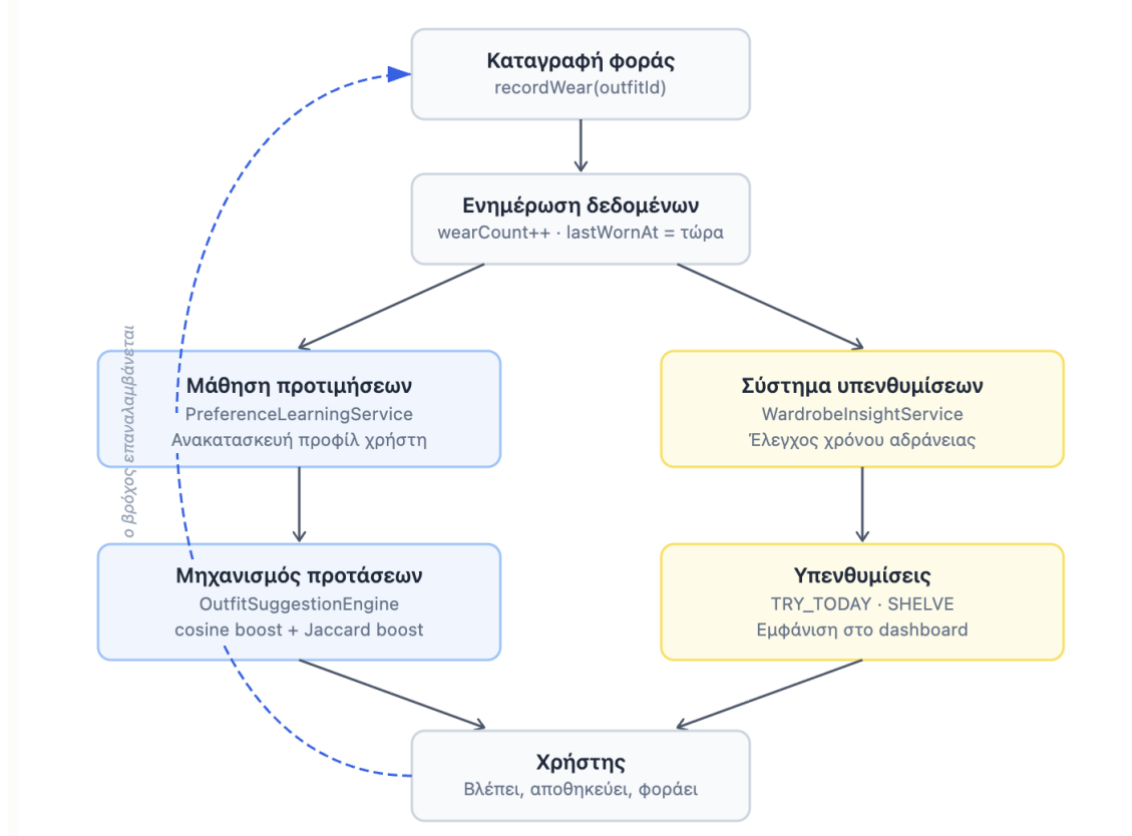
Πίνακας 3.1. Ρητό και σιωπηρό μοντέλο χρήστη: σύγκριση προσεγγίσεων και εφαρμογή στο Armoire

Χαρακτηριστικό	Ρητό μοντέλο (explicit)	Σιωπηρό μοντέλο (implicit)	Armoire
Πηγή δεδομένων	Ο χρήστης δηλώνει προτιμήσεις (ερωτηματολόγιο, βαθμολογίες)	Το σύστημα παρατηρεί τη συμπεριφορά	Σιωπηρό: αποθηκευμένες εμφανίσεις (βάρος 1.0) + καταγραφή φοράς (βάρος $2.0 \times \text{wearCount}$)

Χαρακτηριστικό	Ρητό μοντέλο (explicit)	Σιωπηρό μοντέλο (implicit)	Armoire
Γνωστικό φορτίο χρήστη	Υψηλό — απαιτεί ενεργή συμμετοχή	Μηδενικό — δεν ζητείται τίποτα	Μηδενικό — ο χρήστης δεν ρωτιέται ποτέ για τις προτιμήσεις του
Ψυχρή εκκίνηση (cold start)	Λύνεται αμέσως — ο χρήστης δηλώνει κατά την εγγραφή	Πρόβλημα — χωρίς ιστορικό, το μοντέλο είναι κενό	Αντιμετωπίζεται με υποβάθμιση: χωρίς δεδομένα, η βαθμολόγηση βασίζεται μόνο σε δομικούς κανόνες (χρωματική αρμονία, πληρότητα)
Ακρίβεια Μακροπρόθεσμα	Στατική — αν δεν ενημερωθεί, ξεπερνιέται	Αυξάνεται με τη χρήση	Αυξάνεται: κάθε καταγραφή φοράς ενισχύει το προφίλ
Διαφάνεια	Ο χρήστης ξέρει τι δήλωσε	Ο χρήστης μπορεί να μην ξέρει τι έμαθε το σύστημα	Υβριδική: σιωπηρή μάθηση, αλλά ορατή στο UI (ένδειξη «+N from your style history» στις κάρτες πρότασης και ένδειξη «On · style profile» όταν δεν βρίσκονται προτάσεις)

Οι Purificato et al. [49] διακρίνουν μεταξύ «ρητής αλληλεπίδρασης» (ο χρήστης βαθμολογεί ή δηλώνει) και «σιωπηρής αλληλεπίδρασης» (το σύστημα παρατηρεί). Αυτοί δείχνουν ότι τα σύγχρονα συστήματα τείνουν προς τη σιωπηρή αλληλεπίδραση, γιατί μειώνει την κόπωση του χρήστη. Το Armoire υιοθετεί αυτή τη φιλοσοφία, αλλά με μια σημαντική σχεδιαστική επιλογή: κάθε σιωπηρή μάθηση εκδηλώνεται *ορατά* στο UI (π.χ. badge «+N from your style history» στις κάρτες πρότασης, ή ένδειξη «On · style profile» στο διαγνωστικό πλαίσιο όταν δεν βρίσκονται προτάσεις). Αυτό ακολουθεί τη λογική του explainable recommendation: ο χρήστης μπορεί να καταλάβει *γιατί* η εφαρμογή τον γνωρίζει, αντί να αισθάνεται ότι παρακολουθείται.

Το Recommender Systems Handbook (Ricci et al. [50]) εισάγει την έννοια του βρόχου ανατροφοδότησης (feedback loop): ένα προσαρμοστικό σύστημα πρέπει να μαθαίνει → να προτείνει → να παρατηρεί αν η πρόταση ακολουθήθηκε → να ενημερώνει το μοντέλο. Στο Armoire, αυτός ο βρόχος λειτουργεί ως εξής: η καταγραφή φοράς (recordWear) ενημερώνει τον μετρητή wearCount κάθε ρούχου → το PreferenceLearningService ξαναχτίζει το προφίλ → οι επόμενες προτάσεις κατατάσσονται με βάση το ενημερωμένο μοντέλο. Αυτός ο βρόχος δεν γίνεται σε πραγματικό χρόνο (ο χρήστης πρέπει να ζητήσει νέες προτάσεις), αλλά η ενημέρωση του μοντέλου είναι άμεση — κάτι που μπορεί να βελτιωθεί μελλοντικά με proactive re-ranking.



Εικόνα 3.1. Βρόχος ανατροφοδότησης: καταγραφή φοράς, μάθηση προτιμήσεων και υπενθυμίσεις στο Armoire

Οι Sayed et al. [51] παρουσιάζουν ένα προσαρμοστικό σύστημα e-learning όπου η διαδρομή μάθησης αλλάζει ανάλογα με την απόδοση του μαθητή. Η αναλογία με το Armoire δεν βρίσκεται στο recommender, αλλά στο σύστημα nudge: όπως ένα e-learning σύστημα προτρέπει τον μαθητή να επαναλάβει ύλη που δεν κατέκτησε, έτσι το Armoire προτρέπει τον χρήστη να δοκιμάσει ρούχα που αγνοεί (TRY_TODAY) ή να αρχειοθετήσει ρούχα που δεν χρησιμοποιεί (SHELVE). Τα κατώφλια αυτά είναι σήμερα σταθερά (configurable σε αριθμό ημερών), αλλά θα μπορούσαν στο μέλλον να προσαρμόζονται ανά χρήστη — π.χ. κάποιος που φοράει κυρίως εποχιακά ρούχα χρειάζεται μεγαλύτερο κατώφλι.

Στον χώρο της εκπαίδευσης, οι Chrysafiadi et al. [52] αξιολογούν ένα προσαρμοστικό σύστημα εκμάθησης προγραμματισμού (ITS) βασισμένο σε ασαφή λογική και μηχανική μάθηση, και δείχνουν ότι η προσαρμογή του περιεχομένου στο επίπεδο του μαθητή βελτιώνει τόσο τα μαθησιακά αποτελέσματα όσο και την εμπειρία χρήστη. Στο Armoire, η αντίστοιχη προσαρμογή λειτουργεί μέσω του μηχανισμού μάθησης προτιμήσεων: το σύστημα κατασκευάζει ένα μοντέλο χρήστη εξ ολοκλήρου από σιωπηρά σήματα — ποιες εμφανίσεις αποθήκευσε ο χρήστης και πόσες φορές τις φόρεσε. Η βαρύτητα της φοράς είναι μεγαλύτερη από αυτήν της αποθήκευσης, γιατί η πράξη του «φορώ» εκφράζει ισχυρότερη προτίμηση — ακριβώς όπως σε ένα ITS η σωστή απάντηση σε άσκηση βαρύνει περισσότερο από την απλή ανάγνωση θεωρίας. Η

αντιστοιχία είναι εννοιολογική: και στα δύο συστήματα, η πράξη αποκαλύπτει περισσότερα από τη δήλωση.

Οι Chrysafiadi et al. [53] επεκτείνουν αυτή τη λογική στα εκπαιδευτικά παιχνίδια, εφαρμόζοντας προσαρμοστικότητα ταυτόχρονα στην πλοκή και στο περιβάλλον του παιχνιδιού — δηλαδή σε πολλαπλά επίπεδα αλληλεπίδρασης. Στο Armoire, η πολυεπίπεδη προσαρμογή εκδηλώνεται σε δύο σημεία: πρώτον, στην κατάταξη των προτάσεων, όπου η σειρά αλλάζει ανάλογα με το μοντέλο χρήστη (adaptive content), και δεύτερον, στα επεξηγηματικά κείμενα κάθε κάρτας πρότασης, που αλλάζουν ανάλογα με τη δύναμη του σήματος ομοιότητας — αν η ομοιότητα υπερβαίνει ένα κατώφλι, ο χρήστης βλέπει ένδειξη ότι η εμφάνιση ταιριάζει με το στυλ του, αλλιώς η εμφάνιση παρουσιάζεται χωρίς αυτήν (adaptive presentation). Αυτή η διπλή προσαρμογή — τι βλέπει ο χρήστης και πώς του παρουσιάζεται — αντιστοιχεί στη λογική πλοκής/περιβάλλοντος που περιγράφουν οι Chrysafiadi et al.

Οι Papadimitriou et al. [54] προτείνουν προσαρμοστικά κουίζ με χρήση ασαφούς γενετικού αλγορίθμου, που επιλέγει δυναμικά τη δυσκολία κάθε ερώτησης ανάλογα με την απόδοση του μαθητή. Η αναλογία με το Armoire βρίσκεται στο σύστημα υπενθυμίσεων (nudges): η ένταση της υπενθύμισης κλιμακώνεται ανάλογα με τη διάρκεια αδράνειας — ένα ρούχο που δεν φορέθηκε για μικρό χρονικό διάστημα λαμβάνει ήπια παρότρυνση, ένα που αγνοήθηκε πολύ περισσότερο λαμβάνει εντονότερη, ενώ ένα ρούχο που δεν φορέθηκε ποτέ αντιμετωπίζεται νωρίτερα. Αυτή η κλιμακωτή ένταση αποτελεί μορφή προσαρμοστικής συμπεριφοράς — ακριβώς εκεί η ασαφής λογική θα μπορούσε να εισέλθει μελλοντικά, προσαρμόζοντας τα κατώφλια ανάλογα με τις συνήθειες κάθε χρήστη αντί να χρησιμοποιεί σταθερές ημέρες.

Παράλληλα, οι Chrysafiadi et al. [55] αναπτύσσουν γνωστικά-βασισμένα προσαρμοστικά σενάρια σε εκπαιδευτικά παιχνίδια χρησιμοποιώντας ασαφή συλλογισμό, ενώ οι Chrysafiadi et al. [56] παρουσιάζουν μηχανισμό αυτόματης εξατομικευμένης αξιολόγησης βασισμένο σε ασαφή λογική. Και οι δύο εργασίες δείχνουν ότι η ασαφής λογική μπορεί να μοντελοποιεί τη γνωστική κατάσταση ενός χρήστη χωρίς δυαδικές κατηγοριοποιήσεις — κάτι που στο Armoire θα σήμαινε ότι ένας χρήστης δεν θεωρείται απλώς «νέος» ή «έμπειρος», αλλά βρίσκεται σε ένα συνεχές φάσμα εξοικείωσης, που θα μπορούσε να επηρεάζει τόσο τον αριθμό προτάσεων που εμφανίζονται (λιγότερες αρχικά, περισσότερες αργότερα) όσο και τον βαθμό λεπτομέρειας στις εξηγήσεις που τις συνοδεύουν. Τέλος, οι Chrysafiadi et al. [57] παρουσιάζουν προσαρμοστικό περιβάλλον μάθησης που συνδυάζει ασαφή λογική και μηχανική μάθηση — ένα υβριδικό μοντέλο που δείχνει ότι οι δύο τεχνικές δεν είναι ανταγωνιστικές αλλά συμπληρωματικές, κάτι σχετικό με τη σχεδιαστική φιλοσοφία του Armoire, που ήδη συνδυάζει κανόνες (χρωματικοί, περίστασης) με στατιστική μάθηση (ομοιότητα συνημιτόνου) στον ίδιο μηχανισμό βαθμολόγησης.

Τέλος, οι Tibensky και Kompan [58] προτείνουν μια μετα-στρατηγική: αντί ένα σύστημα να χρησιμοποιεί πάντα τον ίδιο αλγόριθμο, να επιλέγει δυναμικά ποιος αλγόριθμος ταιριάζει καλύτερα σε κάθε χρήστη και κάθε περίπτωση. Το Armoire δεν εφαρμόζει σήμερα αυτή τη στρατηγική — χρησιμοποιεί πάντα τον ίδιο συνδυασμό βαθμολόγησης — αλλά η αρχιτεκτονική του (ξεχωριστό PreferenceLearningService,

ξεχωριστό OutfitSuggestionEngine) επιτρέπει μελλοντικά να εναλλάσσονται στρατηγικές χωρίς αλλαγή στο υπόλοιπο σύστημα.

3.8 Αλληλεπίδραση Ανθρώπου-Υπολογιστή / Διεπαφές Χρήστη (HCI/UX)

Ο σχεδιασμός μιας εφαρμογής γκαρνταρόμπας θέτει ένα συγκεκριμένο πρόβλημα HCI: ο χρήστης αλληλεπιδρά με οπτικό περιεχόμενο (φωτογραφίες ρούχων, χρωματικούς συνδυασμούς) και πρέπει να παίρνει γρήγορες αποφάσεις. Αυτή η ενότητα εξετάζει τις αρχές HCI που καθοδήγησαν τρεις κεντρικές σχεδιαστικές αποφάσεις του Armoire: την ευρετική αξιολόγηση της διεπαφής, τη μετάβαση μεταξύ κενής και πλήρους κατάστασης, και τον σχεδιασμό στόχων αλληλεπίδρασης.

Η Virvou [59] εξετάζει τη σχέση αμοιβαιότητας μεταξύ τεχνητής νοημοσύνης και εμπειρίας χρήστη: η τεχνητή νοημοσύνη βελτιώνει τη διεπαφή μέσω εξατομίκευσης, αλλά ταυτόχρονα μια καλή διεπαφή ενθαρρύνει συμπεριφορές που τροφοδοτούν τα δεδομένα εκπαίδευσης της τεχνητής νοημοσύνης. Στο Armoire, αυτός ο κύκλος είναι κεντρικός: η σελίδα εποπτείας εμφανίζει στατιστικά (ενεργά ρούχα, αποθηκευμένες εμφανίσεις, αφόρετα ρούχα, συνολικές φορές) και υπενθυμίζει, ενθαρρύνοντας τον χρήστη να καταγράψει τη φορά εμφανίσεων. Κάθε καταγραφή ενημερώνει το προφίλ προτιμήσεων, που βελτιώνει τις επόμενες προτάσεις, οι οποίες με τη σειρά τους κάνουν τη σελίδα προτάσεων πιο χρήσιμη. Χωρίς μια σαφή και ελκυστική διεπαφή, ο χρήστης δεν θα κατέγραφε τη φορά, και χωρίς δεδομένα φοράς, οι προτάσεις θα παρέμεναν στην κατάσταση ψυχρής εκκίνησης. Η αμοιβαιότητα που περιγράφει η Virvou δεν είναι λοιπόν αφηρημένη αρχή για το Armoire, αλλά η κινητήρια δύναμη που συνδέει τη διεπαφή με τη μηχανή μάθησης.

Οι Chrysafiadi και Virvou [60] αξιολογούν την εμπειρία χρήστη ενός ευφυούς εκπαιδευτικού συστήματος βασισμένου σε ασαφή λογική, και δείχνουν ότι η αξιολόγηση ευχρηστίας πρέπει να μετράει τόσο αντικειμενική αποδοτικότητα (ο χρήστης ολοκλήρωσε τη διαδρομή;) όσο και υποκειμενική ικανοποίηση (ένιωθε ότι το αποτέλεσμα ήταν σωστό;). Στο Armoire, αυτή η διάκριση λήφθηκε υπόψη στον σχεδιασμό: η διαδρομή μέχρι μια πρόταση εμφάνισης κρατήθηκε σύντομη (επιλογή περίστασης → λήψη αποτελεσμάτων → αποθήκευση), ενώ κάθε πρόταση συνοδεύεται από επεξήγηση, ώστε ο χρήστης να κατανοεί τη λογική πίσω από τις προτάσεις και να μην τις θεωρεί αυθαίρετες.

Οι Mougiakou et al. [61] εξετάζουν τη συγκατάθεση GDPR σε εκπαιδευτικά παιχνίδια που χρησιμοποιούν τεχνητή νοημοσύνη, αναδεικνύοντας ότι η εμπειρία χρήστη πρέπει να λαμβάνει υπόψη τη διαφάνεια στη χρήση προσωπικών δεδομένων. Στο Armoire, η σιωπηρή μάθηση — το σύστημα παρατηρεί τι φοράει ο χρήστης χωρίς να του ζητά ρητή συγκατάθεση για κάθε καταγραφή — δημιουργεί ακριβώς αυτήν την ανάγκη. Γι' αυτό η εφαρμογή εμφανίζει ρητές ενδείξεις όταν το μοντέλο προτιμήσεων είναι ενεργό, ώστε ο χρήστης να αντιλαμβάνεται ότι οι προτάσεις βασίζονται στη συμπεριφορά του. Η ισορροπία μεταξύ εξατομίκευσης και διαφάνειας που τονίζουν οι Mougiakou et al. αποτελεί σχεδιαστική αρχή που εφαρμόζεται σε κάθε σύστημα σύστασης — στη μόδα ίσως ακόμα περισσότερο, καθώς οι προτιμήσεις ρουχισμού θεωρούνται προσωπικές.

Οι Nielsen και Landauer [62] δείχνουν ότι πέντε χρήστες αρκούν για να εντοπιστεί το 85% των προβλημάτων ευχρηστίας — ένα εύρημα ιδιαίτερα χρήσιμο για ένα πτυχιακό έργο, όπου δεν υπάρχει ο προϋπολογισμός για μεγάλες μελέτες χρηστών. Ο Nielsen [63] ορίζει δέκα ευρετικές που χρησιμοποιήθηκαν ως οδηγός κατά τον σχεδιασμό της διεπαφής. Ο Πίνακας 3.2 παρουσιάζει τις τρεις που είναι ιδιαίτερα σημαντικές για το Armoire και πώς εφαρμόζονται.

Πίνακας 3.2. Ευρετικές του Nielsen και εφαρμογή τους στο Armoire

Ευρετική	Πρόβλημα που αντιμετωπίζει	Εφαρμογή στο Armoire
Ορατότητα κατάστασης (Visibility of system status)	Ο χρήστης δεν ξέρει αν το σύστημα ανταποκρίνεται ή αν κάτι πήγε στραβά	Κάθε οθόνη εμφανίζει ενδείξεις φόρτωσης κατά την αναμονή δεδομένων, μηνύματα σφάλματος όταν μια ενέργεια αποτυγχάνει, και αλλαγή κατάστασης κουμπιού κατά τη διάρκεια μιας ενέργειας (π.χ. «Removing...»)
Πρόληψη σφαλμάτων (Error prevention)	Ο χρήστης κάνει μια μη έγκυρη επιλογή και βλέπει ακατανόητο σφάλμα	Η φόρμα προσθήκης ρούχου αποτρέπει μη έγκυρους συνδυασμούς (ίδιο πρωτεύον/δευτερεύον χρώμα, ασύμβατη υποκατηγορία) πριν την υποβολή — ο χρήστης καθοδηγείται, δεν τιμωρείται
Αναγνώριση αντί ανάκλησης (Recognition rather than recall)	Ο χρήστης πρέπει να θυμάται πληροφορίες που δεν είναι ορατές	Κάθε ρούχο εμφανίζεται ως κάρτα με φωτογραφία, χρώμα, κατηγορία, εποχές και περιστάσεις — ο χρήστης αναγνωρίζει οπτικά αντί να ανακαλεί από μνήμη

Ο MacKenzie [64] μαθηματικοποιεί τη δυσκολία επιλογής ενός στόχου στην οθόνη: όσο μικρότερος ο στόχος και μεγαλύτερη η απόσταση, τόσο πιο αργή η αλληλεπίδραση. Στο Armoire, αυτό μεταφράστηκε σε δύο σχεδιαστικές επιλογές: πρώτον, τα κουμπιά ενεργειών (προσθήκη ρούχου, αποθήκευση εμφάνισης, αρχειοθέτηση) είναι σχεδιασμένα ως ευδιάκριτα κουμπιά με επαρκή περιθώρια (padding), ώστε να επιλέγονται εύκολα. Δεύτερον, η κάρτα ρούχου εμφανίζει τη φωτογραφία σε μεγάλο μέγεθος — καθώς η αναγνώριση ρούχου είναι εγγενώς οπτική, ο χρήστης πρέπει να αναγνωρίζει το ρούχο από τη φωτογραφία, όχι από το κείμενο.

Η ενότητα 3.6 αναφέρθηκε στην κενή κατάσταση ως πρόβλημα εμπειρίας χρήστη. Από σκοπιά HCI, η πρόκληση είναι βαθύτερη: η ίδια σελίδα πρέπει να λειτουργεί ως onboarding (όταν η γκαρνταρόμπα είναι κενή), ως εργαλείο εποπτείας (όταν υπάρχουν λίγα ρούχα), και ως dashboard (όταν η γκαρνταρόμπα είναι πλήρης). Στο Armoire, όταν η γκαρνταρόμπα είναι κενή το dashboard εμφανίζει μια κατάσταση κενού περιεχομένου με καθοδήγηση, ενώ όταν υπάρχουν ρούχα εμφανίζει στατιστικά, ανάλυση κατηγοριών/περιστάσεων, υπενθυμίσεις και συμβουλές· το εισαγωγικό μήνυμα προσαρμόζεται σταδιακά ανάλογα με τον αριθμό ρούχων και εμφανίσεων. Αυτή η σταδιακή αποκάλυψη (progressive disclosure) σχετίζεται με το εύρημα των Lindgaard et al. [42] — αυτή τη φορά όχι ως πρώτη εντύπωση, αλλά ως συνεχόμενη εμπειρία πλοήγησης: ο χρήστης δεν βλέπει ποτέ μια υπερβολικά πολύπλοκη οθόνη ούτε μια αδικαιολόγητα κενή, καθώς η πολυπλοκότητα αυξάνεται οργανικά μαζί με τα δεδομένα του.

Οι Khajouei et al. [65] συγκρίνουν δύο μεθόδους αξιολόγησης — ευρετική επιθεώρηση (heuristic evaluation) και γνωστική διερεύνηση (cognitive walkthrough) — και δείχνουν ότι η συνδυαστική χρήση τους εντοπίζει προβλήματα που καμία μέθοδος μόνη δεν θα έβρισκε. Ο συνδυασμός τέτοιων μεθόδων αποτελεί φυσικό επόμενο βήμα για την αξιολόγηση της διεπαφής του Armoire.

3.9 Σύνοψη Κεφαλαίου

Η βιβλιογραφική ανασκόπηση δείχνει ότι το Armoire δεν είναι μια μεμονωμένη ιδέα, αλλά συνδυάζει γνωστές και δοκιμασμένες προσεγγίσεις: αρχές σχεδιασμού APIs και ενοποίησης συστημάτων (§3.1–3.2), έξυπνες γκαρνταρόμπες (§3.3), εφαρμογές οργάνωσης ντουλάπας και ρουχισμού (§3.4), συστήματα συστάσεων και τεχνικές τεχνητής νοημοσύνης (§3.5), σχεδιασμό εμπειρίας χρήστη (§3.6), προσαρμοστικά συστήματα (§3.7), και αρχές αλληλεπίδρασης ανθρώπου-υπολογιστή και ευχρηστίας (§3.8). Τα επόμενα κεφάλαια χτίζουν πάνω σε αυτή τη βάση για να καθορίσουν τις απαιτήσεις, την αρχιτεκτονική και την υλοποίηση του συστήματος.

4. Περιγραφή Προβλήματος

Οι άνθρωποι σήμερα έχουν όλο και περισσότερα ρούχα, αλλά δεν έχουν έναν εύκολο τρόπο να τα διαχειριστούν. Αυτό δημιουργεί τρία βασικά προβλήματα. Πρώτον, ξεχνάμε ρούχα: ένα ρούχο που αγοράστηκε με ενθουσιασμό καταλήγει ξεχασμένο στο βάθος μιας ντουλάπας, και έτσι δεν το φοράμε ξανά — με αποτέλεσμα να αγοράζουμε καινούρια χωρίς λόγο. Δεύτερον, επαναλαμβάνουμε τους ίδιους συνδυασμούς: χωρίς μια καθαρή εικόνα του τι έχουμε, καταλήγουμε πάντα στα ίδια «ασφαλή» ρούχα, χωρίς να δοκιμάζουμε νέους συνδυασμούς. Τρίτον, δεν μπορούμε να φιλτράρουμε εύκολα τα ρούχα ανάλογα με την περίπτωση — π.χ. να δούμε γρήγορα τι έχουμε για τη δουλειά ή για μια επίσημη εκδήλωση.

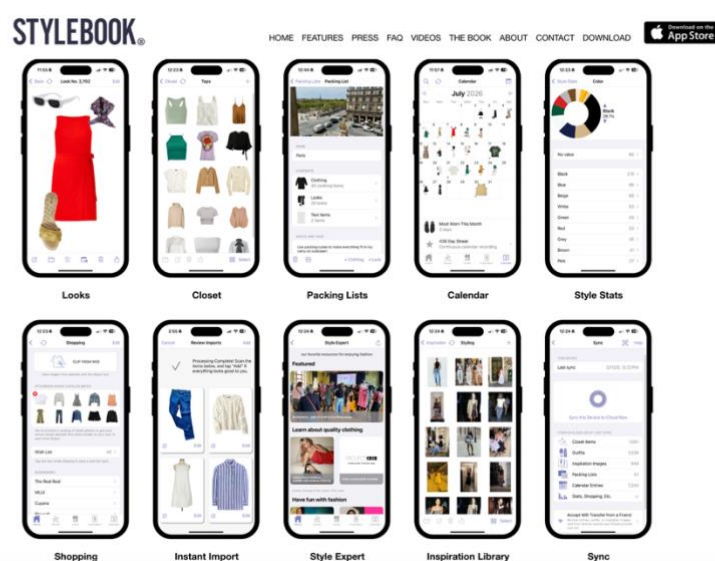
Αυτά τα προβλήματα γίνονται χειρότερα όταν τα ρούχα υπάρχουν μόνο ως μια μεγάλη, ανοργάνωτη συλλογή φωτογραφιών, όπου η αναζήτηση είναι δύσκολη. Οι κοινές εφαρμογές φωτογραφιών ή σημειώσεων δεν βοηθάνε, γιατί δεν ξέρουν τι είναι «ρούχο», «χρώμα» ή «περίσταση». Το Armoire λύνει ακριβώς αυτό το πρόβλημα: όταν ο χρήστης προσθέτει ένα ρούχο, του δίνει αμέσως ετικέτες (κατηγορία, χρώμα, εποχή, περίσταση), ώστε η εφαρμογή —και όχι η μνήμη του χρήστη— να θυμάται τι υπάρχει στη γκαρνταρόμπα.

5. Αναφορά σε Παρόμοιες Εφαρμογές

Υπάρχουν ήδη αρκετές εφαρμογές που ασχολούνται με τη διαχείριση γκαρνταρόμπας και τον σχεδιασμό εμφανίσεων. Η μεγάλη πλειονότητα είναι εφαρμογές κινητού (iOS, Android), κάτι που σημαίνει ότι η χρήση τους γίνεται αποκλειστικά από τηλέφωνο. Το Armoire, αντίθετα, είναι responsive διαδικτυακή εφαρμογή (web application), προσβάσιμη από οποιονδήποτε φυλλομετρητή, σε υπολογιστή αλλά και σε κινητό, χωρίς εγκατάσταση· δεν είναι native εφαρμογή κινητού. Η επιλογή αυτή επιτρέπει χρήση από οποιαδήποτε συσκευή, αξιοποίηση μεγαλύτερης οθόνης για την προβολή φωτογραφιών ρούχων όπου υπάρχει, και ευκολότερη δημιουργία εμφανίσεων, ενώ σχετίζεται με την προσέγγιση των προοδευτικών διαδικτυακών εφαρμογών που εξετάζει ο Desai [6]. Στη συνέχεια παρουσιάζονται τέσσερις από τις πιο γνωστές εφαρμογές, με τις λειτουργίες τους και τα σημεία διαφοροποίησης από το Armoire.

Stylebook (<https://www.stylebookapp.com/>)

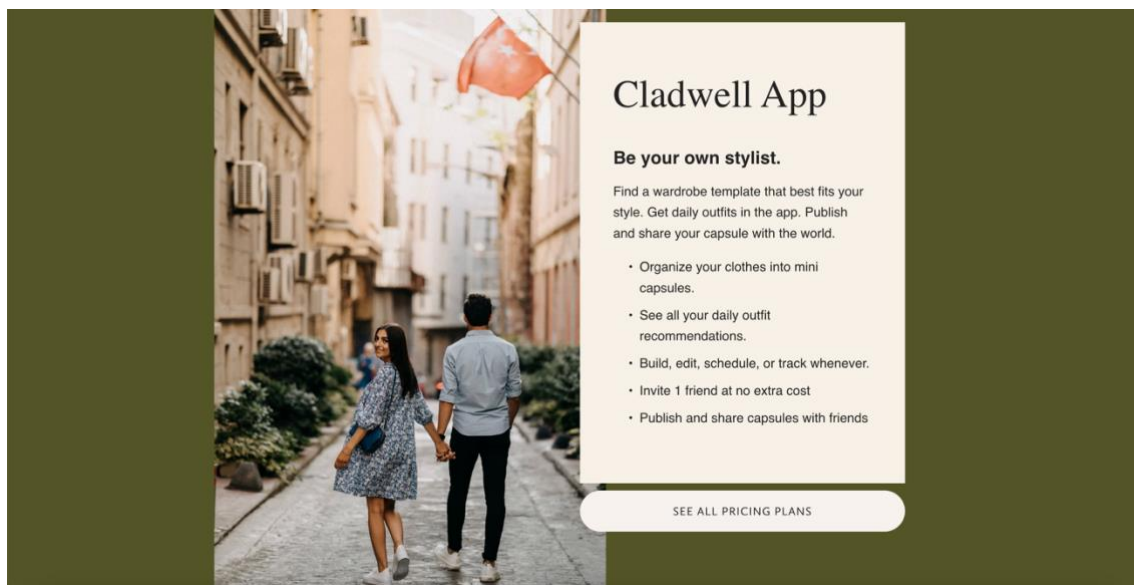
Το Stylebook είναι μία από τις παλαιότερες εφαρμογές γκαρνταρόμπας, διαθέσιμη αποκλειστικά σε iOS και iPad. Ο χρήστης φωτογραφίζει τα ρούχα του, η εφαρμογή αφαιρεί αυτόματα το φόντο με χρήση τεχνητής νοημοσύνης, και τα ρούχα οργανώνονται σε κατηγορίες δύο επιπέδων (φάκελοι και υποκατηγορίες). Η δημιουργία εμφανίσεων γίνεται με μεταφορά ρούχων σε ελεύθερο καμβά, ενώ η λειτουργία Outfit Shuffle ανακατεύει τυχαία τα ρούχα του χρήστη για να αποκαλύψει νέους συνδυασμούς. Η εφαρμογή προσφέρει ημερολόγιο φοράς, στατιστικά κόστους ανά φορά (cost-per-wear), και αναγνώριση ρούχων που δεν έχουν χρησιμοποιηθεί σε καμία εμφάνιση ή δεν έχουν καταγραφεί στο ημερολόγιο. Σε σχέση με το Armoire, η βασική διαφορά είναι ότι οι προτάσεις εμφανίσεων είναι τυχαίες (shuffle) και όχι αλγοριθμικές: δεν λαμβάνεται υπόψη η χρωματική αρμονία μεταξύ των ρούχων, η κατάσταση χρήσης, ή οι προτιμήσεις του χρήστη. Αντίθετα, στο Armoire κάθε πρόταση ελέγχεται για χρωματική συμβατότητα, φιλτράρεται κατά περίπτωση, και κατατάσσεται με βάση το μαθημένο προφίλ στυλ. Επιπλέον, η εφαρμογή απαιτεί εγκατάσταση σε συσκευή Apple, ενώ το Armoire λειτουργεί μέσω φυλλομετρητή σε οποιαδήποτε συσκευή.



Εικόνα 5.1. Στιγμιότυπο της εφαρμογής Stylebook

Cladwell (<https://cladwell.com/>)

Το Cladwell είναι μια εφαρμογή γκαρνταρόμπας που εστιάζει στη φιλοσοφία της capsule wardrobe — μια ελάχιστη, πολυχρηστική γκαρνταρόμπα. Ο χρήστης μπορεί να ξεκινήσει χωρίς να φωτογραφίσει ρούχα, χρησιμοποιώντας έτοιμα πρότυπα γκαρνταρόμπας (wardrobe templates) που στη συνέχεια προσαρμόζει. Η εφαρμογή παρέχει ημερήσιες προτάσεις εμφανίσεων εξατομικευμένες βάσει καιρού και δραστηριότητας, αυτόματη αφαίρεση φόντου από φωτογραφίες, παρακολούθηση κόστους ανά φορά (cost-per-wear), και chatbot στυλιστικών συμβουλών που τροφοδοτείται από το ChatGPT. Η βασική διαχείριση γκαρνταρόμπας είναι δωρεάν, αλλά οι ημερήσιες προτάσεις εμφανίσεων απαιτούν μηνιαία ή ετήσια συνδρομή. Σε σχέση με το Armoire, η βασική διαφορά είναι η αδιαφάνεια στις προτάσεις: ο χρήστης λαμβάνει μια εμφάνιση χωρίς να γνωρίζει αν βασίστηκε στον καιρό, στο στυλ, ή σε κάτι άλλο. Αντίθετα, στο Armoire κάθε πρόταση συνοδεύεται από ρητή επεξήγηση (χρωματική αρμονία, ταίριασμα στυλ, πληρότητα). Επίσης, το Cladwell δεν ελέγχει χρωματική αρμονία μεταξύ ρούχων ούτε εντοπίζει κενά στη γκαρνταρόμπα ανά περίπτωση, ενώ το Armoire είναι πλήρως δωρεάν χωρίς καμία συνδρομή.



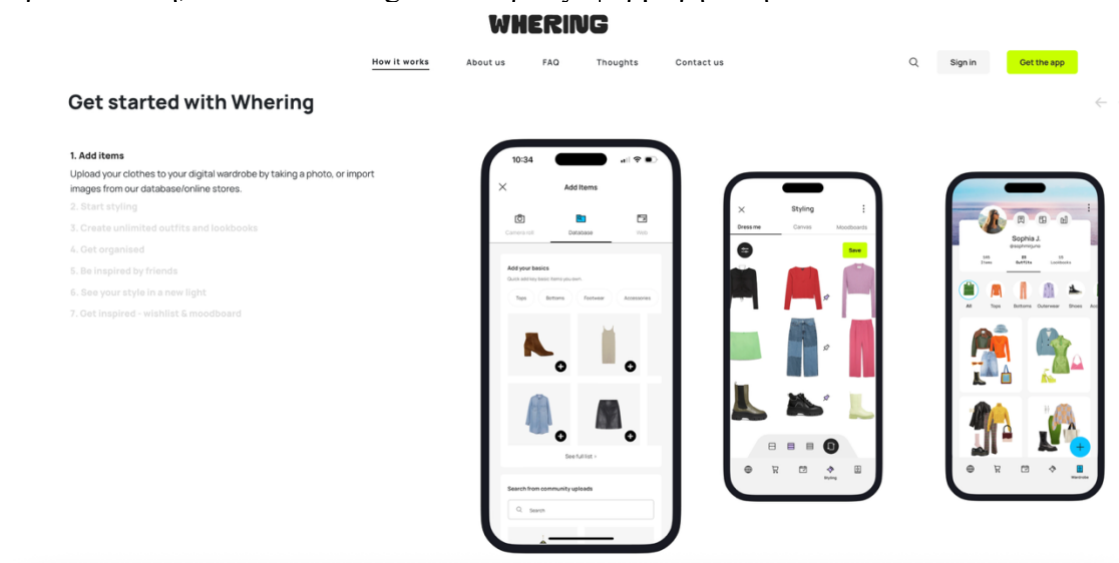
Εικόνα 5.2. Στιγμιότυπο της εφαρμογής Cladwell

Πηγή: <https://cladwell.com/>

Whering (<https://whering.co.uk/>)

Το Whering είναι μια δημοφιλής εφαρμογή γκαρνταρόμπας με πάνω από 10 εκατομμύρια χρήστες, διαθέσιμη σε iOS και Android. Εστιάζει στη βιώσιμη μόδα, βοηθώντας τους χρήστες να αξιοποιήσουν τα ρούχα που ήδη κατέχουν. Ο χρήστης προσθέτει ρούχα μέσω φωτογραφίας, βάσης δεδομένων ρούχων, ή επέκτασης φυλλομετρητή (browser extension). Η εφαρμογή διαθέτει λειτουργία shuffle που

προτείνει τυχαίους συνδυασμούς, καθώς και κοινωνικές λειτουργίες (social features): οι χρήστες μπορούν να μοιράζονται τη γκαρνταρόμπα τους, να βλέπουν πώς ντύνονται οι φίλοι τους, και να δημιουργούν εμφανίσεις για άλλους. Σε σχέση με το Armoire, η βασική διαφορά βρίσκεται στην ποιότητα των προτάσεων: η δημιουργία εμφανίσεων στο Whering βασίζεται σε τυχαία ανακάτεμα (shuffle) — δεν υπάρχει αλγόριθμος που λαμβάνει υπόψη χρωματική αρμονία, περίσταση, ή ιστορικό προτιμήσεων. Αντίθετα, στο Armoire οι προτάσεις φιλτράρονται πρώτα κατά περίσταση, ελέγχονται για χρωματική αρμονία, και κατατάσσονται με βάση τις μαθημένες προτιμήσεις του χρήστη. Επίσης, το Whering δεν διαθέτει σύστημα υπενθυμίσεων για αφόρετα ρούχα ούτε εντοπισμό κενών στη γκαρνταρόμπα, ενώ το Armoire εντοπίζει ρούχα που δεν έχουν φορεθεί εδώ και καιρό και παροτρύνει τον χρήστη να τα δοκιμάσει ή να τα αρχειοθετήσει. Τέλος, το Armoire λειτουργεί εξ ολοκλήρου μέσω φυλλομετρητή χωρίς εγκατάσταση, ενώ το Whering είναι κυρίως εφαρμογή κινητού.

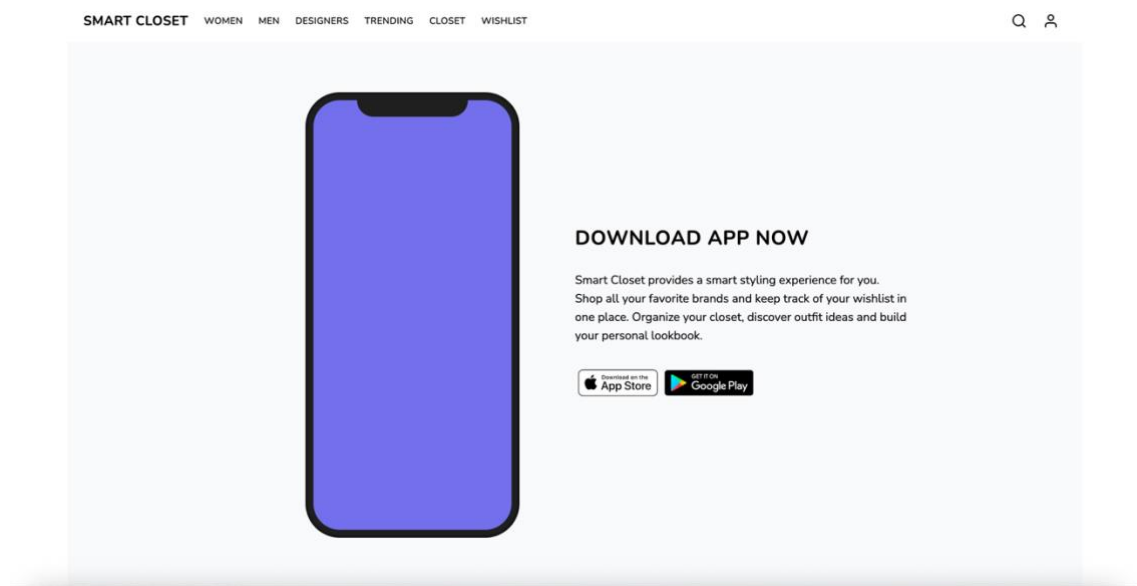


Εικόνα 5.3. Στιγμιότυπο της εφαρμογής Whering

Πηγή: <https://whering.co.uk/>

Smart Closet (<https://smartcloset.me/>)

Το Smart Closet είναι διαθέσιμο σε iOS και Android και συνδυάζει τη διαχείριση γκαρνταρόμπας με ενσωματωμένο κατάστημα μόδας: ο χρήστης μπορεί να περιηγηθεί σε εκατοντάδες μάρκες, να αποθηκεύσει ρούχα σε λίστα επιθυμιών (wishlist), και να καταχωρεί τα δικά του ρούχα με φωτογραφία και ετικέτες. Η εφαρμογή προσφέρει ανακάλυψη ιδεών για εμφανίσεις (outfit ideas) και δημιουργία προσωπικού lookbook. Σε σχέση με το Armoire, η διαφορά είναι θεμελιώδης: το Smart Closet δεν μαθαίνει τίποτα από τις επιλογές του χρήστη. Οι προτεινόμενοι συνδυασμοί δεν λαμβάνουν υπόψη την περίσταση, τη χρωματική αρμονία, ή τις προηγούμενες προτιμήσεις. Αντίθετα, στο Armoire οι προτάσεις βελτιώνονται με τον χρόνο: όσο περισσότερο χρησιμοποιεί ο χρήστης την εφαρμογή (αποθηκεύει εμφανίσεις, καταγράφει φορές), τόσο πιο σχετικές γίνονται οι μελλοντικές προτάσεις. Επιπλέον, το Smart Closet δεν διαθέτει σελίδα εποπτείας (dashboard) με στατιστικά, ούτε σύστημα υπενθυμίσεων για αφόρετα ρούχα. Τέλος, το Armoire λειτουργεί μέσω φυλλομετρητή χωρίς εγκατάσταση, ενώ το Smart Closet είναι εφαρμογή κινητού.



Εικόνα 5.4. Στιγμιότυπο της εφαρμογής Smart Closet

Πηγή: <https://smartcloset.me/>

Πίνακας 5.1. Σύγκριση του Armoire με παρόμοιες εφαρμογές γκαρνταρόμπας

Χαρακτηριστικό	Stylebook	Cladwell	Whering	Smart Closet	Armoire
Πλατφόρμα	iOS	iOS, Android	iOS, Android	iOS, Android	Web
Αυτόματες προτάσεις εμφανίσεων	Τυχαίες (shuffle)	Ναι (καιρός)	Τυχαίες (shuffle)	Όχι	Ναι (αλγοριθμικές)
Έλεγχος χρωματικής αρμονίας	Όχι	Όχι	Όχι	Όχι	Ναι
Φιλτράρισμα ανά περίσταση	Όχι	Όχι	Όχι	Όχι	Ναι
Μάθηση προτιμήσεων χρήστη	Όχι	Όχι	Όχι	Όχι	Ναι
Επεξήγηση προτάσεων	Όχι	Όχι	Όχι	Όχι	Ναι
Υπενθυμίσεις αφόρετων ρούχων	Στατιστικά μόνο	Όχι	Όχι	Όχι	Ναι (στο dashboard)
Εντοπισμός κενών γκαρνταρόμπας	Όχι	Όχι	Όχι	Όχι	Ναι
Κόστος	Εφάπαξ αγορά	Freemium (συνδρομή)	Δωρεάν	Δωρεάν	Δωρεάν

Από τον πίνακα προκύπτει ότι καμία από τις τέσσερις εφαρμογές δεν συνδυάζει: κανόνες περίστασης, έλεγχο χρωματικής αρμονίας, και μάθηση από τις προτιμήσεις του χρήστη — σε μία δωρεάν εφαρμογή. Αυτό ακριβώς προσπαθεί να καλύψει το Armoire. Επιπλέον, το Armoire εμφανίζει υπενθυμίσεις για αφόρετα ρούχα και εντοπίζει κενά στη γκαρνταρόμπα ανά περίσταση — κάτι που δεν προσφέρει καμία από τις εφαρμογές που εξετάστηκαν. Τέλος, λειτουργεί μέσω φυλλομετρητή, χωρίς εγκατάσταση, σε οποιαδήποτε συσκευή.

6. Ανάλυση Απαιτήσεων

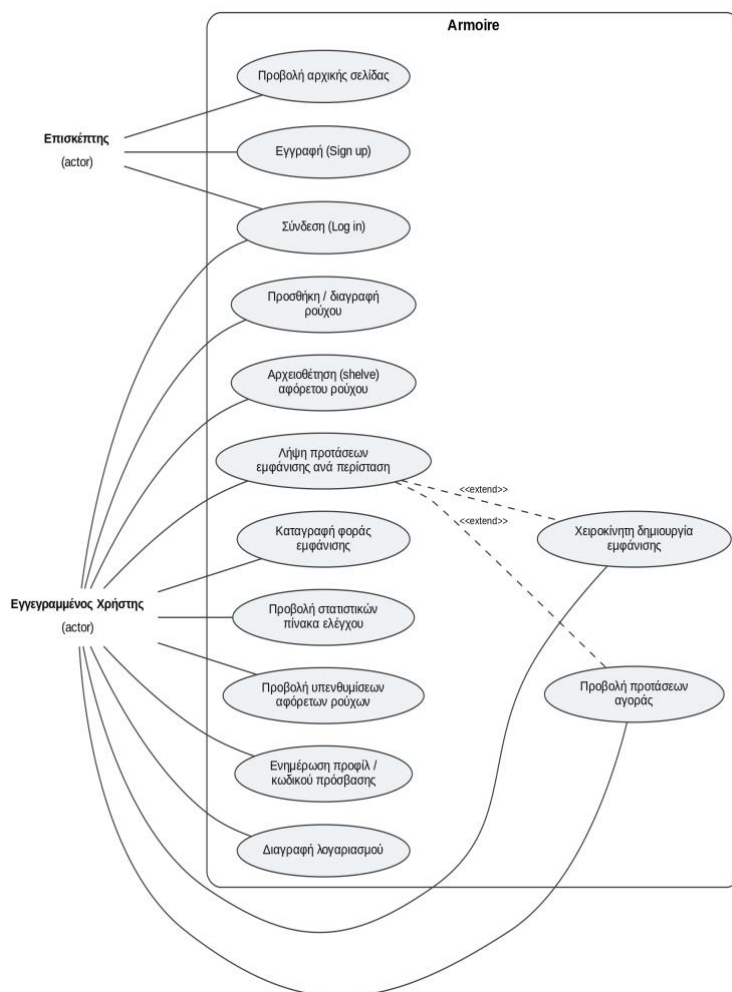
6.1 Ρόλοι Χρηστών και Λειτουργίες (Use-Case Diagram)

Στο Armoire υπάρχουν δύο ρόλοι χρηστών: ο Επισκέπτης, που βλέπει την αρχική σελίδα και μπορεί να εγγραφεί ή να συνδεθεί, και ο Εγγεγραμμένος Χρήστης, που έχει τη δική του γκαρνταρόμπα και εμφανίσεις. Υπάρχει και ένας τρίτος ρόλος, ο Διαχειριστής, που είναι σχεδιασμένος για το μέλλον αλλά δεν υπάρχει ακόμα στην εφαρμογή. Ο Πίνακας 6.1 δείχνει τους ρόλους και τι μπορεί να κάνει ο καθένας.

Πίνακας 6.1. Ρόλοι χρηστών και λειτουργίες

Ρόλος	Περιγραφή	Λειτουργίες
Επισκέπτης	Χρήστης που δεν έχει συνδεθεί	Βλέπει την αρχική σελίδα Εγγράφεται Συνδέεται
Εγγεγραμμένος Χρήστης	Συνδεδεμένος κάτοχος γκαρνταρόμπα	Προσθέτει / διαγράφει ρούχα Φτιάχνει εμφανίσεις με το χέρι Ζητάει προτάσεις ανά περίπτωση Καταγράφει πότε φόρεσε μια εμφάνιση Βλέπει υπενθυμίσεις και αρχειοθετεί ρούχα Βλέπει το dashboard Αλλάζει προφίλ / κωδικό Διαγράφει τον λογαριασμό του
Διαχειριστής (μελλοντικός)	Δεν υπάρχει ακόμα στην εφαρμογή	Έλεγχος περιεχομένου Στατιστικά για όλους τους χρήστες

Η Εικόνα 6.1 δείχνει ένα διάγραμμα περιπτώσεων χρήσης (use-case diagram) που συνοψίζει τι μπορεί να κάνει κάθε ρόλος στο σύστημα.



Εικόνα 6.1. Διάγραμμα περιπτώσεων χρήσης (use-case) του Armoire

6.2 Λειτουργικές και Μη Λειτουργικές Απαιτήσεις

Ο Πίνακας 6.2 δείχνει τις λειτουργικές απαιτήσεις του συστήματος: ποιες είναι ήδη υλοποιημένες (✓) και ποιες είναι σχεδιασμένες για το μέλλον (○).

Πίνακας 6.2. Λειτουργικές απαιτήσεις

#	Απαίτηση	Κατάσταση
FR1	Αρχική σελίδα με παρουσίαση της εφαρμογής	✓ Υλοποιημένη
FR2	Εγγραφή νέου χρήστη	✓ Υλοποιημένη
FR3	Σύνδεση χρήστη με JWT	✓ Υλοποιημένη
FR4	Κατακερματισμός (hashing) κωδικού πριν σταλεί στον server	✓ Υλοποιημένη
FR5	Προσθήκη / λίστα / διαγραφή ρούχων με φωτογραφία και ετικέτες	✓ Υλοποιημένη

#	Απαίτηση	Κατάσταση
FR6	Χειροκίνητη δημιουργία εμφανίσεων	✓ Υλοποιημένη
FR7	Αυτόματες προτάσεις εμφανίσεων ανά περίπτωση	✓ Υλοποιημένη
FR8	Αυστηρό ταίριασμα με την περίπτωση (όχι τυχαίες προτάσεις)	✓ Υλοποιημένη
FR9	Απόρριψη συνδυασμών με ασύμβατα χρώματα	✓ Υλοποιημένη
FR10	Καθαρή εξήγηση όταν δεν βρίσκεται πρόταση	✓ Υλοποιημένη
FR11	Καταγραφή πότε φοριέται μια εμφάνιση	✓ Υλοποιημένη
FR12	Μάθηση προτιμήσεων του χρήστη με τον καιρό	✓ Υλοποιημένη
FR13	Υπενθυμίσεις για αφόρετα ρούχα, με δυνατότητα αρχειοθέτησης	✓ Υλοποιημένη
FR14	Στατιστικά γκαρνταρόμπας σε dashboard	✓ Υλοποιημένη
FR15	Ρυθμίσεις λογαριασμού: προφίλ, κωδικός, διαγραφή	✓ Υλοποιημένη
FR16	Επεξεργασία υπάρχοντος ρούχου	✓ Υλοποιημένη
FR17	Αλλαγή email	○ Σχεδιασμένη
FR18	Αναγνώριση ρούχων από φωτογραφία	○ Σχεδιασμένη
FR19	Αναζήτηση παρόμοιων ρούχων με εικόνα	○ Σχεδιασμένη
FR20	Πλήρη γραφήματα με στατιστικά φοράς	○ Σχεδιασμένη

Εκτός από τις λειτουργικές απαιτήσεις, το σύστημα πρέπει να πληροί και κάποιες βασικές προϋποθέσεις: να λειτουργεί καλά μέσω φυλλομετρητή τόσο σε κινητό όσο και σε υπολογιστή (responsive σχεδιασμός), να έχει ένα καθαρό REST API, να χτίζεται με σύγχρονες τεχνολογίες (Java 21, Spring Boot, React 18), και να μην στέλνει ποτέ κωδικούς πρόσβασης σε απλό κείμενο μέσω δικτύου.

6.3 Περιορισμοί και Τεχνικές Προϋποθέσεις Εγκατάστασης/Χρήσης

Για να τρέξει κανείς το Armoire τοπικά χρειάζεται: Java 21 και Maven για το backend, MySQL 8 για τη βάση δεδομένων, και Node.js 18+ με npm για το frontend. Οι πίνακες της βάσης δεδομένων πρέπει να δημιουργηθούν χειροκίνητα, τρέχοντας τα τέσσερα scripts SQL (001–004). Το backend τρέχει στην πόρτα 8080 και το frontend στην πόρτα 5173. Οι πληροφορίες σύνδεσης με τη βάση δεδομένων αποθηκεύονται σε ένα τοπικό αρχείο ρυθμίσεων που δεν ανεβαίνει στο git. Οι πλήρεις οδηγίες εγκατάστασης

βρίσκονται στο README.md του έργου, ενώ τα scripts SQL αναπαράγονται στο Παράρτημα Α.

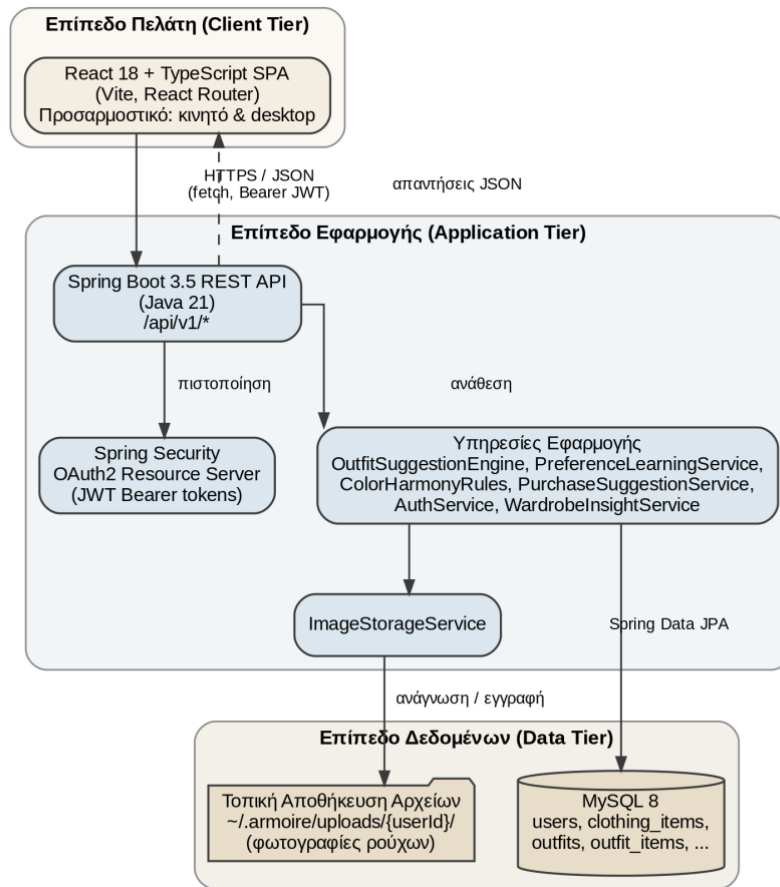
7. Σχεδιασμός και Υλοποίηση

7.1 Αρχιτεκτονική του Συστήματος

Η εφαρμογή Armoire ακολουθεί αρχιτεκτονική τριών επιπέδων (three-tier architecture). Το πρώτο επίπεδο (presentation tier) αποτελείται από μία εφαρμογή React με TypeScript, η οποία εξυπηρετείται μέσω του εργαλείου Vite. Το δεύτερο επίπεδο (application tier) υλοποιείται ως REST API με χρήση Spring Boot 3 και Java 21. Το τρίτο επίπεδο (data tier) αποτελείται από σχεσιακή βάση δεδομένων MySQL, στην οποία η πρόσβαση γίνεται μέσω JPA/Hibernate.

Η επικοινωνία μεταξύ frontend και backend πραγματοποιείται αποκλειστικά μέσω HTTP/JSON. Κάθε αίτημα που απαιτεί αυθεντικοποίηση συνοδεύεται από ένα JWT (JSON Web Token) στην κεφαλίδα Authorization. Ο διακομιστής δεν διατηρεί κατάσταση συνεδρίας (stateless), γεγονός που απλοποιεί τη διαχείριση πόρων και διευκολύνει τη μελλοντική κλιμάκωση.

Στο backend, η δομή του κώδικα ακολουθεί τη λογική του domain-driven design σε τρία πακέτα: `domain` (οντότητες, repositories), `application` (υπηρεσίες επιχειρηματικής λογικής) και `api` (controllers, DTOs). Αυτός ο διαχωρισμός εξασφαλίζει ότι κάθε κλάση έχει μία μόνο ευθύνη και ότι η επιχειρηματική λογική παραμένει ανεξάρτητη από τον τρόπο μεταφοράς δεδομένων (HTTP).



Εικόνα 7.1. Αρχιτεκτονική του συστήματος Armoire (τρία επίπεδα)

7.2 Σχεδιασμός Βάσης Δεδομένων

Η βάση δεδομένων του Armoire αποτελείται από έξι πίνακες που οργανώνονται σε τέσσερα αρχεία migration scripts. Η σχεδιάσή τους ακολουθεί κανονικοποίηση τρίτης κανονικής μορφής (3NF) και αξιοποιεί ξένα κλειδιά (foreign keys) για τη διατήρηση της ακεραιότητας αναφοράς.

7.2.1 Πίνακας users

Ο πίνακας users αποθηκεύει τους λογαριασμούς των χρηστών. Κάθε εγγραφή περιλαμβάνει email (μοναδικό, με UNIQUE constraint), κατακερματισμένο κωδικό (password_hash), όνομα, επώνυμο και χρονοσημάνσεις δημιουργίας/ενημέρωσης.

```

1 CREATE TABLE IF NOT EXISTS users (
2     id            BIGINT AUTO_INCREMENT PRIMARY KEY,
3     email         VARCHAR(255) NOT NULL,
4     password_hash VARCHAR(255) NOT NULL,
5     first_name    VARCHAR(100) NOT NULL,
6     last_name     VARCHAR(100) NOT NULL,
7     created_at    TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
8     updated_at    TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
9     CONSTRAINT uk_users_email UNIQUE (email)
10 );

```

Εικόνα 7.2. Κώδικας SQL δημιουργίας του πίνακα users

Ο κωδικός πρόσβασης δεν αποθηκεύεται ποτέ σε απλό κείμενο· η στήλη `password_hash` περιέχει τιμή `bcrypt`, η οποία περιγράφεται στην ενότητα 7.4. Η χρήση `AUTO_INCREMENT` εξασφαλίζει μοναδικά αναγνωριστικά χωρίς ρίσκο σύγκρουσης, ενώ η στήλη `updated_at` ενημερώνεται αυτόματα σε κάθε τροποποίηση της εγγραφής χάρη στη ρήτρα `ON UPDATE CURRENT_TIMESTAMP`.

7.2.2 Πίνακας `clothing_items` και πίνακες σύνδεσης

Ο πίνακας `clothing_items` αποτελεί τον πυρήνα του μοντέλου δεδομένων. Κάθε ρούχο ανήκει σε έναν χρήστη (foreign key `user_id`), έχει κατηγορία (π.χ. `TOP`, `BOTTOM`, `DRESS`), προαιρετική υποκατηγορία, πρωτεύον και δευτερεύον χρώμα, διαδρομή εικόνας, κατάσταση (`ACTIVE/SHELVED`), μετρητή φορεσιών (`wear_count`) και ημερομηνία τελευταίας χρήσης (`last_worn_at`).

```

1 CREATE TABLE IF NOT EXISTS clothing_items (
2     id            BIGINT AUTO_INCREMENT PRIMARY KEY,
3     user_id       BIGINT NOT NULL,
4     name          VARCHAR(255) NOT NULL,
5     category      VARCHAR(50) NOT NULL,
6     subcategory   VARCHAR(50) NULL,
7     primary_color VARCHAR(50) NOT NULL,
8     secondary_color VARCHAR(50) NULL,
9     image_path    VARCHAR(500) NOT NULL,
10    status        VARCHAR(20) NOT NULL DEFAULT 'ACTIVE',
11    wear_count     INT NOT NULL DEFAULT 0,
12    last_worn_at   TIMESTAMP NULL,
13    notes         VARCHAR(1000) NULL,
14    created_at     TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
15    updated_at     TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
16    CONSTRAINT fk_clothing_items_user FOREIGN KEY (user_id) REFERENCES users (id),
17    INDEX idx_clothing_items_user (user_id),
18    INDEX idx_clothing_items_category (category),
19    INDEX idx_clothing_items_status (status)
20 );

```

Εικόνα 7.3. Κώδικας SQL δημιουργίας του πίνακα `clothing_items`

Σημαντική σχεδιαστική απόφαση είναι η χρήση τριών ευρετηρίων (indexes): ένα στο `user_id` για γρήγορη ανάκτηση της ντουλάπας κάθε χρήστη, ένα στο `category` για φιλτράρισμα κατά κατηγορία, και ένα στο `status` για αποκλεισμό των αρχειοθετημένων (`shelved`) ρούχων. Τα ευρετήρια αυτά βελτιώνουν σημαντικά την απόδοση ερωτημάτων, ειδικά όταν η ντουλάπα ενός χρήστη μεγαλώνει.

Επειδή κάθε ρούχο μπορεί να αντιστοιχεί σε πολλές εποχές και πολλές περιστάσεις (σχέση πολλά-προς-πολλά), δημιουργήθηκαν δύο πίνακες σύνδεσης (junction tables):

```

21 CREATE TABLE IF NOT EXISTS clothing_item_seasons (
22     clothing_item_id BIGINT NOT NULL,
23     season VARCHAR(20) NOT NULL,
24     CONSTRAINT fk_clothing_item_seasons_item FOREIGN KEY (clothing_item_id) REFERENCES clothing_items (id) ON DELETE CASCADE,
25     PRIMARY KEY (clothing_item_id, season)
26 );
27 CREATE TABLE IF NOT EXISTS clothing_item_occasions (
28     clothing_item_id BIGINT NOT NULL,
29     occasion VARCHAR(30) NOT NULL,
30     CONSTRAINT fk_clothing_item_occasions_item FOREIGN KEY (clothing_item_id) REFERENCES clothing_items (id) ON DELETE CASCADE,
31     PRIMARY KEY (clothing_item_id, occasion)
32 );

```

Εικόνα 7.4. Πίνακες σύνδεσης εποχών και περιστάσεων

Η χρήση σύνθετου πρωτεύοντος κλειδιού (composite primary key) στους πίνακες σύνδεσης εγγυάται ότι ένα ρούχο δεν μπορεί να συσχετιστεί δύο φορές με την ίδια εποχή ή περίσταση. Η ρήτρα `ON DELETE CASCADE` εξασφαλίζει ότι όταν διαγράφεται ένα ρούχο, αφαιρούνται αυτόματα και όλες οι εποχές/περιστάσεις του, χωρίς ορφανές εγγραφές.

7.2.3 Πίνακες outfits και outfit_items

Ο πίνακας `outfits` αποθηκεύει τα σύνολα ρούχων. Κάθε σύνολο ανήκει σε έναν χρήστη, έχει όνομα, περίσταση, σημειώσεις, μετρητή χρήσης και ημερομηνία τελευταίας χρήσης. Ο μετρητής χρήσης (`wear_count`) και η ημερομηνία τελευταίας χρήσης (`last_worn_at`) προστέθηκαν σε μεταγενέστερο στάδιο ανάπτυξης μέσω εντολής `ALTER TABLE`, επιτρέποντας την παρακολούθηση συχνότητας χρήσης χωρίς ανακατασκευή του αρχικού πίνακα.

```

1 CREATE TABLE IF NOT EXISTS outfits (
2     id BIGINT AUTO_INCREMENT PRIMARY KEY,
3     user_id BIGINT NOT NULL,
4     name VARCHAR(255) NOT NULL,
5     occasion VARCHAR(30) NOT NULL,
6     notes VARCHAR(1000) NULL,
7     created_at TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
8     updated_at TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
9     CONSTRAINT fk_outfits_user FOREIGN KEY (user_id) REFERENCES users (id),
10    INDEX idx_outfits_user (user_id),
11    INDEX idx_outfits_occasion (occasion)
12 );
13
14 CREATE TABLE IF NOT EXISTS outfit_items (
15     outfit_id BIGINT NOT NULL,
16     clothing_item_id BIGINT NOT NULL,
17     sort_order INT NOT NULL,
18     CONSTRAINT fk_outfit_items_outfit FOREIGN KEY (outfit_id) REFERENCES outfits (id) ON DELETE CASCADE,
19     CONSTRAINT fk_outfit_items_clothing FOREIGN KEY (clothing_item_id) REFERENCES clothing_items (id),
20     PRIMARY KEY (outfit_id, clothing_item_id)
21 );

```

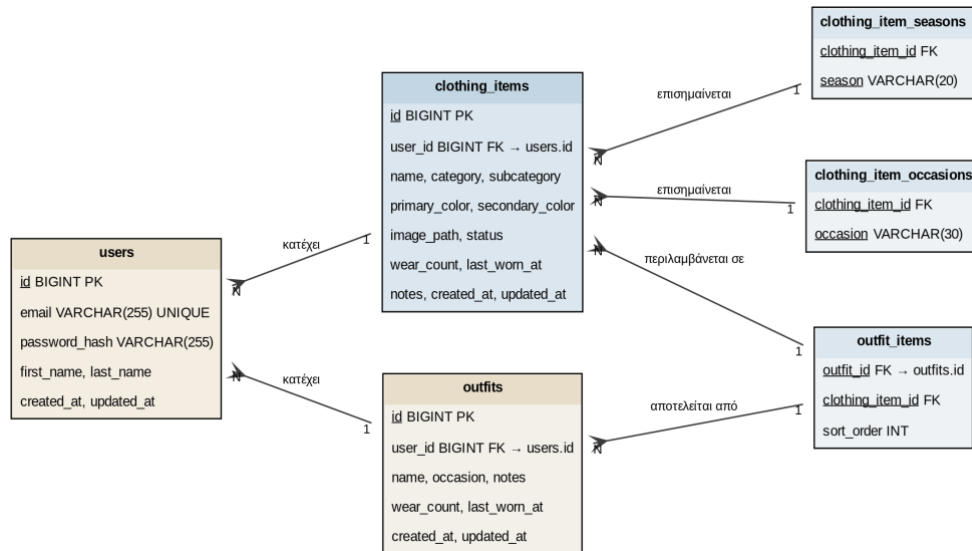
Εικόνα 7.5. Πίνακες outfits και outfit_items

Ο πίνακας `outfit_items` υλοποιεί τη σχέση πολλά-προς-πολλά μεταξύ συνόλων και ρούχων. Η στήλη `sort_order` διατηρεί τη σειρά εμφάνισης των κομματιών. Αξιοσημείωτο είναι ότι η διαγραφή ενός outfit αφαιρεί αυτόματα τις συσχετίσεις (`ON DELETE CASCADE`), αλλά δεν διαγράφει τα ρούχα — αυτά παραμένουν στη ντουλάπα.

Αντίθετα, όταν διαγράφεται ένα ρούχο, διαγράφονται και όλα τα σύνολα που το περιέχουν (μέθοδος `deleteItem`), ώστε να μην υπάρχουν σύνολα με ρούχα που λείπουν.

7.2.4 Διάγραμμα Οντοτήτων-Σχέσεων

Το παρακάτω διάγραμμα απεικονίζει τις σχέσεις μεταξύ των πινάκων.



Εικόνα 7.6. Διάγραμμα Οντοτήτων-Συσχετίσεων (ERD) της βάσης δεδομένων του Armoire

Οι βασικές σχέσεις είναι:

- users** → **clothing_items**: ένα-προς-πολλά (κάθε χρήστης κατέχει πολλά ρούχα)
- users** → **outfits**: ένα-προς-πολλά (κάθε χρήστης δημιουργεί πολλά σύνολα)
- clothing_items** ↔ **seasons**: πολλά-προς-πολλά (ένα ρούχο σε πολλές εποχές)
- clothing_items** ↔ **occasions**: πολλά-προς-πολλά (ένα ρούχο σε πολλές περιστάσεις)
- outfits** ↔ **clothing_items**: πολλά-προς-πολλά μέσω `outfit_items` (ένα σύνολο περιέχει πολλά ρούχα, ένα ρούχο μπορεί σε πολλά σύνολα)

7.3 Αντιστοίχιση Οντοτήτων με JPA/Hibernate

Η Java κλάση `ClothingItem` αντιστοιχεί στον πίνακα `clothing_items` μέσω JPA annotations. Ιδιαίτερο ενδιαφέρον παρουσιάζει ο τρόπος αντιστοίχισης των σχέσεων πολλά-προς-πολλά με εποχές και περιστάσεις: αντί για ξεχωριστές οντότητες, χρησιμοποιείται `@ElementCollection` με `@CollectionTable`, ώστε οι τιμές (enums) να αποθηκεύονται σε πίνακες σύνδεσης χωρίς να χρειάζεται ξεχωριστή JPA entity.

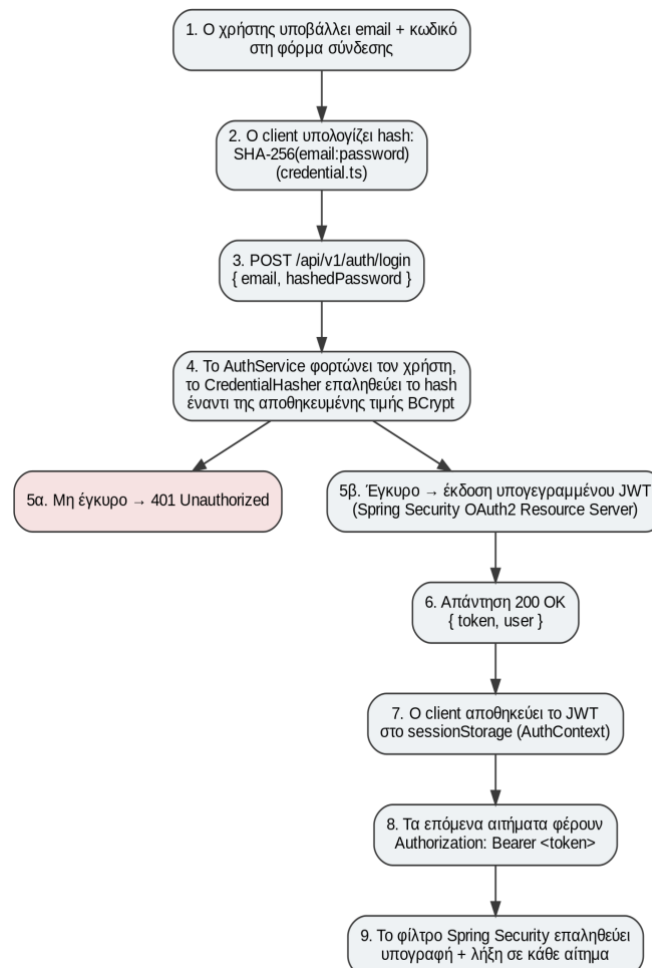
```
19
20 @Entity
21 @Table(name = "clothing_items")
22 @Getter
23 @Setter
24 public class ClothingItem extends BaseEntity {
25
26     @ManyToOne(fetch = FetchType.LAZY, optional = false)
27     @JoinColumn(name = "user_id", nullable = false)
28     private User user;
29
30     @Column(nullable = false)
31     private String name;
32
33     @Enumerated(EnumType.STRING)
34     @Column(nullable = false, length = 50)
35     private ClothingCategory category;
36
37     @Enumerated(EnumType.STRING)
38     @Column(length = 50)
39     private ClothingSubcategory subcategory;
40
41     @Enumerated(EnumType.STRING)
42     @Column(name = "primary_color", nullable = false, length = 50)
43     private ClothingColor primaryColor;
44
45     @Enumerated(EnumType.STRING)
46     @Column(name = "secondary_color", length = 50)
47     private ClothingColor secondaryColor;
48
49     @Column(name = "image_path", nullable = false, length = 500)
50     private String imagePath;
51
52     @Enumerated(EnumType.STRING)
53     @Column(nullable = false, length = 20)
54     private ItemStatus status = ItemStatus.ACTIVE;
55
56     @Column(name = "wear_count", nullable = false)
57     private int wearCount = 0;
```

Εικόνα 7.7. JPA οντότητα ClothingItem: αντιστοίχιση @ElementCollection

Η επιλογή `FetchType.EAGER` για τα `@ElementCollection` σημαίνει ότι εποχές και περιστάσεις φορτώνονται μαζί με κάθε ρούχο κάθε φορά που ανακτάται ένα ρούχο. Αυτό είναι απαραίτητο επειδή στην εφαρμογή είναι απενεργοποιημένο το `open-in-view` (`spring.jpa.open-in-view: false`): χωρίς EAGER φόρτωση, η πρόσβαση στις περιστάσεις μετά το κλείσιμο της συναλλαγής θα προκαλούσε σφάλμα `LazyInitializationException`, ενώ η μηχανή προτάσεων χρειάζεται πρόσβαση στις περιστάσεις κάθε ρούχου κατά την αξιολόγηση συνδυασμών. Η χρήση `FetchType.LAZY` για τη σχέση `@ManyToOne` με τον χρήστη αποφεύγει την άσκοπη φόρτωση δεδομένων χρήστη σε κάθε ανάκτηση ρούχου. Οι κατηγορίες, τα χρώματα, οι εποχές και οι περιστάσεις αποθηκεύονται ως `EnumType.STRING` αντί για `EnumType.ORDINAL`. Αυτή η επιλογή κάνει τα δεδομένα αναγνώσιμα απευθείας στη βάση (π.χ. "TOP" αντί "0") και αποτρέπει σφάλματα σε περίπτωση αλλαγής σειράς στο enum.

7.4 Αυθεντικοποίηση και Ασφάλεια

Η ασφάλεια του Armoire βασίζεται σε σχήμα αυθεντικοποίησης δύο φάσεων: κατακερματισμός κωδικού στον πελάτη (client-side hashing) και επαλήθευση στον διακομιστή μέσω bcrypt.



Εικόνα 7.8. Διάγραμμα ακολουθίας πιστοποίησης (σύνδεση με JWT)

7.4.1 Κατακερματισμός Κωδικού στον Πελάτη

Πριν σταλεί ο κωδικός στον διακομιστή, η εφαρμογή-πελάτης υπολογίζει ένα SHA-256 hash του συνδυασμού email και κωδικού. Ο υπολογισμός γίνεται από τη συνάρτηση hashCredential (ui/src/auth/credential.ts) με το Web Crypto API· η ίδια λογική υπάρχει και στο backend, στην κλάση CredentialHasher (Εικόνα 7.9). Έτσι, ο αρχικός κωδικός του χρήστη δεν εμφανίζεται στα αιτήματα HTTP, κάτι που τον προστατεύει, για παράδειγμα, αν ο χρήστης τον χρησιμοποιεί και σε άλλες υπηρεσίες. Ωστόσο, το hash λειτουργεί πλέον ως ο κωδικός πρόσβασης: όποιος το υποκλέψει μπορεί να το χρησιμοποιήσει για σύνδεση. Για τον λόγο αυτό, σε περιβάλλον παραγωγής η επικοινωνία πρέπει να γίνεται υποχρεωτικά μέσω HTTPS (TLS).

```
public final class CredentialHasher { 1 usage

    public static String hash(String email, String password) {
        try {
            String normalizedEmail = email.trim().toLowerCase();
            MessageDigest digest = MessageDigest.getInstance("SHA-256");
            byte[] hash = digest.digest((normalizedEmail + ":" + password).getBytes(StandardCharsets.UTF_8));
            return HexFormat.of().formatHex(hash);
        } catch (NoSuchAlgorithmException ex) {
            throw new IllegalStateException("SHA-256 not available", ex);
        }
    }
}
```

Εικόνα 7.9. Κατακερματισμός SHA-256 του συνδυασμού email και κωδικού (CredentialHasher)

Η κανονικοποίηση του email (toLowerCase + trim) πριν τον υπολογισμό του hash εγγυάται ότι ο ίδιος κωδικός παράγει πάντα το ίδιο αποτέλεσμα ανεξαρτήτως κεφαλαιοποίησης.

7.4.2 Επαλήθευση στον Διακομιστή

Στο backend, ο κωδικός (ήδη hashed σε SHA-256) αποθηκεύεται περαιτέρω με bcrypt. Κατά τη σύνδεση, η μέθοδος `passwordMatches` ελέγχει δύο σενάρια: (α) η τιμή που στάλθηκε είναι ήδη SHA-256 hash (η κανονική περίπτωση, αφού η εφαρμογή-πελάτης στέλνει πάντα hash) και συγκρίνεται απευθείας με την τιμή bcrypt, και (β) αν σταλεί απλό κείμενο (π.χ. από άλλο πρόγραμμα-πελάτη του API), ο διακομιστής υπολογίζει πρώτα το SHA-256 hash και μετά το συγκρίνει. Και στις δύο περιπτώσεις, στη βάση αποθηκεύεται πάντα το bcrypt του SHA-256 hash.

```
64 boolean passwordMatches(User user, String email, String submitted) { 2 usages
65     String storedHash = user.getPasswordHash();
66
67     if (passwordEncoder.matches(submitted, storedHash)) {
68         return true;
69     }
70
71     if (!CredentialHasher.isSha256Hex(submitted) && passwordEncoder.matches(submitted, storedHash)) {
72         return true;
73     }
74
75     if (!CredentialHasher.isSha256Hex(submitted)) {
76         String hashed = CredentialHasher.hash(email, submitted);
77         if (passwordEncoder.matches(hashed, storedHash)) {
78             return true;
79         }
80     }
81
82     return false;
83 }
```

Εικόνα 7.10. Μέθοδος `passwordMatches`: επαλήθευση κωδικού στον διακομιστή

7.4.3 Έκδοση και Επαλήθευση JWT

Μετά την επιτυχημένη σύνδεση, ο `JwtService` εκδίδει ένα JWT token υπογεγραμμένο με αλγόριθμο HMAC-SHA256. Το token περιέχει claims: issuer ("armoire"), subject (το user ID), email, χρόνο έκδοσης και λήξης (12 ώρες εξ ορισμού).

```

public String generateAccessToken(User user) {
    Instant now = Instant.now().truncatedTo(ChronoUnit.SECONDS);
    JwtClaimsSet claims = JwtClaimsSet.builder()
        .issuer("armoire")
        .subject(user.getId().toString())
        .issuedAt(now)
        .expiresAt(now.plus(expirationHours, ChronoUnit.HOURS))
        .claim("email", user.getEmail())
        .build();

    JwsHeader headers = JwsHeader.with(MacAlgorithm.HS256).build();
    return jwtEncoder.encode(JwtEncoderParameters.from(headers, claims)).getTokenValue();
}

```

Εικόνα 7.11. Μέθοδος generateAccessToken (JwtService)

7.4.4 Διαμόρφωση Spring Security

Ο `SecurityConfig` ορίζει ποια endpoints είναι δημόσια (εγγραφή, σύνδεση, πληροφορίες εφαρμογής, health check) και ποια απαιτούν αυθεντικοποίηση (όλα τα `/api/**`). Η διαχείριση συνεδρίας ορίζεται ως `STATELESS`, γεγονός που σημαίνει ότι ο διακομιστής δεν δημιουργεί HTTP sessions — κάθε αίτημα πρέπει να φέρει έγκυρο JWT.

```

19 http
20     .csrf(AbstractHttpConfigurer::disable)
21     .sessionManagement(session -> session.sessionCreationPolicy(SessionCreationPolicy.STATELESS))
22     .authorizeHttpRequests(auth -> auth
23         .requestMatchers(HttpMethod.OPTIONS, "/**").permitAll()
24         .requestMatchers(
25             "/api/v1/auth/signup",
26             "/api/v1/auth/login",
27             "/api/v1/info"
28         ).permitAll()
29         .requestMatchers("/actuator/health", "/actuator/info").permitAll()
30         .requestMatchers("/api/**").authenticated()
31         .anyRequest().permitAll())
32     .oauth2ResourceServer(oauth2 -> oauth2
33         .jwt(Customizer.withDefaults()))

```

Εικόνα 7.12. SecurityConfig: κανόνες πρόσβασης

Η απενεργοποίηση CSRF είναι σκόπιμη σε stateless REST API, καθώς δεν χρησιμοποιούνται cookies αυθεντικοποίησης — το JWT μεταφέρεται αποκλειστικά στην κεφαλίδα `Authorization`, η οποία δεν αποστέλλεται αυτόματα από τον browser σε cross-site αιτήματα.

7.5 Διαχείριση Εικόνων

Κάθε ρούχο στο Armoire συνοδεύεται υποχρεωτικά από φωτογραφία. Η υπηρεσία `ImageStorageService` υλοποιεί ασφαλή αποθήκευση, ανάκτηση και διαγραφή εικόνων.

7.5.1 Αποθήκευση με Μοναδικά Ονόματα

Κατά τη μεταφόρτωση εικόνας, η υπηρεσία δημιουργεί ένα UUID (Universally Unique Identifier) ως όνομα αρχείου, αντί να χρησιμοποιεί το αρχικό όνομα. Αυτό αποτρέπει:

Armoire: Διαδικτυακή Εφαρμογή Καταλογοποίησης Γκαρνταρόμπας και Προτάσεων Ρουχισμού και Εμφάνισων

(α) συγκρούσεις ονομάτων αν δύο χρήστες ανεβάσουν αρχείο με το ίδιο όνομα, (β) επιθέσεις path traversal μέσω κακόβουλων ονομάτων αρχείων, (γ) αποκάλυψη μεταδεδομένων μέσω αρχικών ονομάτων.

```
public String store(Long userId, MultipartFile file) {
    validateImage(file);

    String extension = EXTENSIONS.get(file.getContentType());
    String filename = UUID.randomUUID() + extension;
    Path userDirectory = rootDirectory.resolve(String.valueOf(userId));
    Path destination = userDirectory.resolve(filename);

    try {
        Files.createDirectories(userDirectory);
        file.transferTo(destination);
        return userId + "/" + filename;
    } catch (IOException ex) {
        throw new ApiException(HttpStatus.INTERNAL_SERVER_ERROR, "Unable to store image");
    }
}
```

Εικόνα 7.13. Μέθοδος store: αποθήκευση εικόνας με όνομα UUID

Οι εικόνες οργανώνονται σε φακέλους ανά χρήστη (π.χ. uploads/42/a1b2c3d4-...-.jpg). Αυτή η δομή διευκολύνει τη διαγραφή όλων των εικόνων ενός χρήστη αν ζητήσει διαγραφή λογαριασμού.

7.5.2 Επικύρωση Τύπου Εικόνας

Πριν αποθηκευτεί η εικόνα, η μέθοδος validateImage ελέγχει ότι ο τύπος αρχείου είναι ένας εκ των JPEG, PNG ή WebP. Αυτός ο περιορισμός αποτρέπει τη μεταφόρτωση εκτελέσιμων αρχείων ή εγγράφων που θα μπορούσαν να χρησιμοποιηθούν κακόβουλα.

```
private void validateImage(MultipartFile file) {
    if (file == null || file.isEmpty()) {
        throw new ApiException(HttpStatus.BAD_REQUEST, "Image is required");
    }

    String contentType = file.getContentType();
    if (contentType == null) {
        throw new ApiException(HttpStatus.BAD_REQUEST, "Image type is required");
    }

    String normalizedType = contentType.toLowerCase(Locale.ROOT);
    if (!ALLOWED_CONTENT_TYPES.contains(normalizedType)) {
        throw new ApiException(HttpStatus.BAD_REQUEST, "Image must be JPEG, PNG, or WebP");
    }
}
```

Εικόνα 7.14. Επικύρωση τύπου εικόνας (validateImage)

7.5.3 Προστασία Path Traversal

Κατά την ανάκτηση ή τη διαγραφή μιας εικόνας, η διαδρομή του αρχείου υπολογίζεται από τη μέθοδο `resolvePath`. Η μέθοδος συνενώνει τη σχετική διαδρομή με τον ριζικό φάκελο αποθήκευσης, την κανονικοποιεί (`normalize`) ώστε να αφαιρεθούν τμήματα όπως «../», και ελέγχει ότι το αποτέλεσμα βρίσκεται μέσα στον ριζικό φάκελο (`startsWith`). Αν η διαδρομή «ξεφεύγει» από τον φάκελο, το αίτημα απορρίπτεται με το σφάλμα «Invalid image path». Έτσι αποτρέπεται η πρόσβαση σε αρχεία εκτός του φακέλου εικόνων (`path traversal`).

```
public Path resolvePath(String relativePath) {
    Path resolved = rootDirectory.resolve(relativePath).normalize();
    if (!resolved.startsWith(rootDirectory)) {
        throw new ApiException(HttpStatus.BAD_REQUEST, "Invalid image path");
    }
    return resolved;
}
```

Εικόνα 7.15. Προστασία από path traversal (`resolvePath`)

7.6 Επικύρωση Ρούχων κατά την Εισαγωγή

Κατά την προσθήκη νέου ρούχου στη ντουλάπα, το σύστημα εκτελεί δύο ελέγχους εγκυρότητας πριν την αποθήκευση:

1. **Συμβατότητα υποκατηγορίας-κατηγορίας:** Αν ο χρήστης επιλέξει υποκατηγορία (π.χ. "SNEAKERS"), αυτή πρέπει να αντιστοιχεί στη σωστή κατηγορία (SHOES). Αν ο χρήστης δηλώσει κατηγορία TOP και υποκατηγορία SNEAKERS, το σύστημα απορρίπτει το αίτημα.
2. **Μοναδικότητα χρωμάτων:** Το δευτερεύον χρώμα πρέπει να διαφέρει από το πρωτεύον. Αν ο χρήστης ορίσει και τα δύο ως BLACK, το σύστημα επιστρέφει σφάλμα.

```
private void validateCreateRequest(CreateClothingItemRequest request) {
    if (request.subcategory() != null && !request.subcategory().matches(request.category())) {
        throw new ApiException(HttpStatus.BAD_REQUEST, "Subcategory does not match category");
    }

    if (request.secondaryColor() != null && request.secondaryColor() == request.primaryColor()) {
        throw new ApiException(HttpStatus.BAD_REQUEST, "Secondary color must differ from primary color");
    }
}
```

Εικόνα 7.16. Επικύρωση στοιχείων ρούχου κατά την εισαγωγή (`validateCreateRequest`)

Ο παρακάτω πίνακας παρουσιάζει τις κατηγορίες, υποκατηγορίες και τη σχέση τους:

Πίνακας 7.1. Αντιστοίχιση κατηγοριών και υποκατηγοριών ρούχων

Κατηγορία	Υποκατηγορίες
TOP	T_SHIRT, SHIRT, BLOUSE, SWEATER, HOODIE, TANK_TOP, POLO
BOTTOM	JEANS, TROUSERS, SHORTS, SKIRT, JOGGERS, LEGGINGS

Κατηγορία	Υποκατηγορίες
DRESS	CASUAL_DRESS, FORMAL_DRESS, JUMPSUIT
OUTERWEAR	JACKET, BLAZER, COAT, VEST, RAINCOAT
SHOES	SNEAKERS, HEELS, LOAFERS, BOOTS, FLATS, SANDALS
ACCESSORY	BAG, HAT, SCARF, BELT, JEWELRY, SUNGLASSES

7.7 Κανόνες Σύνθεσης Συνόλου (Outfit Composition)

Η δημιουργία ενός συνόλου ρούχων δεν είναι μια απλή αντιστοίχιση — υπόκειται σε αυστηρούς κανόνες που εξασφαλίζουν ότι κάθε σύνολο αποτελεί μια πρακτικά ορθή ενδυματολογική επιλογή. Η μέθοδος `validateOutfitComposition` εφαρμόζει τέσσερις ελέγχους σε κάθε νέο σύνολο:

Κανόνας 1 — Κατάσταση ρούχων (status check): Μόνο ενεργά ρούχα μπορούν να χρησιμοποιηθούν. Ρούχα που έχουν μεταφερθεί σε κατάσταση «αρχειοθετημένα» (SHELVED) αποκλείονται αυτόματα. Αυτός ο κανόνας αποτρέπει τη δημιουργία συνόλων με ρούχα που ο χρήστης έχει σκόπιμα αφαιρέσει από την ενεργή του ντουλάπα.

Κανόνας 2 — Αμοιβαίος αποκλεισμός φορέματος και top+bottom (dress XOR rule): Ένα σύνολο μπορεί να περιλαμβάνει *είτε* φόρεμα *είτε* συνδυασμό πάνω+κάτω, αλλά ποτέ και τα δύο. Αυτός ο κανόνας αντικατοπτρίζει την πραγματικότητα της ένδυσης: ένα φόρεμα αντικαθιστά τόσο το πάνω όσο και το κάτω μέρος.

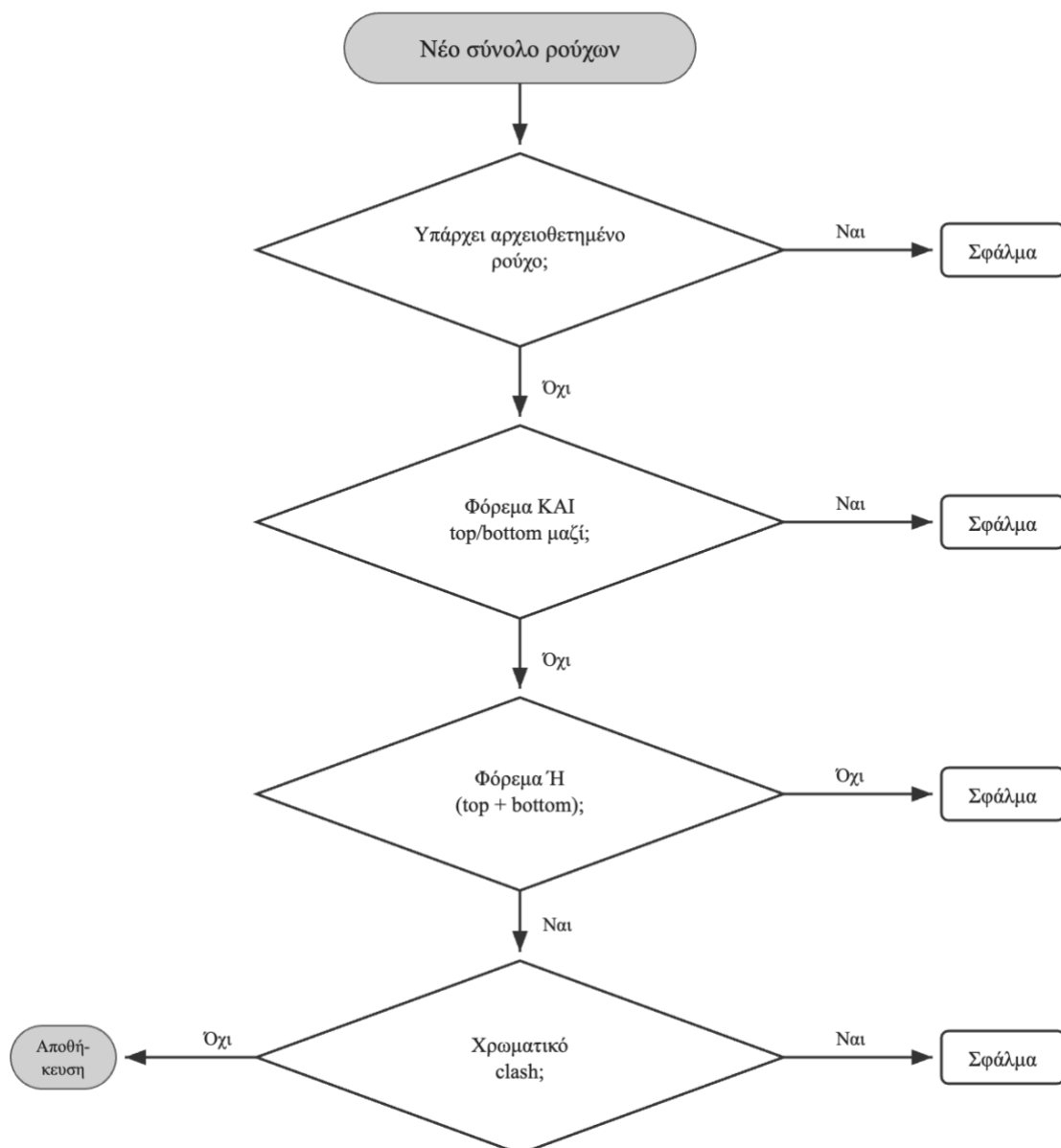
Κανόνας 3 — Ελάχιστη βάση (minimum base rule): Κάθε σύνολο πρέπει να έχει ένα φόρεμα ή τουλάχιστον ένα πάνω και ένα κάτω κομμάτι. Δεν επιτρέπονται σύνολα μόνο από παπούτσια ή μόνο από πανωφόρι. Επιπλέον, η μέθοδος `validateCreateRequest` απαιτεί τουλάχιστον δύο ρούχα χωρίς διπλότυπα: έτσι, ένα φόρεμα αποθηκεύεται μόνο σε συνδυασμό με παπούτσια ή πανωφόρι.

Κανόνας 4 — Έλεγχος χρωματικής αρμονίας (color clash check): Αν τα χρώματα των ρούχων δημιουργούν γνωστό χρωματικό clash (βλ. ενότητα 7.10), το σύνολο απορρίπτεται. Αυτός ο έλεγχος χρησιμοποιεί τις ίδιες ακριβώς αντιστοιχίσεις που χρησιμοποιεί και η μηχανή προτάσεων, εξασφαλίζοντας συνέπεια μεταξύ χειροκίνητης δημιουργίας και αυτόματων προτάσεων.

```
private void validateOutfitComposition(List<ClothingItem> items) {  
    for (ClothingItem item : items) {  
        if (item.getStatus() != ItemStatus.ACTIVE) {  
            throw new ApiException(HttpStatus.BAD_REQUEST, "Shelved items cannot be used in outfits");  
        }  
    }  
  
    boolean hasDress = items.stream().anyMatch(item -> item.getCategory() == ClothingCategory.DRESS);  
    boolean hasTop = items.stream().anyMatch(item -> item.getCategory() == ClothingCategory.TOP);  
    boolean hasBottom = items.stream().anyMatch(item -> item.getCategory() == ClothingCategory.BOTTOM);  
  
    if (hasDress && (hasTop || hasBottom)) {  
        throw new ApiException(  
            HttpStatus.BAD_REQUEST,  
            "Choose either a dress or a top and bottom combination, not both"  
        );  
    }  
  
    if (!hasDress && !(hasTop && hasBottom)) {  
        throw new ApiException(  
            HttpStatus.BAD_REQUEST,  
            "Outfits need a dress or both a top and a bottom"  
        );  
    }  
  
    if (ColorHarmonyRules.comboHasClash(items)) {  
        throw new ApiException(  
            HttpStatus.BAD_REQUEST,  
            "These items have clashing colors and cannot be combined"  
        );  
    }  
}
```

Εικόνα 7.17. Κανόνες σύνθεσης συνόλου (validateOutfitComposition)

Το παρακάτω διάγραμμα ροής απεικονίζει τη σειρά των ελέγχων:



Εικόνα 7.18. Διάγραμμα ροής επικύρωσης συνόλου

7.8 Αλυσιδωτή Καταγραφή Χρήσης (Cascading Wear Recording)

Όταν ο χρήστης δηλώνει ότι φόρεσε ένα σύνολο, η εφαρμογή ενημερώνει ταυτόχρονα τον μετρητή χρήσης και τις ημερομηνίες τόσο του συνόλου όσο και κάθε μεμονωμένου ρούχου που περιέχει. Αυτή η αλυσιδωτή ενημέρωση (cascading update) εξασφαλίζει ότι

τα στατιστικά χρήσης παραμένουν συνεπή χωρίς να απαιτείται ο χρήστης να δηλώσει τη χρήση κάθε ρούχου ξεχωριστά.

```
@Transactional
public OutfitResponse recordWear(Long userId, Long outfitId) {
    Outfit outfit = requireOwnedOutfit(userId, outfitId);
    java.time.Instant now = java.time.Instant.now();

    outfit.setWearCount(outfit.getWearCount() + 1);
    outfit.setLastWornAt(now);

    for (ClothingItem item : outfit.getItems()) {
        item.setWearCount(item.getWearCount() + 1);
        item.setLastWornAt(now);
    }

    return OutfitResponse.from(outfitRepository.save(outfit));
}
```

Εικόνα 7.19. Αλυσιδωτή καταγραφή χρήσης (recordWear)

Η χρήση `@Transactional` εγγυάται ότι η ενημέρωση του συνόλου και όλων των ρούχων του γίνεται ατομικά: αν κάτι αποτύχει, καμία αλλαγή δεν αποθηκεύεται. Τα δεδομένα χρήσης τροφοδοτούν τρεις άλλες λειτουργίες:

- Τη **μηχανή εκμάθησης προτιμήσεων** (ενότητα 7.12), που δίνει μεγαλύτερο βάρος σε ρούχα που φοριούνται συχνά.
- Τις **υπενθυμίσεις ντουλάπας** (ενότητα 7.15), που εντοπίζουν ρούχα που δεν έχουν χρησιμοποιηθεί εδώ και καιρό.
- Τα **στατιστικά dashboard** (ενότητα 7.21), που εμφανίζουν τη συνολική χρήση.

7.9 Μηχανή Πρότασης Συνόλων — Pipeline Πέντε Φάσεων

Η μηχανή πρότασης συνόλων (`OutfitSuggestionEngine`) αποτελεί τον πυρήνα ευφυΐας της εφαρμογής. Ο αλγόριθμος ακολουθεί pipeline πέντε φάσεων:

Φάση 1 — Φιλτράρισμα ανά περίπτωση: Από τη ντουλάπα του χρήστη, επιλέγονται μόνο τα ενεργά ρούχα (ACTIVE) που φέρουν ετικέτα για τη ζητούμενη περίπτωση. Αν δεν βρεθούν ρούχα, επιστρέφεται κατάσταση `NO_OCCASION_ITEMS`.

Φάση 2 — Δημιουργία βάσεων (base generation): Κατασκευάζονται όλοι οι δυνατοί βασικοί συνδυασμοί: κάθε φόρεμα μόνο του, και κάθε ζεύγος πάνω×κάτω. Αν δεν σχηματίζεται καμία βάση, επιστρέφεται `INSUFFICIENT_ITEMS`.

Φάση 3 — Επέκταση με προαιρετικά κομμάτια: Κάθε βάση επεκτείνεται με παπούτσια και/ή πανωφόρι. Π.χ. αν μια βάση είναι [μπλούζα, παντελόνι], παράγονται: [μπλούζα, παντελόνι], [μπλούζα, παντελόνι, σακάκι], [μπλούζα, παντελόνι, παπούτσια], [μπλούζα, παντελόνι, σακάκι, παπούτσια]. Στην τρέχουσα υλοποίηση χρησιμοποιείται μόνο το πρώτο πανωφόρι και τα πρώτα παπούτσια της περίπτωσης (τα πιο πρόσφατα προστεθειμένα), οπότε δεν δοκιμάζονται συνδυασμοί με διαφορετικά πανωφόρια ή παπούτσια.

Φάση 4 — Χρωματικός έλεγχος: Κάθε υπονήφιος συνδυασμός ελέγχεται για χρωματικά clashes. Οι συνδυασμοί με clashes αποκλείονται.

Φάση 5 — Βαθμολόγηση και κατάταξη: Οι εγκεκριμένοι συνδυασμοί βαθμολογούνται (βλ. ενότητα 7.11) και επιστρέφονται οι 3 κορυφαίοι.

```
public OutfitSuggestionsResultResponse suggest(Long userId, List<ClothingItem> wardrobe, Occasion occasion) {
    UserPreferenceProfile profile = preferenceLearningService.buildProfile(userId);

    List<ClothingItem> activeItems = wardrobe.stream()
        .filter(item -> item.getStatus() == ItemStatus.ACTIVE)
        .toList();

    List<ClothingItem> occasionItems = activeItems.stream()
        .filter(item -> item.getOccasions().contains(occasion))
        .toList();

    if (occasionItems.isEmpty()) {
        SuggestionContextResponse context = SuggestionContextBuilder.build(
            wardrobe, occasionItems, occasion, profile, List.of()
        );

        return emptyResult(
            occasion,
            SuggestionStatus.NO_OCCASION_ITEMS,
            "No %s pieces in your wardrobe".formatted(formatOccasion(occasion)),
            buildNoOccasionMessage(activeItems.size(), occasion),
            "Tag existing items with the %s occasion when adding or editing them, or shop for pieces below."
                .formatted(formatOccasion(occasion)),
            context,
            wardrobe,
            profile
        );
    }

    if (!canFormBaseOutfit(occasionItems)) {
        SuggestionContextResponse context = SuggestionContextBuilder.build(
            wardrobe, occasionItems, occasion, profile, List.of()
        );
    }
}
```

Εικόνα 7.20. Μηχανή πρότασης: τμήμα του pipeline πέντε φάσεων (suggest)

7.10 Κανόνες Χρωματικής Αρμονίας

Η κλάση `ColorHarmonyRules` ορίζει τους κανόνες χρωματικής συμβατότητας. Τα χρώματα χωρίζονται σε δύο ομάδες: ουδέτερα (neutrals) που ταιριάζουν με τα πάντα, και έντονα (bold) που μπορούν να δημιουργήσουν clashes μεταξύ τους. Τα 6 ουδέτερα χρώματα είναι: μαύρο, λευκό, γκρι, μπεζ, navy, καφέ. Αυτά ταιριάζουν με κάθε χρώμα χωρίς εξαίρεση. Για τα έντονα χρώματα, ορίζονται 7 σύνολα ασυμβατοτήτων βάσει θεωρίας χρωματικού τροχού. Π.χ. το κόκκινο συγκρούεται με πράσινο, πορτοκαλί, ροζ και μοβ· το μπλε μόνο με πορτοκαλί.

```

static boolean colorsClash(ClothingColor first, ClothingColor second) {
    if (first == second) {
        return false;
    }
    if (isNeutral(first) || isNeutral(second)) {
        return false;
    }

    return clashesOneWay(first, second) || clashesOneWay(second, first);
}

```

Εικόνα 7.21. Κανόνες χρωματικής αρμονίας (ColorHarmonyRules)

Ο έλεγχος είναι αμφίδρομος (bidirectional): αν A clash με B, τότε και B clash με A. Αυτό υλοποιείται μέσω της μεθόδου `clashesOneWay` που ελέγχεται και προς τις δύο κατευθύνσεις.

Κατά τον έλεγχο ενός συνόλου, η μέθοδος `findClashingPairs` εξετάζει κάθε ζεύγος ρούχων ($O(n^2)$), λαμβάνοντας υπόψη τόσο το πρωτεύον όσο και το δευτερεύον χρώμα κάθε ρούχου.

Ο παρακάτω πίνακας συνοψίζει τις χρωματικές ασυμβατότητες:

Χρώμα	Ασυμβατότητα με
Κόκκινο	Πίνακας 7.2. Χρωματικές ασυμβατότητες Πράσινο, Πορτοκαλί, Ροζ, Μοβ
Πορτοκαλί	Μοβ, Ροζ, Κόκκινο, Μπλε
Μοβ	Πορτοκαλί, Κίτρινο, Κόκκινο
Πράσινο	Κόκκινο, Ροζ
Μπλε	Πορτοκαλί
Κίτρινο	Μοβ
Ροζ	Κόκκινο, Πορτοκαλί, Πράσινο

7.11 Σύστημα Βαθμολόγησης Συνδυασμών

Κάθε υποψήφιος συνδυασμός αξιολογείται αριθμητικά. Η βαθμολογία υπολογίζεται αθροιστικά: Για τη χρωματική αρμονία λαμβάνεται υπόψη μόνο το κύριο χρώμα κάθε ρούχου.

Κριτήριο	Πόντοι
Βάση ανά ρούχο (κάθε κομμάτι στο σύνολο)	+12 ανά ρούχο
Μπόνους ουδέτερου χρώματος	+4 ανά ουδέτερο ρούχο

Κριτήριο	Πόντοι
Χρωματική αρμονία (≤ 2 χρώματα)	+8
Χρωματική αρμονία (3+ χρώματα, όλα ουδέτερα)	+10
Χρωματική αρμονία (3 χρώματα, με μη ουδέτερο)	+2
Χρωματική αρμονία (4+ χρώματα, με μη ουδέτερο)	-4
Περιλαμβάνει παπούτσια	+3
Περιλαμβάνει πανωφόρι	+2
Cosine similarity $\times 15$ (αν υπάρχει προφίλ)	0 – 15
Jaccard similarity $\times 10$ (αν υπάρχει προφίλ)	0 – 10

```

private int scoreCombination(
    Long userId,
    List<ClothingItem> items,
    Occasion occasion,
    UserPreferenceProfile profile
) {
    int score = 0;

    for (ClothingItem item : items) {
        score += 12;
        if (NEUTRAL_COLORS.contains(item.getPrimaryColor())) {
            score += 4;
        }
    }

    score += colorHarmonyScore(items);

    if (items.stream().anyMatch(item -> item.getCategory() == ClothingCategory.SHOES)) {
        score += 3;
    }
    if (items.stream().anyMatch(item -> item.getCategory() == ClothingCategory.OUTERWEAR)) {
        score += 2;
    }

    if (!profile.isEmpty()) {
        double cosine = profile.comboSimilarity(items);
        score += (int) Math.round(cosine * MAX_PREFERENCE_BOOST);

        double jaccard = preferenceLearningService.savedOutfitJaccard(userId, items);
        score += (int) Math.round(jaccard * MAX_JACCARD_BOOST);
    }

    return score;
}

```

Εικόνα 7.22. Μέθοδος scoreCombination: υπολογισμός βαθμολογίας

Η αρχιτεκτονική βαθμολόγησης είναι αθροιστική και κλιμακωτή. Ένα σύνολο 2 ρούχων ξεκινά με 24 πόντους, ενώ ένα πλήρες σύνολο 4 ρούχων (πάνω, κάτω, παπούτσια, πανωφόρι) ξεκινά με 53+ πόντους. Αυτό εξασφαλίζει ότι πληρέστερα σύνολα κατατάσσονται υψηλότερα. Η cosine και Jaccard similarity εξηγούνται αναλυτικά στις επόμενες ενότητες.

7.12 Εκμάθηση Προτιμήσεων Χρήστη

Η υπηρεσία `PreferenceLearningService` κατασκευάζει ένα προφίλ γούστου για κάθε χρήστη, χρησιμοποιώντας content-based filtering με TF-weighted feature vectors. Το προφίλ ενημερώνεται δυναμικά και τροφοδοτεί τη βαθμολόγηση συνδυασμών.

Η κατασκευή βασίζεται σε δύο πηγές πληροφορίας:

1. **Αποθηκευμένα σύνολα (saved outfits):** Κάθε ρούχο μέσα σε αποθηκευμένο σύνολο συνεισφέρει βάρος 1.0 στο προφίλ.
2. **Ιστορικό χρήσης (wear count):** Κάθε ρούχο που έχει φορεθεί συνεισφέρει βάρος 2.0 × αριθμό χρήσεων. Η πραγματική χρήση θεωρείται ισχυρότερο σήμα προτίμησης από την αποθήκευση.

```
public UserProfile buildProfile(Long userId) {
    Map<String, Double> weights = new HashMap<>();
    Set<Long> savedItemIds = new HashSet<>();

    List<Outfit> outfits = outfitRepository.findByUserIdOrderByCreatedAtDesc(userId);
    for (Outfit outfit : outfits) {
        for (ClothingItem item : outfit.getItems()) {
            savedItemIds.add(item.getId());
            PreferenceFeatures.merge(
                weights,
                PreferenceFeatures.itemVector(item, SAVED_OUTFIT_WEIGHT)
            );
        }
    }

    List<ClothingItem> wardrobe = clothingItemRepository.findByUserIdOrderByCreatedAtDesc(userId);
    for (ClothingItem item : wardrobe) {
        if (item.getWearCount() <= 0) {
            continue;
        }
        double wearBoost = WEAR_WEIGHT * item.getWearCount();
        PreferenceFeatures.merge(weights, PreferenceFeatures.itemVector(item, wearBoost));
    }

    return UserProfile.fromWeightedFeatures(weights);
}
```

Εικόνα 7.23. Κατασκευή προφίλ προτιμήσεων (buildProfile)

7.12.1 Feature Vectors

Κάθε ρούχο μετατρέπεται σε αραιό διάνυσμα χαρακτηριστικών (sparse feature vector) από τη μέθοδο `itemVector`. Τα χαρακτηριστικά περιλαμβάνουν:

- **Κατηγορία** (βάρος 1.0): π.χ. "category:TOP" → 1.0
- **Πρωτεύον χρώμα** (βάρος 1.0): π.χ. "color:BLACK" → 1.0
- **Δευτερεύον χρώμα** (βάρος 0.5): μισή συνεισφορά, αντανακλά τη μικρότερη σημασία
- **Περιστάσεις** (βάρος 1.0): π.χ. "occasion:WORK" → 1.0
- **Εποχές** (βάρος 0.5): π.χ. "season:WINTER" → 0.5

```
static Map<String, Double> itemVector(ClothingItem item, double weight) {  
    Map<String, Double> vector = new HashMap<>();  
    add(vector, categoryKey(item.getCategory()), weight);  
    add(vector, colorKey(item.getPrimaryColor()), weight);  
    if (item.getSecondaryColor() != null) {  
        add(vector, colorKey(item.getSecondaryColor()), weight * 0.5);  
    }  
    for (Occasion occasion : item.getOccasions()) {  
        add(vector, occasionKey(occasion), weight);  
    }  
    for (Season season : item.getSeasons()) {  
        add(vector, seasonKey(season), weight * 0.5);  
    }  
    return vector;  
}
```

Εικόνα 7.24. Δημιουργία διανύσματος χαρακτηριστικών ανά ρούχο (itemVector)

7.13 Cosine Similarity — Ταίριασμα Στυλ

Για τη σύγκριση ενός υποψήφιου συνόλου με το προφίλ του χρήστη, χρησιμοποιείται ομοιότητα συνημιτόνου (cosine similarity). Η μέτρηση δίνει τιμές μεταξύ 0 (πλήρης ασυμφωνία) και 1 (πλήρης ταύτιση).

```
private double cosineSimilarity(Map<String, Double> otherVector) {  
    if (otherVector.isEmpty() || magnitude == 0.0) {  
        return 0.0;  
    }  
  
    double dot = 0.0;  
    for (Map.Entry<String, Double> entry : otherVector.entrySet()) {  
        Double weight = weights.get(entry.getKey());  
        if (weight != null) {  
            dot += weight * entry.getValue();  
        }  
    }  
  
    double otherMagnitude = computeMagnitude(otherVector);  
    if (otherMagnitude == 0.0) {  
        return 0.0;  
    }  
  
    return dot / (magnitude * otherMagnitude);  
}
```

Εικόνα 7.25. Υπολογισμός ομοιότητας συνημιτόνου (cosine similarity)

Για ένα σύνολο πολλών ρούχων, η μέθοδος `comboSimilarity` υπολογίζει τον μέσο όρο cosine similarity όλων των ρούχων του συνόλου. Αυτή η προσέγγιση εξασφαλίζει ότι

ένα σύνολο δεν ευνοείται μόνο επειδή ένα μεμονωμένο κομμάτι του ταιριάζει στο προφίλ. Αν η τιμή ξεπεράσει το κατώφλι 0.35, εμφανίζεται ειδικό μήνυμα «Aligns with your saved style» στην πρόταση.

7.14 Jaccard Similarity — Ομοιότητα με Αποθηκευμένα Σύνολα

Παράλληλα με τη cosine similarity, χρησιμοποιείται Jaccard similarity για τη σύγκριση ενός υποψήφιου συνόλου με τα ήδη αποθηκευμένα σύνολα του χρήστη. Ενώ η cosine μετρά ομοιότητα χαρακτηριστικών, η Jaccard μετρά ομοιότητα σύνθεσης: πόσα από τα ίδια ρούχα εμφανίζονται στα δύο σύνολα.

```
public double savedOutfitJaccard(Long userId, List<ClothingItem> candidateItems) {
    Set<Long> candidateIds = candidateItems.stream()
        .map(ClothingItem::getId)
        .collect(java.util.stream.Collectors.toSet());

    if (candidateIds.isEmpty()) {
        return 0.0;
    }

    double best = 0.0;
    for (Outfit outfit : outfitRepository.findByIdOrderByCreatedAtDesc(userId)) {
        Set<Long> savedIds = outfit.getItems().stream()
            .map(ClothingItem::getId)
            .collect(java.util.stream.Collectors.toSet());

        Set<Long> intersection = new HashSet<>(candidateIds);
        intersection.retainAll(savedIds);

        Set<Long> union = new HashSet<>(candidateIds);
        union.addAll(savedIds);

        if (union.isEmpty()) {
            continue;
        }

        double jaccard = (double) intersection.size() / union.size();
        best = Math.max(best, jaccard);
    }

    return best;
}
```

Εικόνα 7.26. Υπολογισμός ομοιότητας Jaccard (savedOutfitJaccard)

Η μέθοδος επιστρέφει τη μέγιστη Jaccard similarity μεταξύ του υποψήφιου και όλων των αποθηκευμένων συνόλων. Π.χ. αν ο χρήστης έχει αποθηκευμένο σύνολο {A, B, C} και ο υποψήφιος είναι {A, B, D}, τότε η ομοιότητα είναι $|\{A,B\}|/|\{A,B,C,D\}| = 2/4 = 0.5$. Αυτό μεταφράζεται σε +5 πόντους ($0.5 \times \text{MAX_JACCARD_BOOST} = 10$).

7.15 Υπενθυμίσεις και Αξιολογήσεις Ντουλάπας (Wardrobe Insights)

Η υπηρεσία `WardrobeInsightService` αναλύει τη ντουλάπα του χρήστη και εντοπίζει ρούχα που χρήζουν προσοχής, βασιζόμενη σε χρονικά κατώφλια. Κάθε ρούχο κατατάσσεται σε μία από δύο κατηγορίες υπενθύμισης:

- **TRY_TODAY**: Ρούχο που δεν έχει φορεθεί ποτέ (μετά από 14 ημέρες) ή δεν έχει χρησιμοποιηθεί πρόσφατα (μετά από 30 ημέρες). Στόχος: να ενθαρρύνει τον χρήστη να αξιοποιήσει ρούχα που ξεχνά.
- **SHELVE**: Ρούχο αδρανές για 90+ ημέρες. Το σύστημα προτείνει αρχειοθέτηση (shelving) για να απλοποιηθεί η ντουλάπα.

```
private NudgeType classifyNudge(ClothingItem item, int daysUnworn) {
    if (item.getLastWornAt() == null) {
        if (daysUnworn >= properties.neverWornThresholdDays()) {
            return NudgeType.TRY_TODAY;
        }
        return null;
    }

    if (daysUnworn >= properties.shelveThresholdDays()) {
        return NudgeType.SHELVE;
    }

    if (daysUnworn >= properties.tryTodayThresholdDays()) {
        return NudgeType.TRY_TODAY;
    }

    return null;
}
```

Εικόνα 7.27. Κατηγοριοποίηση υπενθυμίσεων (classifyNudge)

Τα κατώφλια είναι παραμετροποιήσιμα μέσω `@ConfigurationProperties`, γεγονός που επιτρέπει αλλαγή τους χωρίς τροποποίηση κώδικα, μέσω αρχείων ρυθμίσεων ή μεταβλητών περιβάλλοντος.

Η ημερομηνία αναφοράς υπολογίζεται ως εξής: αν το ρούχο έχει φορεθεί τουλάχιστον μία φορά, χρησιμοποιείται η `lastWornAt`: αν δεν έχει φορεθεί ποτέ, χρησιμοποιείται η `createdAt` (η ημερομηνία που προστέθηκε στη ντουλάπα)

7.16 Διαγνωστικό Πλαίσιο Αδύνατων Προτάσεων (Suggestion Context)

Όταν η μηχανή δεν μπορεί να παράγει προτάσεις, ο `SuggestionContextBuilder` κατασκευάζει ένα πλαίσιο επεξήγησης που εμφανίζεται στον χρήστη. Αυτό περιλαμβάνει:

- **Πλήθος ενεργών ρούχων**: πόσα ρούχα έχει ο χρήστης στη ντουλάπα του
- **Πλήθος ρούχων ανά περίπτωση**: πόσα ρούχα φέρουν ετικέτα για τη ζητούμενη περίπτωση
- **Κατανομή κατηγοριών**: πόσα tops, bottoms, dresses κλπ.

- **Κατηγορίες που λείπουν:** τι χρειάζεται ο χρήστης για να σχηματίσει σύνολο
- **Χρωματικά clashes:** ποια ρούχα δημιουργούν ασυμβατότητα

```
static SuggestionContextResponse build(
    List<ClothingItem> wardrobe,
    List<ClothingItem> occasionItems,
    Occasion occasion,
    UserProfile profile,
    List<ColorHarmonyRules.ColorClash> clashes
) {
    List<ClothingItem> activeItems = wardrobe.stream()
        .filter(item -> item.getStatus() == ItemStatus.ACTIVE)
        .toList();

    Map<ClothingCategory, Long> counts = occasionItems.stream()
        .collect(Collectors.groupingBy(ClothingItem::getCategory, Collectors.counting()));

    List<CategoryCountResponse> categoryCounts = counts.entrySet().stream()
        .map(entry -> new CategoryCountResponse(entry.getKey(), entry.getValue().intValue()))
        .sorted((left, right) -> left.category().compareTo(right.category()))
        .toList();

    Set<ClothingCategory> owned = EnumSet.noneOf(ClothingCategory.class);
    owned.addAll(counts.keySet());

    List<ClothingCategory> missing = missingCategories(owned);

    List<ColorClashDetailResponse> clashDetails = clashes.stream()
        .map(clash -> new ColorClashDetailResponse(
            clash.itemOneName(),
            clash.itemOneColor(),
            clash.itemTwoName(),
            clash.itemTwoColor(),
            clash.description()
        ))
        .limit(5)
        .toList();

    return new SuggestionContextResponse(
        activeItems.size(),
        occasionItems.size(),
        categoryCounts,
        missing,
        clashDetails
    );
}
```

Εικόνα 7.28. Κατασκευή διαγνωστικού πλαισίου (SuggestionContextBuilder)

Αντί να εμφανίζει ένα γενικό μήνυμα «Δεν βρέθηκαν προτάσεις», το σύστημα εξηγεί γιατί δεν μπόρεσε και τι πρέπει να αλλάξει ο χρήστης.

7.17 Προτάσεις Αγορών (Purchase Suggestions)

Η υπηρεσία `PurchaseSuggestionService` λειτουργεί συμπληρωματικά: ενεργοποιείται όταν η μηχανή δεν μπορεί να σχηματίσει προτάσεις (λόγω ελλείψεων στη γκαρνταρόμπα ή χρωματικών clashes) και προτείνει τι να αγοράσει ο χρήστης. Ακολουθεί τρεις στρατηγικές:

Στρατηγική 1 — Βάσει Προφίλ (STYLE_MATCH)

Αν ο χρήστης έχει προφίλ προτιμήσεων, προτείνεται ρούχο στην αγαπημένη κατηγορία και χρώμα. Η υπηρεσία εξειδικεύει ακόμα και υποκατηγορία βάσει περίπτωσης: π.χ. αν ο χρήστης ζητά πρόταση για WORK και λείπει TOP, προτείνεται SHIRT (πουκάμισο). Αν ζητά για SPORT, προτείνεται T_SHIRT.

Στρατηγική 2 — Δομικά Κενά (WARDROBE_GAP)

Αν δεν υπάρχει φόρεμα ούτε πάνω+κάτω, προτείνονται τα βασικά κομμάτια που λείπουν. Η σειρά προτεραιότητας είναι: πάνω → κάτω → παπούτσια.

Στρατηγική 3 — Επανασήμανση (Retagging)

Αν δεν υπάρχουν ρούχα για τη ζητούμενη περίπτωση αλλά υπάρχουν ενεργά ρούχα χωρίς αυτή την ετικέτα, το σύστημα προτείνει στον χρήστη να τα ετικετάρει αντί να αγοράσει νέα. Στη διεπαφή, η πρόταση αυτή εμφανίζεται με την κατηγορία WARDROBE_GAP.

```
private ClothingSubcategory subcategoryFor(Occasion occasion, ClothingCategory category) {
    return switch (occasion) {
        case FORMAL, DATE_NIGHT -> switch (category) {
            case DRESS -> ClothingSubcategory.FORMAL_DRESS;
            case SHOES -> ClothingSubcategory.HEELS;
            case TOP -> ClothingSubcategory.BLOUSE;
            case BOTTOM -> ClothingSubcategory.TROUSERS;
            case OUTERWEAR -> ClothingSubcategory.BLAZER;
            default -> null;
        };
        case PARTY -> switch (category) {
            case DRESS -> ClothingSubcategory.FORMAL_DRESS;
            case SHOES -> ClothingSubcategory.HEELS;
            case TOP -> ClothingSubcategory.BLOUSE;
            default -> null;
        };
        case WORK -> switch (category) {
            case TOP -> ClothingSubcategory.SHIRT;
            case BOTTOM -> ClothingSubcategory.TROUSERS;
            case SHOES -> ClothingSubcategory.LOAFERS;
            case OUTERWEAR -> ClothingSubcategory.BLAZER;
            default -> null;
        };
        case SPORT -> switch (category) {
            case TOP -> ClothingSubcategory.T_SHIRT;
            case BOTTOM -> ClothingSubcategory.JOGGERS;
            case SHOES -> ClothingSubcategory.SNEAKERS;
            default -> null;
        };
        case TRAVEL -> switch (category) {
            case OUTERWEAR -> ClothingSubcategory.JACKET;
            case SHOES -> ClothingSubcategory.SNEAKERS;
            default -> null;
        };
        default -> switch (category) {
            case TOP -> ClothingSubcategory.T_SHIRT;
            case BOTTOM -> ClothingSubcategory.JEANS;
            case SHOES -> ClothingSubcategory.SNEAKERS;
            default -> null;
        };
    };
}
```

Εικόνα 7.29. Αντιστοίχιση περίπτωσης-υποκατηγορίας (subcategoryFor)

7.18 Διαχείριση Ειδικών Περιπτώσεων (Edge Case Handling)

Η μηχανή πρότασης χειρίζεται τρεις ειδικές καταστάσεις αποτυχίας, κάθε μία με εξειδικευμένο μήνυμα:

NO_OCCASION_ITEMS: Ο χρήστης δεν έχει κανένα ρούχο για τη ζητούμενη περίσταση. Το μήνυμα αναφέρει τον αριθμό ενεργών ρούχων και προτείνει να ετικετάρει υπάρχοντα ή να αγοράσει νέα.

INSUFFICIENT_ITEMS: Ο χρήστης έχει ρούχα για την περίσταση αλλά δεν σχηματίζεται πλήρες σύνολο (π.χ. μόνο tops, χωρίς bottoms). Το μήνυμα είναι στοχευμένο: «You have a work top but no matching bottom or dress yet.».

NO_VALID_COMBOS: Υπάρχουν αρκετά ρούχα αλλά όλοι οι συνδυασμοί αποκλείονται λόγω χρωματικών clashes. Το μήνυμα εμφανίζει τα συγκεκριμένα ζεύγη clash και προτείνει ουδέτερα εναλλακτικά.

Η μηχανή χειρίζεται επίσης τη λεγόμενη **ψυχρή εκκίνηση (cold start)**: όταν ο χρήστης δεν έχει αποθηκευμένα σύνολα ούτε ιστορικό χρήσης, δεν υπάρχει προφίλ προτιμήσεων. Σε αυτή την περίπτωση, η μηχανή χρησιμοποιεί μόνο τα αντικειμενικά κριτήρια (χρωματική αρμονία, πληρότητα) χωρίς προσωπικά μπόνους, αντί να αποτυγχάνει.

7.19 Εμφάνιση Αποτελεσμάτων στον Χρήστη — Κάρτα Πρότασης

Κάθε πρόταση εμφανίζεται μέσω του component `OutfitSuggestionCard`, το οποίο παρουσιάζει:

- **Εικόνες** κάθε ρούχου μέσω αυθεντικοποιημένου αιτήματος
- **Ετικέτα περίστασης** (π.χ. "Work match")
- **Βαθμολογία** (αριθμητική τιμή)
- **Σύνοψη**: περιγραφή χρωματικής παλέτας και ταιριάσματος στυλ
- **Χρωματική παλέτα**: οπτική αναπαράσταση χρωμάτων
- **Highlights**: κύρια χαρακτηριστικά (π.χ. "Includes shoes for a complete look", "Matches your style preferences")
- **Preference boost**: αν υπάρχει, αναφέρει πόσοι πόντοι προέρχονται από ιστορικό χρήσης
- **Κουμπί αποθήκευσης**: αποθηκεύει το σύνολο στη ντουλάπα

7.20 Εμφάνιση Αδύνατων Προτάσεων — Διαγνωστικό Περιβάλλον

Όταν δεν υπάρχουν προτάσεις, το component `SuggestionEmptyState` εμφανίζει πλήρες διαγνωστικό:

- **Στατιστικά**: ενεργά ρούχα, κομμάτια για τη ζητούμενη περίσταση, κατάσταση προφίλ στυλ
- **Κατανομή κατηγοριών**: τι έχει ο χρήστης (π.χ. "3 tops, 1 bottom")
- **Κατηγορίες που λείπουν**: τι χρειάζεται (π.χ. "Dress", "Shoes")
- **Χρωματικά clashes**: ποια ρούχα δημιουργούν ασυμβατότητα
- **Προτάσεις αγορών**: κάρτες με συγκεκριμένες συστάσεις, κατηγοριοποιημένες ως "Your style" (STYLE_MATCH) ή "Wardrobe gap" (WARDROBE_GAP)

Αυτή η προσέγγιση μετατρέπει μια αποτυχία σε εκπαιδευτική ευκαιρία: ο χρήστης μαθαίνει πώς λειτουργεί η ντουλάπα του.

7.21 Dashboard Αναλυτικών Στοιχείων

Η κεντρική σελίδα μετά τη σύνδεση (Dashboard, «Home» στη γραμμή πλοήγησης) εμφανίζει αναλυτικά στοιχεία για τη ντουλάπα. Τα στατιστικά υπολογίζονται client-side μέσω της συνάρτησης `buildWardrobeInsights`:

- **Ενεργά / αρχειοθετημένα ρούχα:** πόσα ρούχα σε κάθε κατάσταση
- **Αριθμός συνόλων:** πόσα σύνολα έχει δημιουργήσει ο χρήστης
- **Ρούχα χωρίς χρήση:** πόσα ενεργά ρούχα δεν έχουν φορεθεί ποτέ
- **Κατανομή ανά κατηγορία:** πόσα tops, bottoms, dresses κλπ.
- **Κατανομή ανά περίσταση:** πόσα ρούχα ανά περίσταση
- **Ακάλυπτες περιστάσεις:** περιστάσεις χωρίς κανένα ρούχο
- **Κυρίαρχα χρώματα:** τα 4 πιο συχνά χρώματα

```
export function buildWardrobeInsights(
  items: ClothingItemResponse[],
  outfits: OutfitResponse[]
): WardrobeInsights {
  const activeItems: ClothingItemResponse[] = items.filter((item: ClothingItemResponse) : boolean => item.status === 'ACTIVE')
  const shelvedItems: ClothingItemResponse[] = items.filter((item: ClothingItemResponse) : boolean => item.status === 'SHELVED')
  const neverWornCount: number = activeItems.filter((item: ClothingItemResponse) : boolean => item.wearCount === 0).length
  const totalItemWears: number = activeItems.reduce((sum: number, item: ClothingItemResponse) : number => sum + item.wearCount, 0)
  const totalOutfitWears: number = outfits.reduce((sum: number, outfit: OutfitResponse) : number => sum + outfit.wearCount, 0)

  const categoryCounts: CountEntry<ClothingCategory>[] = countBy(activeItems, (item: ClothingItemResponse) : ClothingCategory => item.category, ALL_CATEGORIES)
  const occasionCounts: CountEntry<Occasion>[] = countOccasions(activeItems)
  const uncoveredOccasions: Occasion[] = ALL_OCCASIONS.filter(
    (occasion: Occasion) : boolean => !occasionCounts.some((entry: CountEntry<Occasion>) : boolean => entry.key === occasion && entry.count > 0)
  )

  const topColors: CountEntry<ClothingColor>[] = countBy(activeItems, (item: ClothingItemResponse) : ClothingColor => item.primaryColor, [])
  .sort((left: CountEntry<ClothingColor>, right: CountEntry<ClothingColor>) : number => right.count - left.count)
  .slice(0, 4)

  const recentItems: ClothingItemResponse[] = [...activeItems]
  .sort((left: ClothingItemResponse, right: ClothingItemResponse) : number => Date.parse(right.createdAt) - Date.parse(left.createdAt))
  .slice(0, 4)

  const recentOutfits: OutfitResponse[] = [...outfits]
  .sort((left: OutfitResponse, right: OutfitResponse) : number => Date.parse(right.createdAt) - Date.parse(left.createdAt))
  .slice(0, 3)

  const tips: string[] = buildTips({
    activeItems,
    shelvedItems,
    outfits,
    categoryCounts,
    uncoveredOccasions,
    neverWornCount,
  })

  const intro: string = buildIntro({
    activeItemCount: activeItems.length,
    outfitCount: outfits.length,
  })
}
```

Εικόνα 7.30. Υπολογισμός αναλυτικών στοιχείων (`buildWardrobeInsights`)

7.22 Δυναμικές Συμβουλές και Προσωπικό Εισαγωγικό Μήνυμα

Η συνάρτηση `buildTips` παράγει έως 4 στοχευμένες συμβουλές βάσει της τρέχουσας κατάστασης της ντουλάπας:

1. **Πρώτο σύνολο:** αν δεν υπάρχει κανένα, προτρέπει τη δημιουργία
2. **Ρούχα χωρίς χρήση:** αν υπάρχουν, υπενθυμίζει τις υπενθυμίσεις
3. **Ελλιπείς κατηγορίες:** αν υπάρχουν tops αλλά όχι bottoms
4. **Ακάλυπτες περιστάσεις:** αν λείπουν ετικέτες περίστασης
5. **Αρχειοθετημένα ρούχα:** υπενθύμιση για αναθεώρηση

Η συνάρτηση `buildIntro` δημιουργεί εξατομικευμένο εισαγωγικό μήνυμα σε πέντε φάσεις (σε ελληνική απόδοση· στην εφαρμογή τα μηνύματα εμφανίζονται στα αγγλικά):

- **Κενή ντουλάπα:** «Η ψηφιακή ντουλάπα σας είναι έτοιμη. Προσθέστε το πρώτο σας κομμάτι...»
- **1-2 ρούχα:** «Ξεκινήσατε με X κομμάτι(α). Συνεχίστε...»
- **3+ ρούχα, 0 σύνολα:** «Έχετε X ρούχα σε Y περιστάσεις. Δημιουργήστε το πρώτο σύνολο...»
- **Ρούχα χωρίς χρήση:** «X κομμάτι(α) περιμένουν πρώτη χρήση.»
- **Πλήρης ντουλάπα:** «Η ντουλάπα σας σε καλή κατάσταση — X ενεργά ρούχα, Y σύνολα...»

7.23 Κατάλογος REST API Endpoints

Ο παρακάτω πίνακας παρουσιάζει τα REST API endpoints της εφαρμογής:

Πίνακας 7.4. Κατάλογος REST API endpoints

Μέθοδος	Διαδρομή	Περιγραφή
GET	/api/v1/info	Όνομα και έκδοση εφαρμογής
POST	/api/v1/auth/signup	Δημιουργία λογαριασμού
POST	/api/v1/auth/login	Σύνδεση χρήστη
GET	/api/v1/auth/user/{userId}	Στοιχεία προφίλ χρήστη
PATCH	/api/v1/auth/profile	Αλλαγή ονόματος/επωνύμου
PUT	/api/v1/auth/password	Αλλαγή κωδικού
DELETE	/api/v1/auth/account	Διαγραφή λογαριασμού
GET	/api/v1/wardrobe/items	Λίστα ρούχων
GET	/api/v1/wardrobe/items/{id}	Στοιχεία ρούχου
POST	/api/v1/wardrobe/items	Προσθήκη ρούχου
PUT	/api/v1/wardrobe/items/{id}	Επεξεργασία ρούχου
GET	/api/v1/wardrobe/items/{id}/image	Φωτογραφία ρούχου
DELETE	/api/v1/wardrobe/items/{id}	Διαγραφή ρούχου
POST	/api/v1/wardrobe/items/{id}/shelve	Αρχειοθέτηση ρούχου
GET	/api/v1/wardrobe/nudges	Υπενθυμίσεις αφορέτων ρούχων
GET	/api/v1/outfits	Λίστα εμφανίσεων
GET	/api/v1/outfits/{id}	Στοιχεία εμφάνισης
POST	/api/v1/outfits	Δημιουργία εμφάνισης
GET	/api/v1/outfits/suggestions?occasion=	Προτάσεις εμφάνισης
POST	/api/v1/outfits/{id}/wear	Καταγραφή φοράς
DELETE	/api/v1/outfits/{id}	Διαγραφή εμφάνισης
GET	/actuator/health	Έλεγχος υγείας συστήματος
GET	/actuator/info	Πληροφορίες εφαρμογής (Actuator)

7.24 Δομή Frontend (React + TypeScript)

Το frontend ακολουθεί δομή κατά λειτουργία (feature-based structure). Τα βασικά components οργανώνονται ως εξής:

- **components/wardrobe/**: Διαχείριση ρούχων (ClothingItemCard για την κάρτα ρούχου, WardrobeNudgeCard για τις υπενθυμίσεις, AuthenticatedImage για ασφαλή φόρτωση εικόνων)
- **components/outfit/**: Διαχείριση συνόλων (OutfitCard, OutfitSuggestionCard, SuggestionEmptyState)
- **pages/**: Οι σελίδες της εφαρμογής (HomePage, LoginPage, SignUpPage, AppDashboardPage, WardrobePage, AddClothingPage, OutfitsPage, CreateOutfitPage, SettingsPage)
- **utils/wardrobeInsights.ts**: Υπολογισμός στατιστικών client-side
- **types/**: TypeScript τύποι (wardrobe, outfit) που εξασφαλίζουν type safety
- **components/landing, layout, form, account, brand**: Αρχική σελίδα, πλαίσιο σελίδων και γραμμή πλοήγησης, πεδία φορμών, ρυθμίσεις λογαριασμού και λογότυπο
- **api/ και auth/**: Επικοινωνία με το REST API και διαχείριση σύνδεσης (JWT, hashCredential, ProtectedRoute)

Η χρήση TypeScript εξασφαλίζει ότι τα δεδομένα που ανταλλάσσονται μεταξύ backend (JSON) και frontend (React components) αντιστοιχούν πάντα στα αναμενόμενα πεδία. Τυχόν αναντιστοιχία εντοπίζεται κατά τη μεταγλώττιση και όχι κατά την εκτέλεση.

8. Σενάρια Χρήσης και Ανάδραση Συστήματος

Σκοπός αυτού του κεφαλαίου είναι να παρουσιάσει πώς το σύστημα Armoire ανταποκρίνεται διαφορετικά ανάλογα με το προφίλ, τις επιλογές και το ιστορικό χρήσης κάθε χρήστη. Για τον σκοπό αυτό, παρουσιάζονται δύο πλασματικοί χρήστες — η Μαρία και ο Γιάννης — με διαφορετικές ντουλάπες, προτιμήσεις και συνήθειες χρήσης. Κάθε σενάριο ακολουθεί μια πλήρη ροή αλληλεπίδρασης με την εφαρμογή, ώστε να αναδειχθεί ο τρόπος που η μηχανή προτάσεων, η εκμάθηση προτιμήσεων, οι υπενθυμίσεις και οι προτάσεις αγορών προσαρμόζονται στον κάθε χρήστη ξεχωριστά.

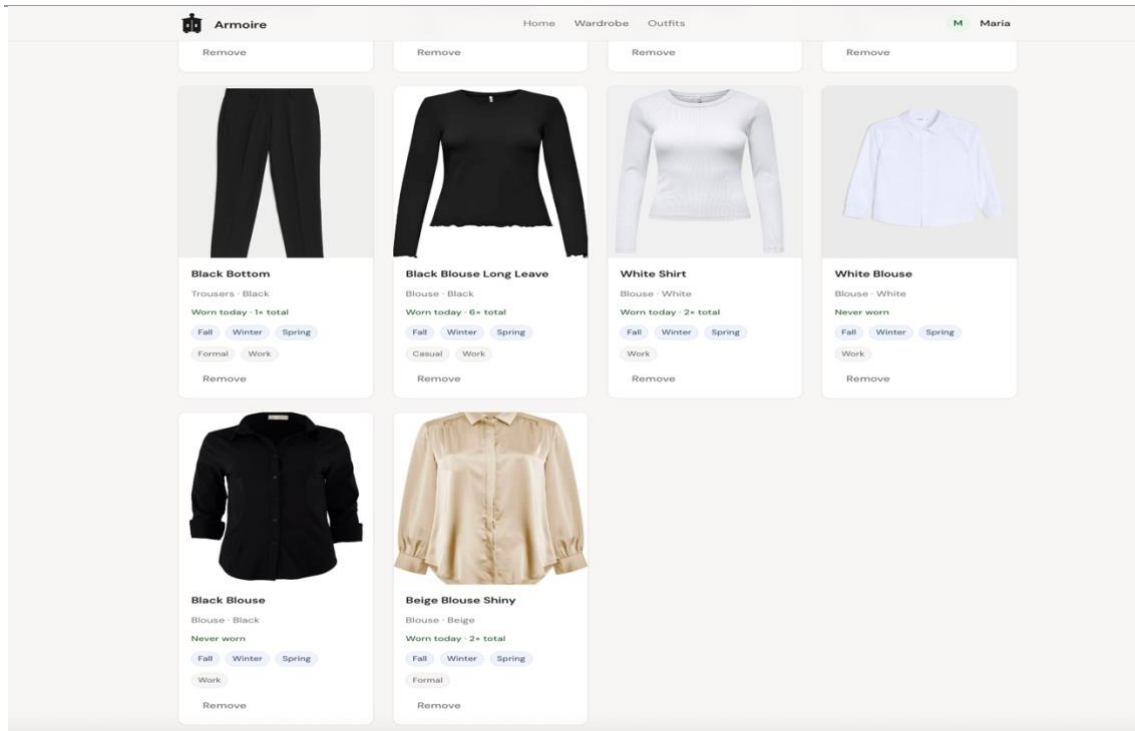
8.1 Σενάριο A: Η Μαρία — Εργαζόμενη με πλούσια ντουλάπα

8.1.1 Προφίλ και Ντουλάπα

Η Μαρία είναι εργαζόμενη σε γραφείο και χρησιμοποιεί το Armoire εδώ και αρκετές εβδομάδες. Έχει καταχωρίσει 18 ενεργά ρούχα στη γκαρνταρόμπα της:

- 5 tops (Black Blouse Long Sleeve, White Shirt, White Blouse, Black Blouse, Beige Blouse Shiny)
- 4 bottoms (Black Bottom, Blue Jean Dark Blue, Gray Long Skirt, Beige Trousers)
- 2 φορέματα (Black Long Dress, Dark Blue Dress)
- 3 πανωφόρια (Black Blazer, Black Coat, Beige Coat)
- 4 ζευγάρια παπούτσια (Black Loafers, Black Closed Heels, Brown Boots, White Sneakers)

Τα περισσότερα ρούχα της είναι σε ουδέτερα χρώματα (μαύρο, navy, μπεζ, λευκό, γκρι, καφέ), με εξαίρεση το Blue Jean Dark Blue, που είναι μπλε (μη ουδέτερο χρώμα). Έχει ετικετάρει 8 ρούχα για Work, 8 για Formal και 5 για Casual. Έχει καταγράψει 46 συνολικές χρήσεις, ενώ 5 ρούχα δεν έχουν φορεθεί ποτέ (White Blouse, Black Blouse, Beige Trousers, Dark Blue Dress, Brown Boots). Η ντουλάπα της, όπως φαίνεται στην εφαρμογή, εμφανίζεται στην Εικόνα 8.1.

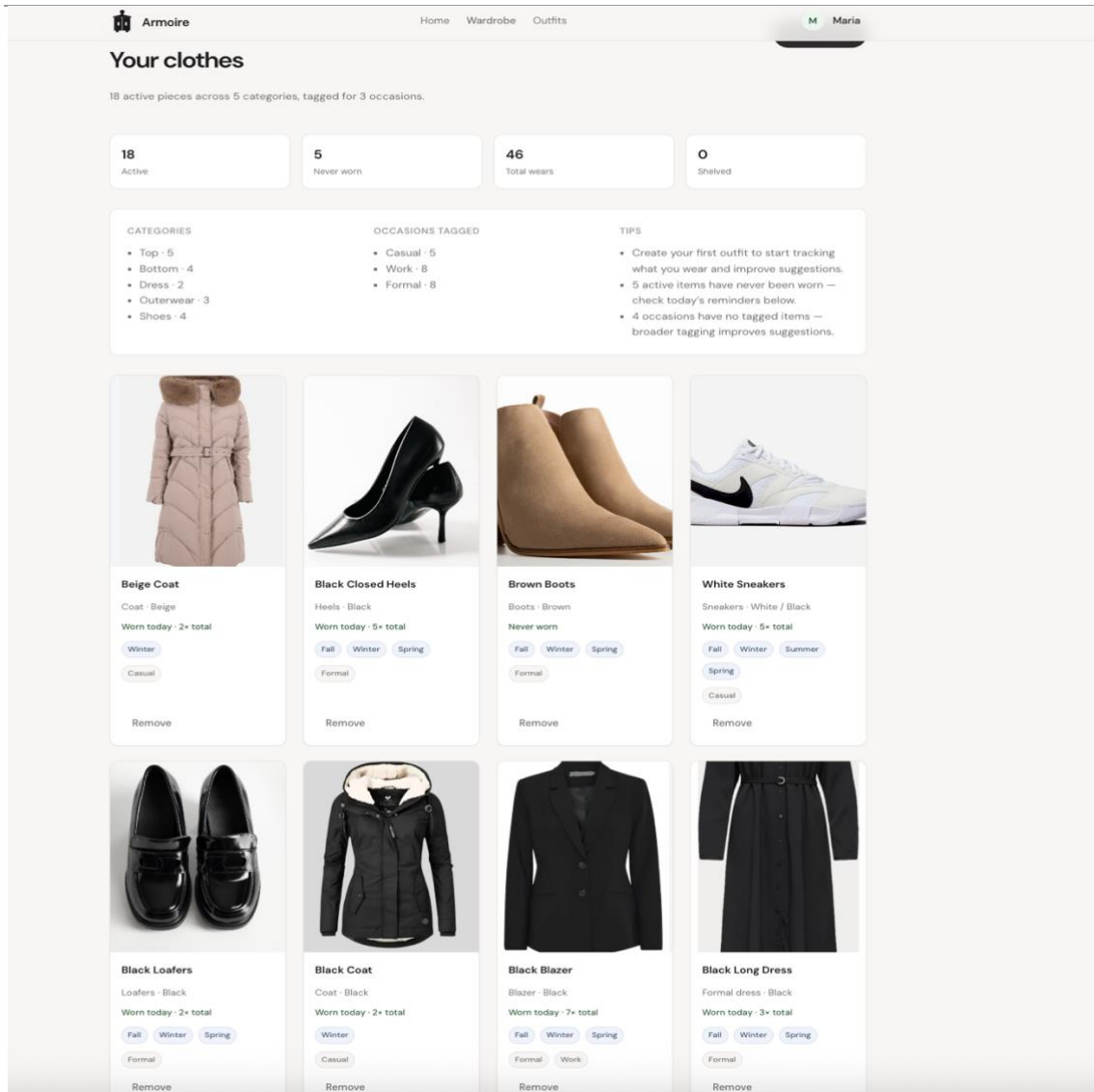


Εικόνα 8.1. Η γκαρνταρόμπα της Μαρίας με κάποια από τα 18 ρούχα

Παρατηρούμε ότι κάθε κάρτα εμφανίζει τη φωτογραφία του ρούχου, το όνομα, τον τύπο και χρώμα, τον αριθμό χρήσεων (π.χ. «Worn today · 7x total» για το Black Blazer), και τις ετικέτες εποχής και περιστάσης. Τα ρούχα που δεν έχουν φορεθεί ποτέ εμφανίζονται με ένδειξη «Never worn» σε πράσινο χρώμα. Η πλειοψηφία των ρούχων φέρει ετικέτα Work ή Formal, γεγονός που αντικατοπτρίζει τις καθημερινές ανάγκες της Μαρίας ως εργαζόμενη.

Στο πάνω μέρος της σελίδας (Εικόνα 8.2) εμφανίζονται τα στατιστικά: 18 ενεργά ρούχα, 5 ρούχα που δεν έχουν φορεθεί ποτέ, 46 συνολικές χρήσεις και 0 αρχειοθετημένα. Επίσης εμφανίζεται η κάλυψη κατηγοριών (Top: 5, Bottom: 4, Dress: 2, Outerwear: 3, Shoes: 4) και οι ετικεταρισμένες περιστάσεις (Casual: 5, Work: 8, Formal: 8), καθώς και δυναμικές συμβουλές:

- «Create your first outfit to start tracking what you wear and improve suggestions.»
- «5 active items have never been worn — check today's reminders below.»
- «4 occasions have no tagged items — broader tagging improves suggestions.»



Εικόνα 8.2. Στατιστικά, κάλυψη κατηγοριών και δυναμικές συμβουλές της Μαρίας

Τα δεδομένα του σεναρίου (π.χ. οι 46 καταγεγραμμένες χρήσεις) είχαν καταχωριστεί πριν από τη λήψη των στιγμιотύπων. Γι' αυτό η Εικόνα 8.2, που λήφθηκε πριν αποθηκευτούν εμφανίσεις, εμφανίζει τη συμβουλή «Create your first outfit», ενώ στο Dashboard της Μαρίας (Εικόνα 8.5) εμφανίζονται 9 αποθηκευμένες εμφανίσεις.

8.1.2 Αίτημα Πρότασης για Δουλειά (WORK)

Η Μαρία μεταβαίνει στη σελίδα «Plan your looks», επιλέγει την περίσταση «Work» από το αναπτυσσόμενο μενού και πατά «Suggest outfits». Η μηχανή εκτελεί το pipeline πέντε φάσεων που περιγράφηκε στην ενότητα 7.9:

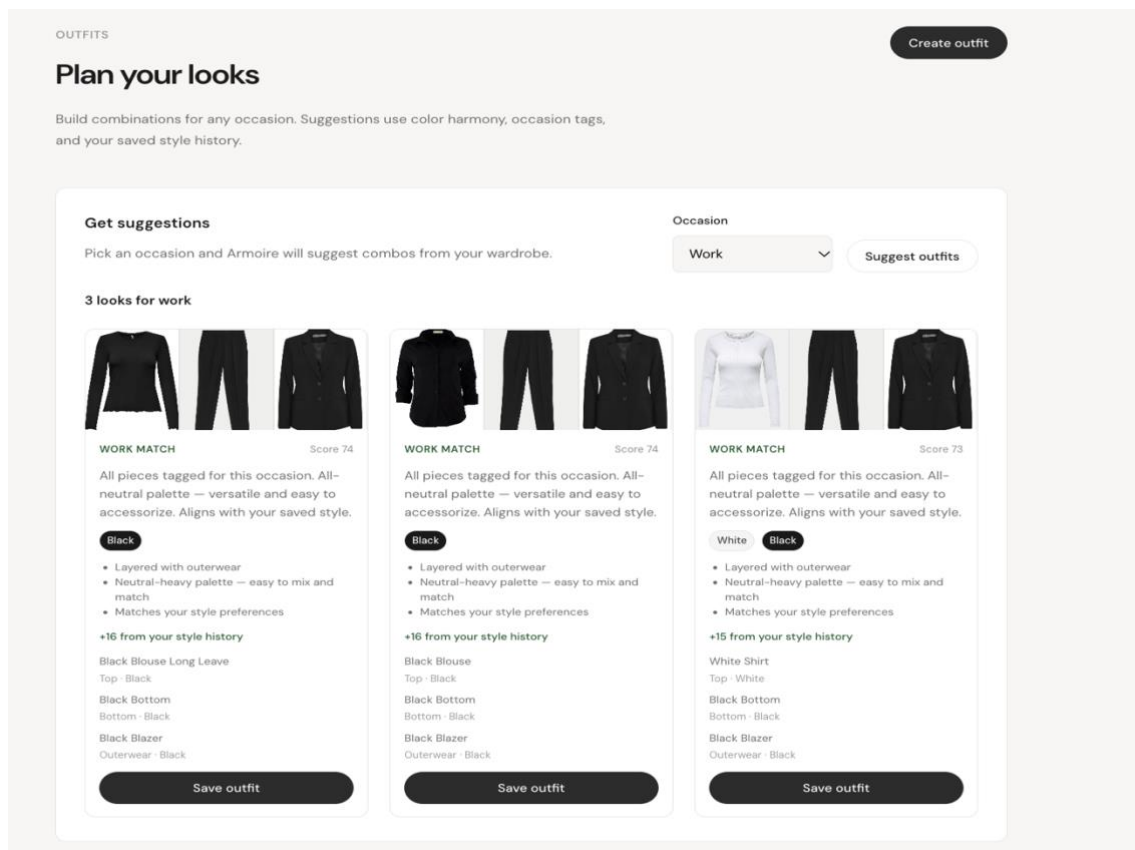
Φάση 1 — Φιλτράρισμα: Από τα 18 ρούχα, 8 φέρουν ετικέτα Work και επιλέγονται: Black Blouse Long Sleeve, White Shirt, White Blouse, Black Blouse, Black Bottom, Beige Trousers, Dark Blue Dress και Black Blazer.

Φάση 2 — Δημιουργία βάσεων: Δημιουργούνται βάσεις συνδυασμών: $4 \text{ tops} \times 2 \text{ bottoms} = 8 \text{ βάσεις top+bottom}$, συν 1 φόρεμα (Dark Blue Dress) = 9 βάσεις.

Φάση 3 — Επέκταση: Κάθε βάση επεκτείνεται με το Black Blazer (μοναδικό πανωφόρι με ετικέτα Work). Δεν υπάρχουν παπούτσια με ετικέτα Work, οπότε δεν προστίθενται υποδήματα. Παράγονται 18 παραλλαγές συνολικά.

Φάση 4 — Χρωματικός έλεγχος: Όλοι οι συνδυασμοί περνούν τον έλεγχο, διότι τα χρώματα (μαύρο, λευκό, μπεζ, navy) είναι ουδέτερα και δεν δημιουργούν χρωματικές ασυμβατότητες (clashes) σύμφωνα με τους κανόνες της ενότητας 7.10.

Φάση 5 — Βαθμολόγηση: Επειδή η Μαρία έχει πλούσιο ιστορικό χρήσης (46 χρήσεις), η μηχανή κατασκευάζει προφίλ προτιμήσεων. Τα μαύρα ρούχα κυριαρχούν στο ιστορικό της (Black Blazer: 7×, Black Blouse Long Leave: 6×, Blue Jean: 6×, Black Closed Heels: 5×, White Sneakers: 5×). Οι συνδυασμοί που μοιάζουν στα αγαπημένα της κερδίζουν υψηλό cosine similarity και Jaccard similarity. Τα αποτελέσματα εμφανίζονται ως κάρτες στην οθόνη. Η μηχανή επιστρέφει 3 προτάσεις (Εικόνα 8.3):



Εικόνα 8.3. Οι τρεις προτάσεις WORK για τη Μαρία

Πίνακας 8.1. Ανάλυση βαθμολογίας προτάσεων WORK για τη Μαρία

Πρόταση	Top	Bottom	Outerwear	Score	Style History
1η	Black Blouse Long Leave	Black Bottom	Black Blazer	74	+16

Πρόταση	Top	Bottom	Outerwear	Score	Style History
2η	Black Blouse	Black Bottom	Black Blazer	74	+16
3η	White Shirt	Black Bottom	Black Blazer	73	+15

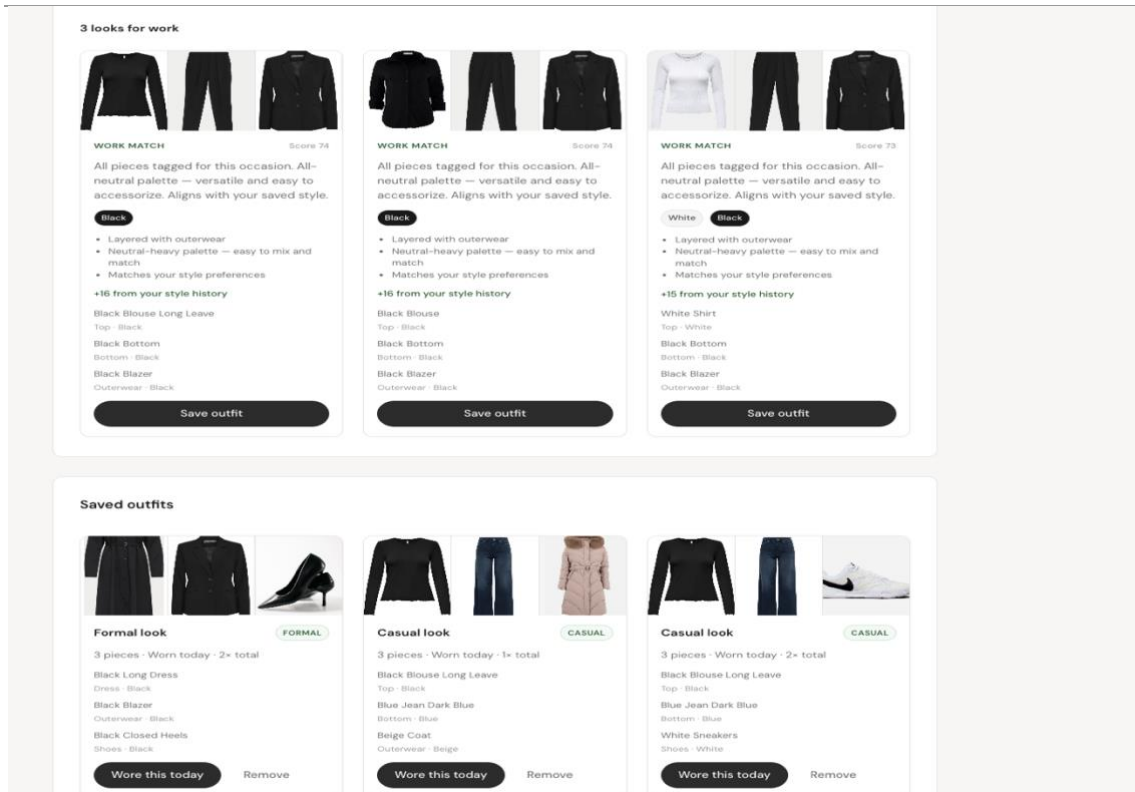
Παρατηρούμε ότι:

- Το Black Bottom και το Black Blazer εμφανίζονται σε όλες τις προτάσεις. Το Black Blazer είναι το μοναδικό πανωφόρι με ετικέτα Work, ενώ το Black Bottom προτιμάται έναντι του Beige Trousers (που επίσης φέρει ετικέτα Work) λόγω του μεγαλύτερου style history boost, αφού το Beige Trousers δεν έχει φορεθεί ποτέ.
- Η διαφοροποίηση γίνεται στο top: οι δύο πρώτες χρησιμοποιούν μαύρα tops (all-black παλέτα), ενώ η τρίτη εισάγει το White Shirt για χρωματική αντίθεση (White + Black).
- Κάθε κάρτα εμφανίζει ετικέτα «WORK MATCH», χρωματικά chips (π.χ. «Black» ή «White», «Black»), και highlights: «Layered with outerwear», «Neutral-heavy palette — easy to mix and match», «Matches your style preferences».
- Η βαθμολογία style history (+16 και +15) οφείλεται στο ότι η Μαρία έχει φορέσει πολλές φορές τα μαύρα ρούχα, οπότε οι all-black συνδυασμοί ευθυγραμμίζονται με το αποθηκευμένο προφίλ στυλ της.

Σημαντικό: Η ένδειξη «Matches your style preferences» εμφανίζεται μόνο όταν ο χρήστης έχει προφίλ προτιμήσεων και η ομοιότητα συνημιτόνου της εμφάνισης με το προφίλ είναι τουλάχιστον 0.35 — σε αντίθεση με τον Γιάννη (βλ. ενότητα 8.2) που δεν θα δει αυτό το μήνυμα.

8.1.3 Αποθήκευση και Καταγραφή Χρήσης

Η Μαρία αποφασίζει να φορέσει την 1η πρόταση. Πατά «Save outfit» για να την αποθηκεύσει στα σύνολά της, και στη συνέχεια, από τη σελίδα Outfits, μπορεί να πατήσει «Wore this today» για να καταγράψει ότι τη φόρεσε σήμερα.



Εικόνα 8.4. Κουμπί «Save outfit» στην κάρτα πρότασης

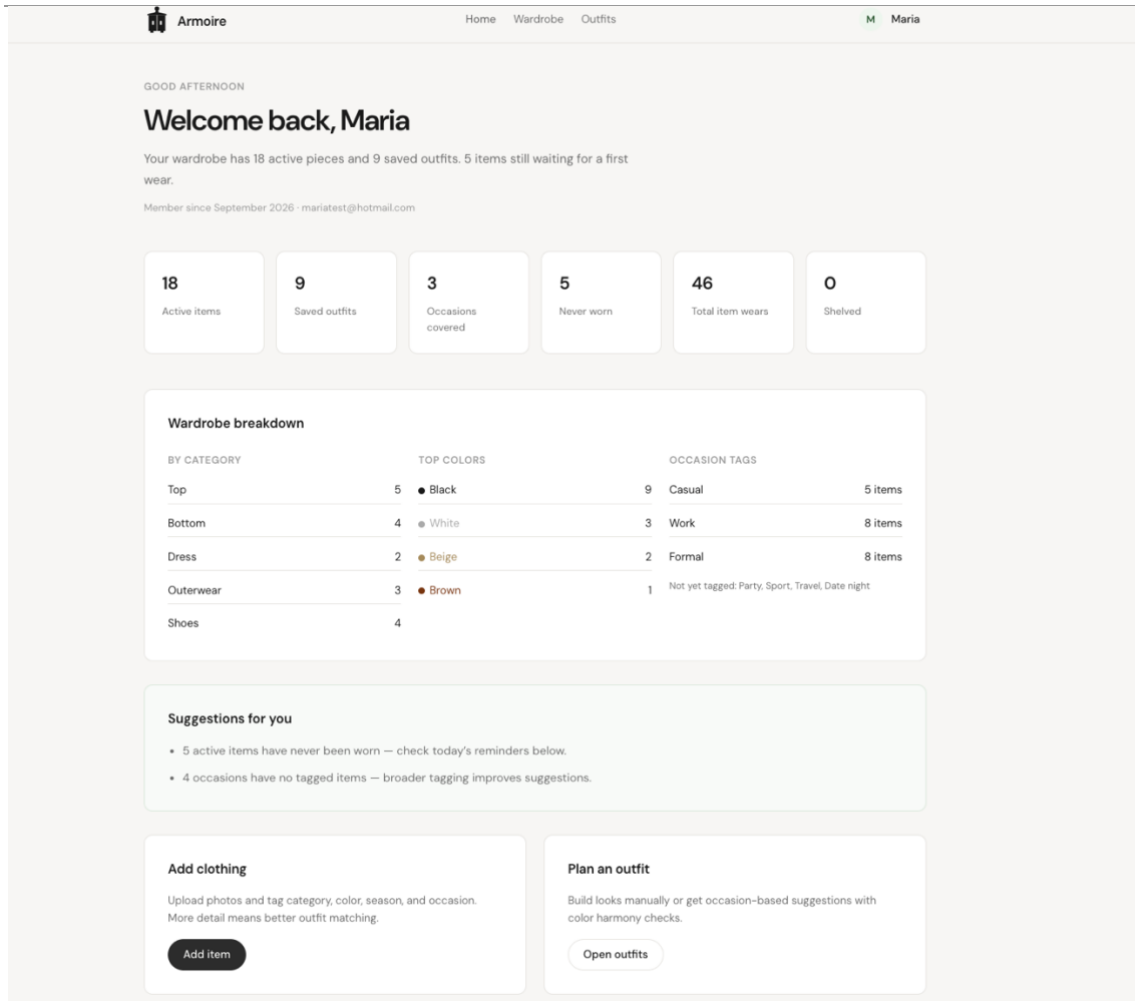
Η καταγραφή ενημερώνει αμέσως:

- Τον μετρητή χρήσεων (wear count) κάθε ρούχου στο σύνολο. Το Black Blouse Long Leave πηγαίνει από 6 σε 7 χρήσεις.
- Την ημερομηνία τελευταίας χρήσης σε σήμερα.
- Το προφίλ προτιμήσεων: η επόμενη φορά που η Μαρία ζητήσει πρόταση, ο συνδυασμός μαύρο top + μαύρο bottom + Black Blazer θα έχει ακόμα μεγαλύτερη βαρύτητα.

8.1.4 Dashboard της Μαρίας

Η Μαρία μεταβαίνει στο Dashboard. Το εισαγωγικό μήνυμα είναι δυναμικό και προσαρμοσμένο στην κατάσταση της:

«Your wardrobe has 18 active pieces and 9 saved outfits. 5 items still waiting for a first wear.»



Εικόνα 8.5. Dashboard της Μαρίας: εισαγωγικό μήνυμα και στατιστικά

Στα στατιστικά εμφανίζονται: 18 ενεργά ρούχα, 5 ρούχα χωρίς χρήση, 46 συνολικές χρήσεις και 0 αρχειοθετημένα. Οι δυναμικές συμβουλές περιλαμβάνουν:

- «5 active items have never been worn — check today's reminders below.» (αναφέρεται στα: White Blouse, Black Blouse, Beige Trousers, Dark Blue Dress, Brown Boots)
- «4 occasions have no tagged items — broader tagging improves suggestions.» (δεν υπάρχουν ρούχα ετικεταρισμένα για Party, Sport, Date Night, Travel)

8.1.5 Υπενθυμίσεις Ντουλάπας

Στο κάτω μέρος του Dashboard, εμφανίζεται η ενότητα «Wardrobe reminders» με κάρτες υπενθυμίσεων. Το σύστημα αξιολογεί κάθε ρούχο και δημιουργεί δύο τύπους υπενθυμίσεων:

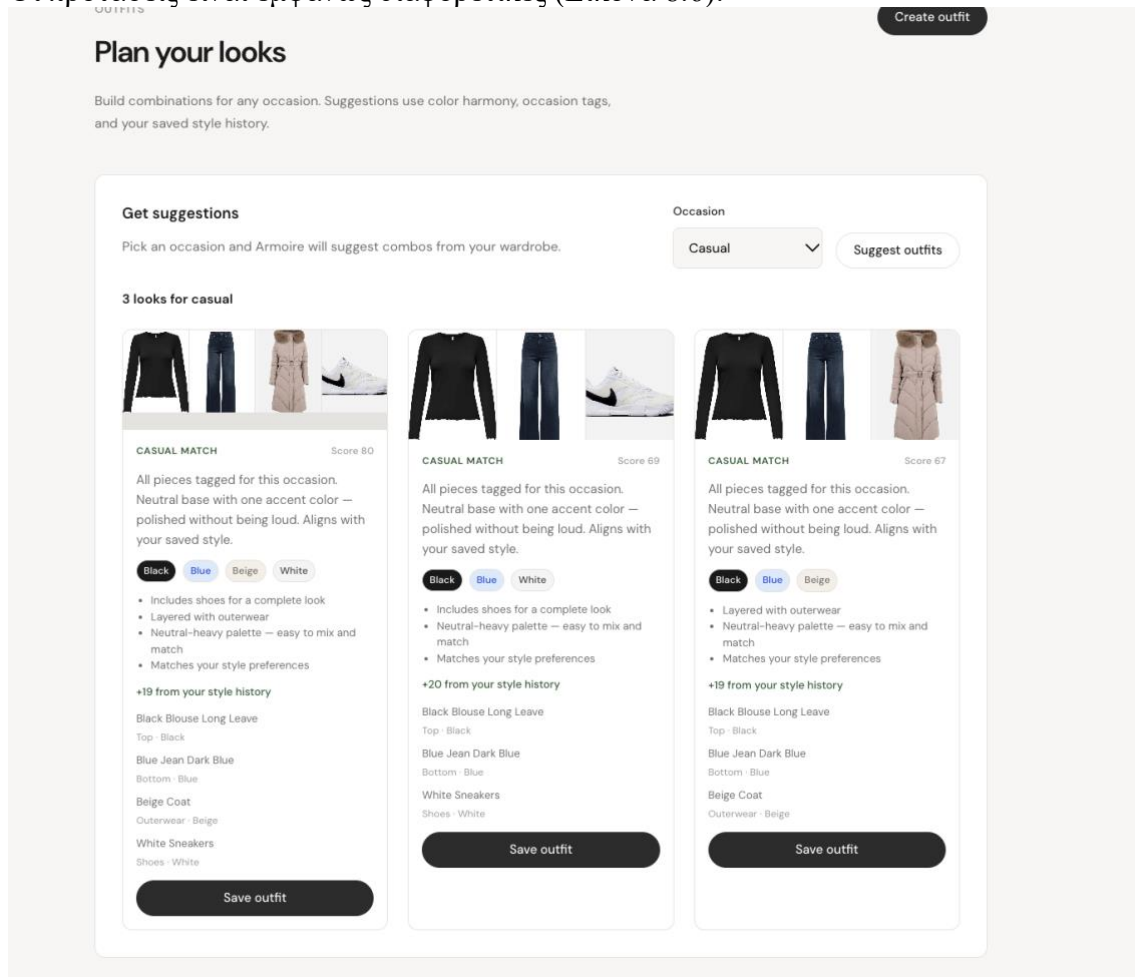
- **TRY_TODAY** — εμφανίζεται όταν ένα ρούχο δεν έχει φορεθεί ποτέ και έχουν περάσει τουλάχιστον 14 ημέρες από την προσθήκη του, ή όταν έχει φορεθεί αλλά όχι τις τελευταίες 30 ημέρες. Κάθε κάρτα δείχνει badge «Try today», το όνομα του ρούχου, τις ημέρες αδράνειας και ένα μήνυμα ενθάρρυνσης.
- **SHELVE** — εμφανίζεται όταν ένα ρούχο έχει φορεθεί στο παρελθόν αλλά όχι τις τελευταίες 90+ ημέρες, με κουμπί «Shelve item» για αρχειοθέτηση.

Η Μαρία έχει 5 ρούχα που δεν έχουν φορεθεί ποτέ (White Blouse, Black Blouse, Beige Trousers, Dark Blue Dress, Brown Boots). Καθώς αυτά τα ρούχα συμπληρώνουν τον απαιτούμενο χρόνο αδράνειας, θα εμφανιστούν ως κάρτες TRY TODAY. Η εφαρμογή εμφανίζει μέχρι 3 υπενθυμίσεις τη φορά, ώστε να μην κατακλύζει τον χρήστη.

Η σχετική δυναμική συμβουλή στα tips λέει ήδη: «5 active items have never been worn — check today's reminders below», παραπέμποντας τον χρήστη στις κάρτες υπενθυμίσεων.

8.1.6 Αίτημα Πρότασης για Casual — Διαφοροποίηση

Η Μαρία δοκιμάζει τώρα μια πρόταση για CASUAL αντί για WORK. Η μηχανή φιλτράρει τα ρούχα εκ νέου — αυτή τη φορά 5 ρούχα φέρουν ετικέτα Casual: Black Blouse Long Leave, Blue Jean Dark Blue, Black Coat, Beige Coat και White Sneakers. Οι προτάσεις είναι εμφανώς διαφορετικές (Εικόνα 8.6):



Εικόνα 8.6. Οι τρεις προτάσεις CASUAL για τη Μαρία

Πίνακας 8.2. Ανάλυση βαθμολογίας προτάσεων CASUAL για τη Μαρία

Πρόταση	Top	Bottom	Outerwear	Shoes	Score	Style History
1η	Black Blouse Long Leave	Blue Jean Dark Blue	Beige Coat	White Sneakers	80	+19
2η	Black Blouse Long Leave	Blue Jean Dark Blue	-	White Sneakers	69	+20
3η	Black Blouse Long Leave	Blue Jean Dark Blue	Beige Coat	-	67	+19

Σε σύγκριση με τις προτάσεις WORK (Πίνακας 8.1), παρατηρούμε σημαντικές διαφορές:

- **Αλλαγή ρούχων:** Το Blue Jean Dark Blue αντικαθιστά το Black Bottom, προστίθενται τα White Sneakers (στις προτάσεις WORK δεν υπήρχαν παπούτσια με ετικέτα Work), και το Beige Coat αντικαθιστά το Black Blazer. Ο χαρακτήρας του συνόλου αλλάζει ριζικά.
- **Χρωματική παλέτα:** Αντί για την all-black παλέτα του WORK, εμφανίζονται 4 χρώματα (Black, Blue, Beige, White) με ετικέτα «Neutral base with one accent color — polished without being loud».
- **Πληρότητα συνόλου:** Η 1η πρόταση (score 80) αποτελείται από 4 κομμάτια (top + bottom + outerwear + shoes), ενώ η 2η παραλείπει το outerwear και η 3η τα παπούτσια. Αυτό εξηγεί τη διαφορά βαθμολογίας: τα πιο πλήρη σύνολα κερδίζουν bonus πληρότητας.
- **Style history υψηλότερο:** Οι βαθμολογίες style history (+19, +20) είναι υψηλότερες από τις αντίστοιχες WORK (+15, +16), γιατί τα casual ρούχα (Blue Jean: 6×, White Sneakers: 5×, Black Blouse Long Leave: 6×) φοριούνται συχνότερα από τα work bottoms (Black Bottom: 1×).
- **Highlights:** Κάθε κάρτα εμφανίζει «Includes shoes for a complete look» (όταν περιλαμβάνει παπούτσια), «Layered with outerwear» (όταν περιλαμβάνει πανωφόρι), «Neutral-heavy palette» και «Matches your style preferences».

Αυτή η σύγκριση αναδεικνύει τον τρόπο που η ίδια μηχανή προτάσεων παράγει εντελώς διαφορετικά σύνολα ανάλογα με την περίσταση, χρησιμοποιώντας τα ίδια ρούχα και το ίδιο ιστορικό χρήσης.

8.2 Σενάριο Β: Ο Γιάννης — Νέος χρήστης με μικρή ντουλάπα

8.2.1 Προφίλ και Ντουλάπα

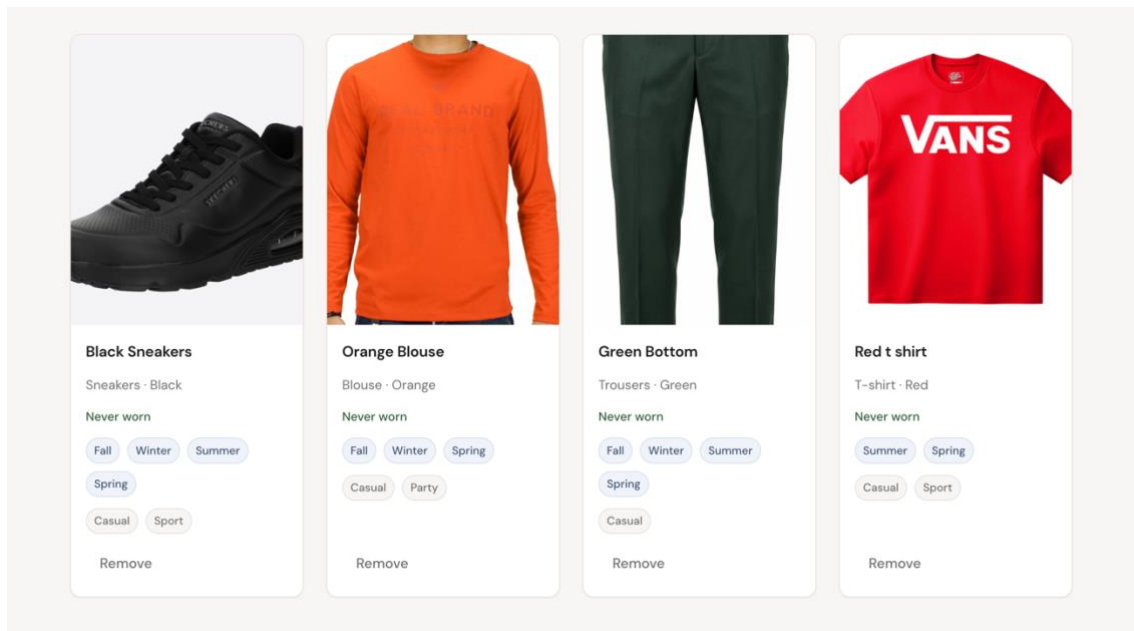
Ο Γιάννης είναι φοιτητής που μόλις εγγράφηκε στο Armoire. Έχει καταχωρίσει μόνο 4 ρούχα στη γκαρνταρόμπα του:

- 2 tops (Orange Blouse, Red T-shirt)
- 1 bottom (Green Bottom)
- 1 ζευγάρι παπούτσια (Black Sneakers)

Σε αντίθεση με τη Μαρία, τα χρώματά του είναι ζωντάρια και έντονα (πορτοκαλί, πράσινο, κόκκινο, μαύρο). Έχει ετικετάρει 4 ρούχα για Casual, 2 για Sport (Black Sneakers, Red T-shirt) και 1 για Party (Orange Blouse). Δεν έχει καταγράψει καμία

χρήση — και τα 4 ρούχα εμφανίζονται ως «Never worn». Δεν υπάρχουν πανωφόρια, φορέματα, ούτε ρούχα ετικεταρισμένα για Work ή Formal.

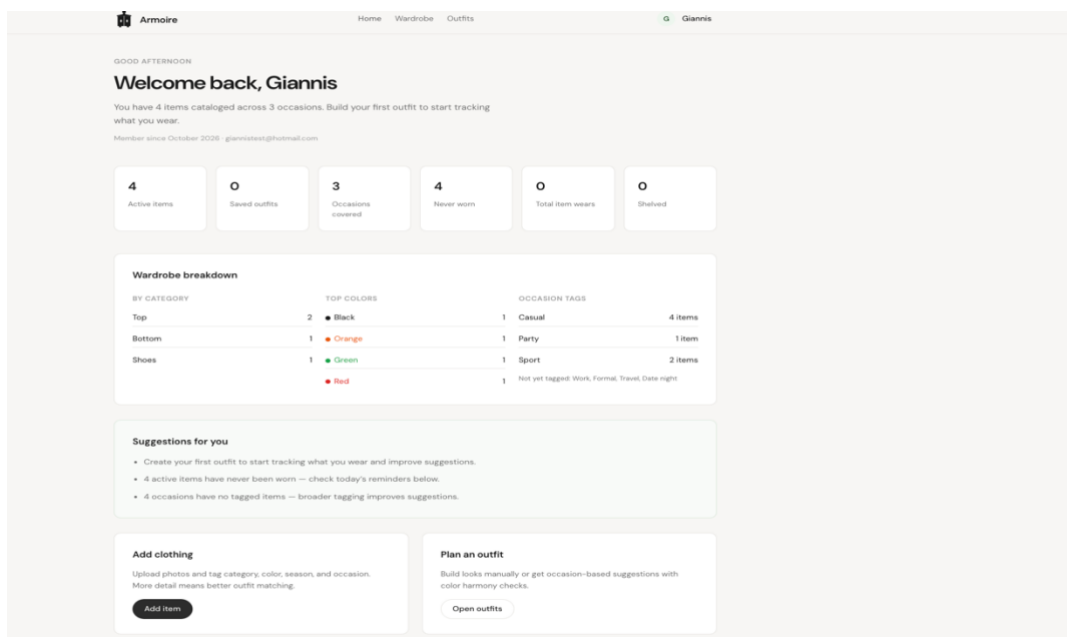
Η ντουλάπα του, όπως φαίνεται στην εφαρμογή, εμφανίζεται στην Εικόνα 8.7.



Εικόνα 8.7. Η γκαρνταρόμπα του Γιάννη (4 ρούχα)

Στο πάνω μέρος εμφανίζονται τα στατιστικά: 4 ενεργά, 4 never worn, 0 χρήσεις, 0 αρχειοθετημένα. Η κάλυψη κατηγοριών (Top: 2, Bottom: 1, Shoes: 1) και οι περιστάσεις (Casual: 4, Party: 1, Sport: 2) δείχνουν αμέσως τις ελλείψεις. Οι δυναμικές συμβουλές αναφέρουν:

- «Create your first outfit to start tracking what you wear and improve suggestions.»
- «4 active items have never been worn — check today's reminders below.»
- «4 occasions have no tagged items — broader tagging improves suggestions.»



Εικόνα 8.8. Στατιστικά, κάλυψη κατηγοριών και συμβουλές του Γιάννη

8.2.2 Αίτημα Πρότασης για Casual (CASUAL)

Ο Γιάννης μεταβαίνει στη σελίδα «Plan your looks» και επιλέγει Casual. Η μηχανή εκτελεί το pipeline πέντε φάσεων:

Φάση 1 — Φιλτράρισμα: Και τα 4 ρούχα φέρουν ετικέτα Casual και επιλέγονται.

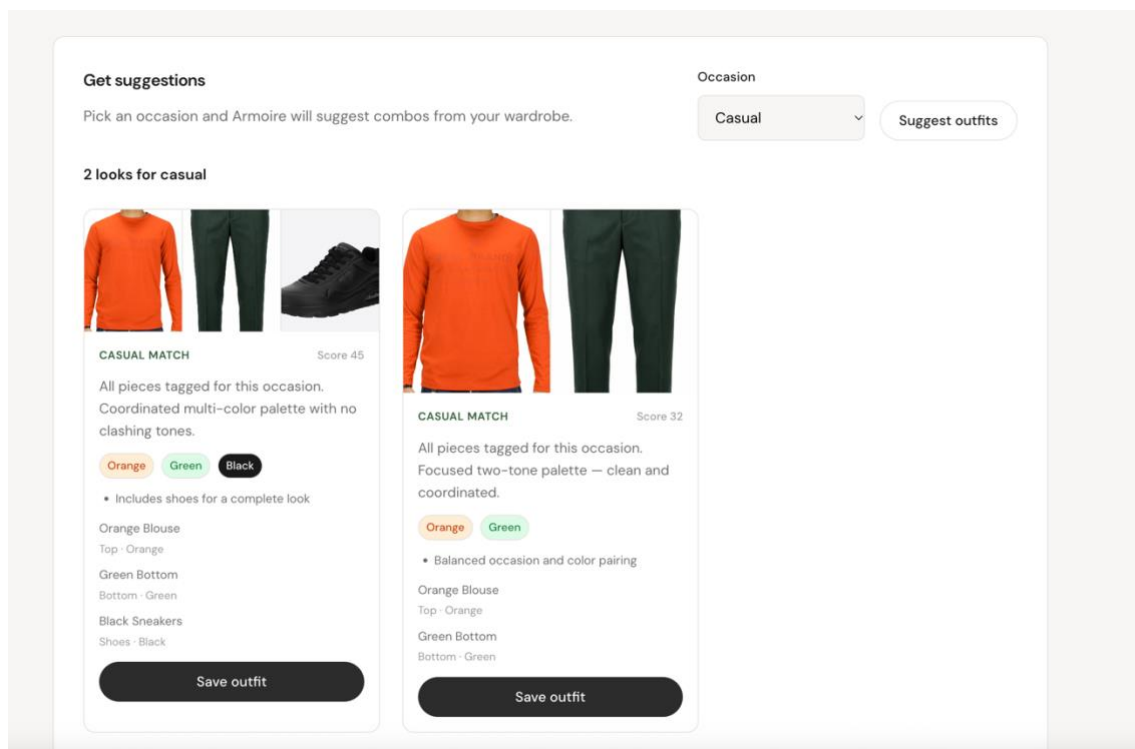
Φάση 2 — Δημιουργία βάσεων: 2 tops × 1 bottom = 2 βάσεις (Orange Blouse + Green Bottom, Red T-shirt + Green Bottom).

Φάση 3 — Επέκταση: Δεν υπάρχει πανωφόρι με ετικέτα Casual. Τα Black Sneakers (Casual) προστίθενται ως παπούτσια. Παράγονται 4 παραλλαγές (2 βάσεις × με/χωρίς παπούτσια).

Φάση 4 — Χρωματικός έλεγχος: Εδώ εντοπίζεται η κρίσιμη διαφορά με τη Μαρία. Ο συνδυασμός Red T-shirt + Green Bottom (κόκκινο + πράσινο) αποτελεί χρωματική ασυμβατότητα (clash) σύμφωνα με τους κανόνες της ενότητας 7.10, και απορρίπτεται. Ο συνδυασμός Orange Blouse + Green Bottom περνά τον έλεγχο (δεν ανήκουν στο ίδιο clash set), με χαρακτηρισμό «Coordinated multi-color palette with no clashing tones».

Φάση 5 — Βαθμολόγηση: Επειδή ο Γιάννης δεν έχει ιστορικό χρήσης (0 χρήσεις, 0 αποθηκευμένα σύνολα), το style history boost είναι 0 για όλες τις προτάσεις. Η βαθμολογία βασίζεται αποκλειστικά στην πληρότητα του συνόλου και τη χρωματική αρμονία.

Η μηχανή επιστρέφει μόνο 2 προτάσεις (Εικόνα 8.9) — σε αντίθεση με τις 3 της Μαρίας:



Εικόνα 8.9. Οι δύο προτάσεις CASUAL για τον Γιάννη

Πίνακας 8.3. Ανάλυση βαθμολογίας προτάσεων CASUAL για τον Γιάννη

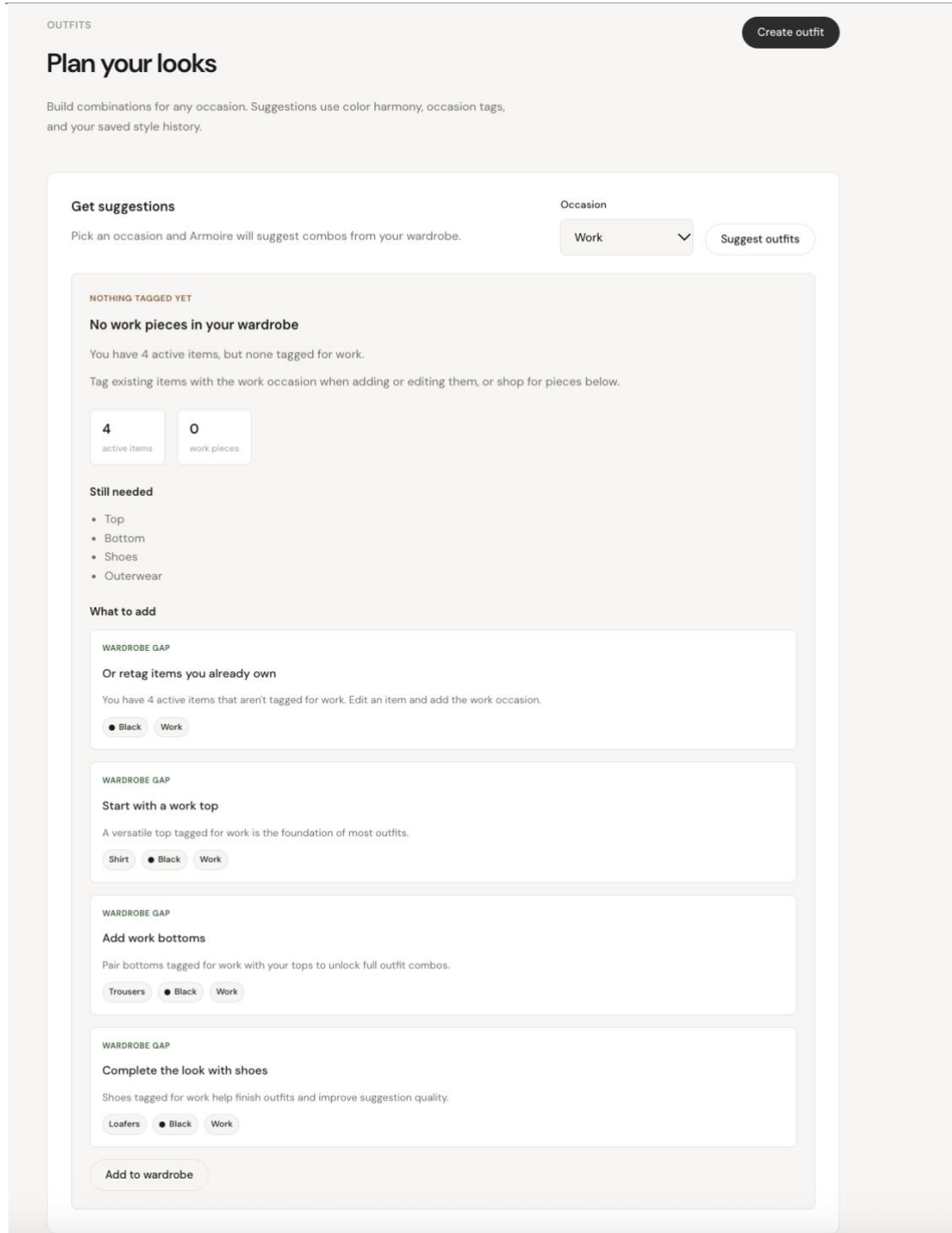
Πρόταση	Top	Bottom	Outerwear	Shoes	Score	Style History
1η	Orange Blouse	Green Bottom	-	Black Sneakers	45	0
2η	Orange Blouse	Green Bottom	-	-	32	0

Παρατηρούμε ότι:

- Και οι δύο προτάσεις χρησιμοποιούν το ίδιο top (Orange Blouse) γιατί ο συνδυασμός Red + Green απορρίφθηκε ως clash. Αυτό δείχνει τη λειτουργία του χρωματικού φίλτρου σε πραγματικές συνθήκες.
- Η 1η πρόταση (score 45) περιλαμβάνει τα Black Sneakers και λαμβάνει highlight «Includes shoes for a complete look», ενώ η 2η (score 32) αποτελείται μόνο από top + bottom.
- Κάθε κάρτα εμφανίζει χρωματικά chips (Orange, Green, Black) και χαρακτηρισμό «Coordinated multi-color palette with no clashing tones» ή «Focused two-tone palette — clean and coordinated».
- Δεν εμφανίζεται η ένδειξη «Matches your style preferences» σε καμία κάρτα, γιατί ο Γιάννης δεν έχει ακόμη προφίλ προτιμήσεων (καμία αποθηκευμένη εμφάνιση ή καταγραφή χρήσης). Αυτή η απουσία επιβεβαιώνει ότι η μηχανή διαφοροποιεί τα highlights ανάλογα με το προφίλ του χρήστη.
- Οι βαθμολογίες (45, 32) είναι σημαντικά χαμηλότερες από αυτές της Μαρίας (80, 69, 67) λόγω: (α) λιγότερων κομματιών ανά σύνολο (χωρίς πανωφόρι), (β) μηδενικού style history boost, και (γ) μικρότερου εύρους ντουλάπας.

8.2.3 Αίτημα Πρότασης για Work — Χωρίς Ρούχα για Work

Ο Γιάννης δοκιμάζει τώρα να ζητήσει πρόταση για Work. Κανένα από τα 4 ρούχα του δεν φέρει ετικέτα Work, οπότε η μηχανή δεν μπορεί να παράξει προτάσεις. Αντί για κάρτες συνόλων, εμφανίζεται η ενότητα «No work pieces in your wardrobe» (Εικόνα 8.10).



Εικόνα 8.10. Μήνυμα «No work pieces in your wardrobe» και προτάσεις αγορών

Η εφαρμογή εμφανίζει:

- Ετικέτα «NOTHING TAGGED YET» σε πορτοκαλί χρώμα
- Μήνυμα: «You have 4 active items, but none tagged for work.»
- Στατιστικά: 4 active items, 0 work pieces

- Λίστα «Still needed»: Top, Bottom, Shoes, Outerwear — δηλαδή λείπουν ρούχα σε όλες τις κατηγορίες για την περίπτωση Work

Ακολουθούν 4 κάρτες προτάσεων αγοράς (WARDROBE GAP):

1. «**Or retag items you already own**» — Προτρέπει τον χρήστη να ετικετάρει υπάρχοντα ρούχα για Work. Εμφανίζει chips: Black, Work.
2. «**Start with a work top**» — Προτείνει αγορά πουκάμισου. Chips: Shirt, Black, Work.
3. «**Add work bottoms**» — Προτείνει αγορά παντελονιού. Chips: Trousers, Black, Work.
4. «**Complete the look with shoes**» — Προτείνει αγορά παπουτσιών. Chips: Loafers, Black, Work.

Παρατηρούμε ότι όλες οι προτάσεις αγοράς προτείνουν μαύρο χρώμα — η μηχανή επιλέγει ουδέτερα χρώματα για Work ώστε να μεγιστοποιήσει τη συνδυαστικότητα. Κάθε κάρτα είναι τύπου WARDROBE_GAP, που σημαίνει ότι η πρόταση βασίζεται σε ανάλυση ελλείψεων της ντουλάπας (ενότητα 7.17).

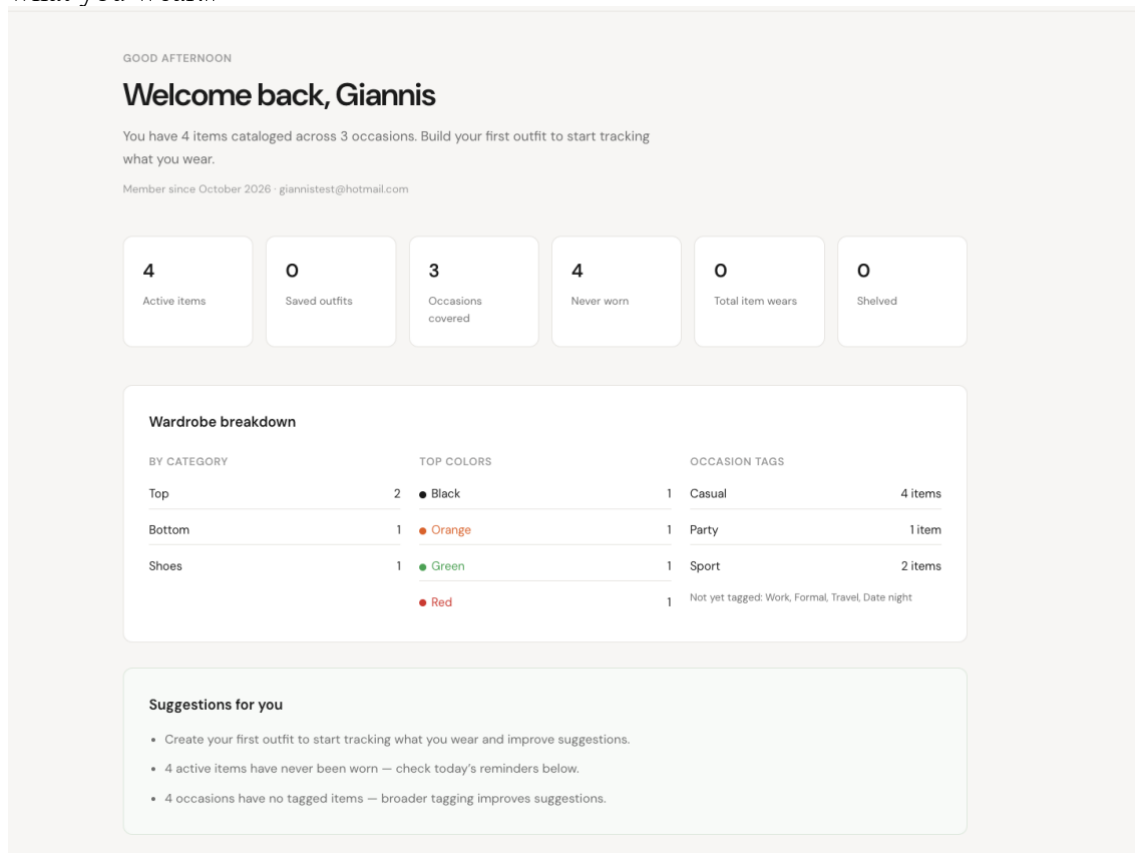
Στο κάτω μέρος εμφανίζεται κουμπί «Add to wardrobe» που οδηγεί τον χρήστη απευθείας στη σελίδα προσθήκης ρούχου.

8.2.4 Dashboard του Γιάννη

Ο Γιάννης μεταβαίνει στο Dashboard. Το εισαγωγικό μήνυμα διαφέρει ριζικά από αυτό της Μαρίας (Εικόνα 8.11):

«Welcome back, Giannis»

«You have 4 items cataloged across 3 occasions. Build your first outfit to start tracking what you wear.»



Εικόνα 8.11. Dashboard του Γιάννη

Τα στατιστικά εμφανίζονται σε 6 κάρτες: 4 Active items, 0 Saved outfits, 3 Occasions covered, 4 Never worn, 0 Total item wears, 0 Shelved. Η ενότητα «Wardrobe breakdown» παρουσιάζει τρεις στήλες:

- **By Category:** Top: 2, Bottom: 1, Shoes: 1
- **Top Colors:** Black: 1, Orange: 1, Green: 1, Red: 1
- **Occasion Tags:** Casual: 4 items, Party: 1 item, Sport: 2 items. «Not yet tagged: Work, Formal, Travel, Date night»

Οι δυναμικές συμβουλές (Suggestions for you) είναι:

- «Create your first outfit to start tracking what you wear and improve suggestions.»
- «4 active items have never been worn — check today's reminders below.»
- «4 occasions have no tagged items — broader tagging improves suggestions.»

Στο κάτω μέρος εμφανίζονται δύο κάρτες δράσης: «Add clothing» (με κουμπί «Add item») και «Plan an outfit» (με κουμπί «Open outfits»), που καθοδηγούν τον νέο χρήστη στα επόμενα βήματα.

8.3 Σύγκριση Σεναρίων

Ο Πίνακας 8.4 συνοψίζει τις βασικές διαφορές μεταξύ των δύο χρηστών:

Πίνακας 8.4. Σύγκριση σεναρίων Μαρίας και Γιάννη

Λειτουργία	Μαρία	Γιάννης
Ρούχα	18 ενεργά (46 χρήσεις, 5 never worn)	4 ενεργά (0 χρήσεις, 4 never worn)
Κατηγορίες	Top: 5, Bottom: 4, Dress: 2, Outerwear: 3, Shoes: 4	Top: 2, Bottom: 1, Shoes: 1
Χρώματα	Κυρίως ουδέτερα (μαύρο, navy, μπεζ, λευκό, γκρι) και ένα μπλε τζιν	Ζωηρά (πορτοκαλί, πράσινο, κόκκινο, μαύρο)
Occasions	Work: 8, Formal: 8, Casual: 5	Casual: 4, Sport: 2, Party: 1
Προτάσεις WORK	3 σύνολα, score 73–74, style history +15/+16	Κανένα ρούχο για Work — εμφάνιση WARDROBE GAP προτάσεων αγοράς
Προτάσεις CASUAL	3 σύνολα (3–4 κομμάτια), score 67–80, style history +19/+20	2 σύνολα (2–3 κομμάτια), score 32–45, style history 0
Χρωματικό φίλτρο	Κανένα clash (ουδέτερα χρώματα)	Red+Green απορρίφθηκε ως clash
Highlight «Matches your style»	Εμφανίζεται (πλούσιο ιστορικό)	Δεν εμφανίζεται (μηδενικό ιστορικό)

Λειτουργία	Μαρία	Γιάννης
Dashboard μήνυμα	«Your wardrobe has 18 active pieces and 9 saved outfits. 5 items still waiting for a first wear.»	«4 items cataloged across 3 occasions. Build your first outfit»
Προτάσεις αγοράς	Δεν εμφανίζονται (πλήρης ντουλάπα)	4 κάρτες WARDROBE GAP για Work

Η σύγκριση αναδεικνύει ότι το Armoire προσαρμόζεται δυναμικά στον κάθε χρήστη: η ίδια μηχανή παράγει 3 πλούσια σύνολα με υψηλές βαθμολογίες για τη Μαρία, ενώ για τον Γιάννη εντοπίζει χρωματικές ασυμβατότητες, περιορίζεται σε 2 προτάσεις, και όταν δεν μπορεί να παράξει σύνολα (Work), προσφέρει στοχευμένες προτάσεις αγοράς για να καλύψει τα κενά της ντουλάπας.

9. Εγχειρίδιο Χρήστη

9.1 Εισαγωγή

Η εφαρμογή Armoire απευθύνεται σε χρήστες που θέλουν να οργανώσουν τη γκαρνταρόμπα τους ψηφιακά, να λαμβάνουν εξατομικευμένες προτάσεις συνόλων, να καταγράφουν τη χρήση των ρούχων τους και να λαμβάνουν υπενθυμίσεις για ρούχα που δεν χρησιμοποιούν αρκετά. Η εφαρμογή είναι προσβάσιμη μέσω οποιουδήποτε φυλλομετρητή (browser) σε υπολογιστή, tablet ή κινητό τηλέφωνο, καθώς η διεπαφή είναι πλήρως responsive. Δεν απαιτείται εγκατάσταση λογισμικού.

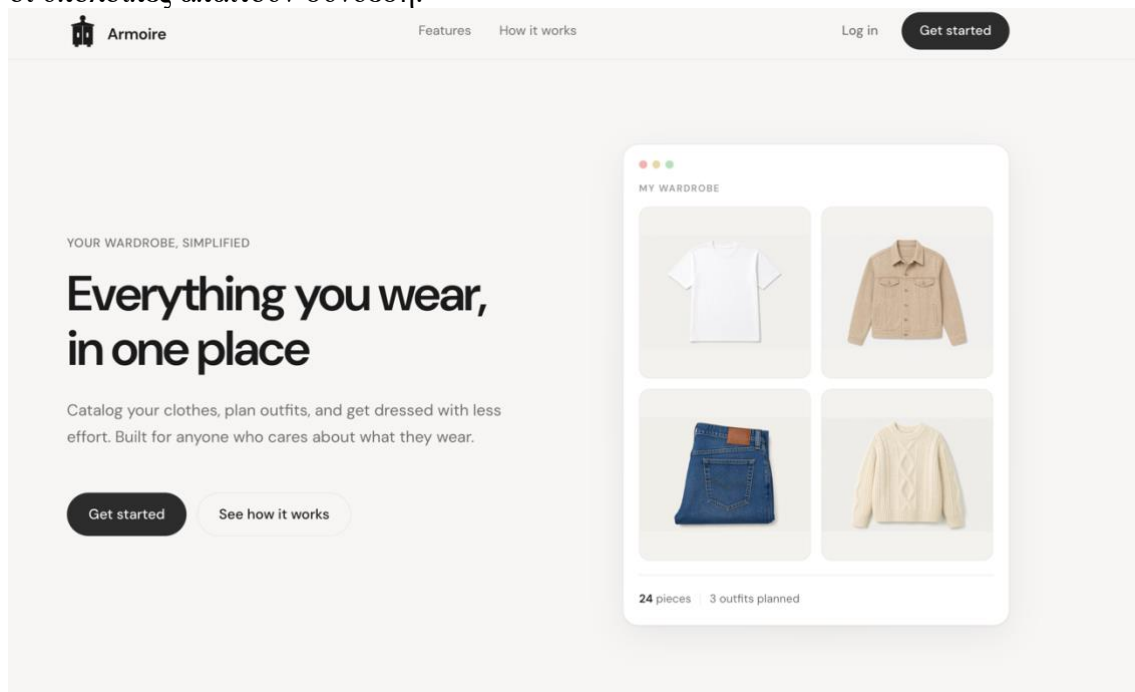
Στις επόμενες ενότητες περιγράφεται αναλυτικά κάθε λειτουργία, με αριθμημένα βήματα και στιγμιότυπα οθόνης.

9.2 Αρχική Σελίδα

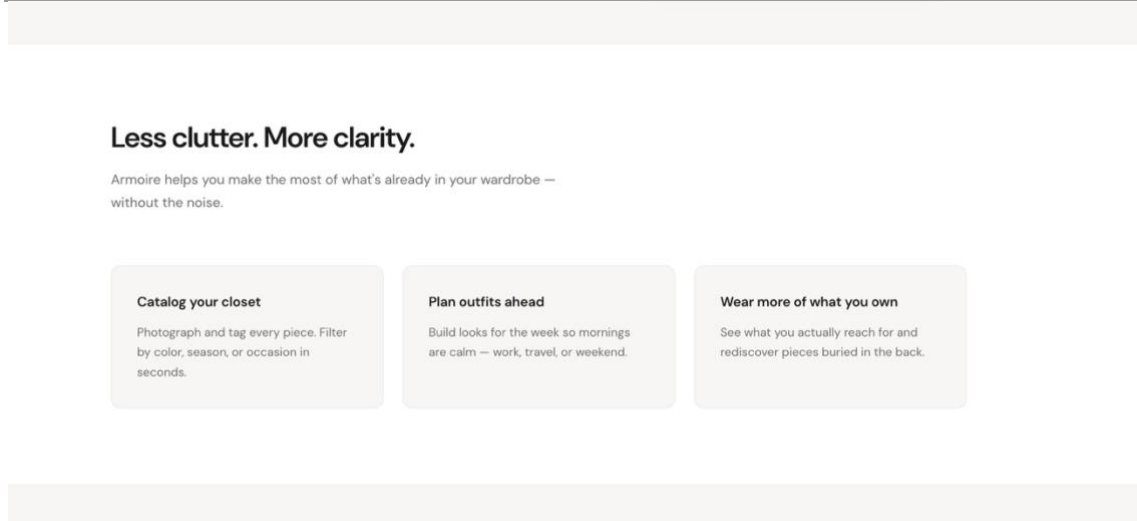
Κατά την πρώτη επίσκεψη, ο χρήστης βλέπει την αρχική σελίδα (landing page) του Armoire. Η σελίδα περιλαμβάνει:

1. Το λογότυπο και το όνομα της εφαρμογής στο επάνω μέρος.
2. Μια σύντομη περιγραφή των δυνατοτήτων.
3. Κουμπί «Get started» που οδηγεί στην εγγραφή.
4. Κουμπί «Log in» στο navigation bar για ήδη εγγεγραμμένους χρήστες.
5. Ενότητες με τα βασικά χαρακτηριστικά (features) σε μορφή καρτών.
6. Footer με πληροφορίες.

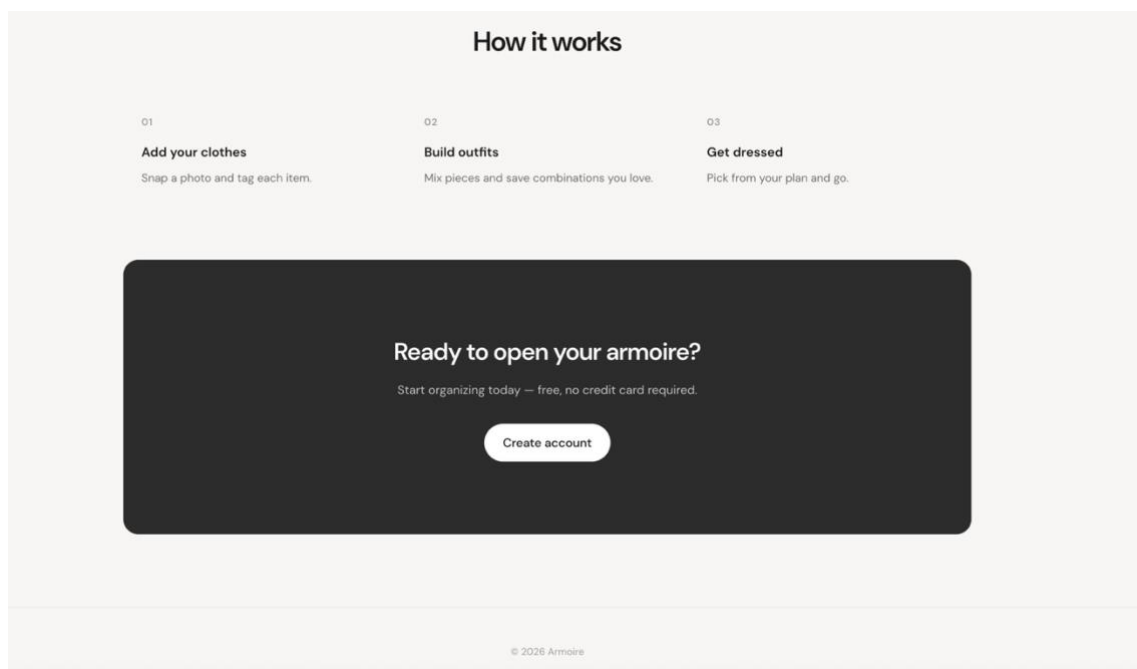
Η αρχική σελίδα, καθώς και οι σελίδες εγγραφής και σύνδεσης, είναι δημόσιες — όλες οι υπόλοιπες απαιτούν σύνδεση.



«Get started» και «Log in» Εικόνα 9.1. Αρχική σελίδα: ενότητα hero με τα κουμπιά «Get started» και «Log in»



Εικόνα 9.2. Αρχική σελίδα: ενότητα χαρακτηριστικών

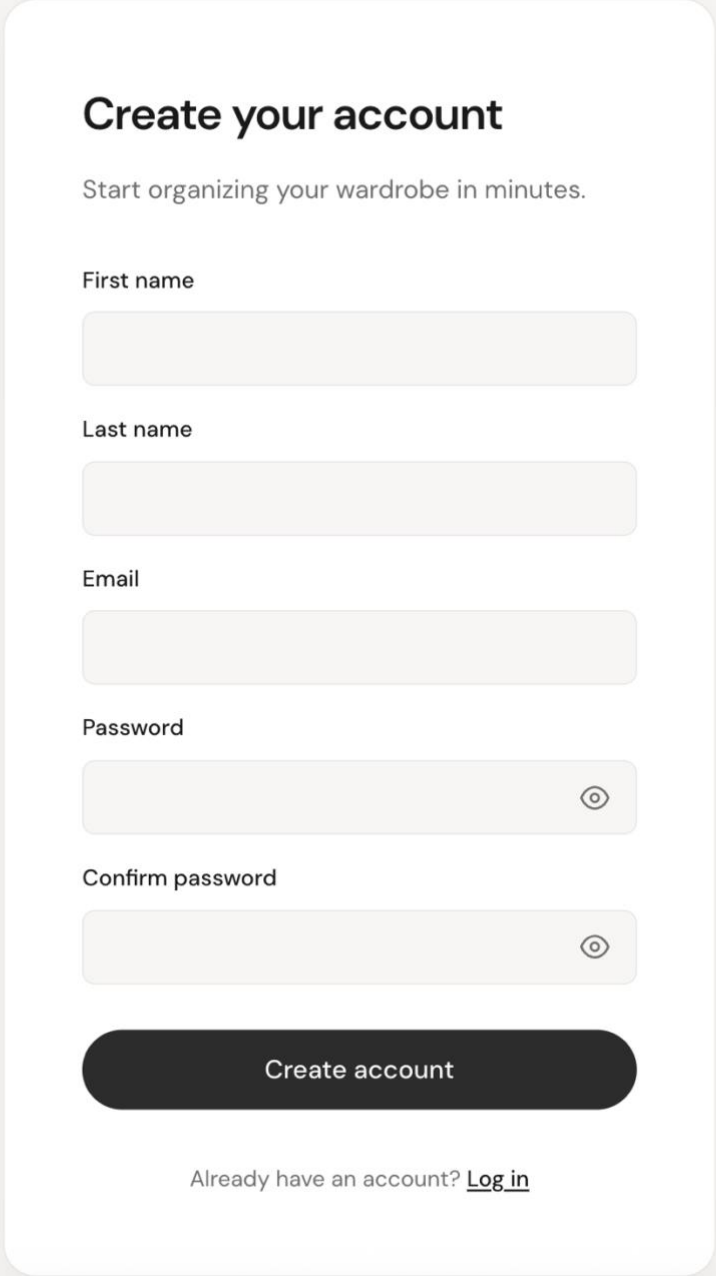


Εικόνα 9.3. Αρχική σελίδα: υποσέλιδο (footer)

9.3 Εγγραφή Νέου Χρήστη

1. Από την αρχική σελίδα, πατήστε «Get started» ή «Sign Up».
2. Συμπληρώστε τα πεδία: Όνομα (First name), Επώνυμο (Last name), Email, Κωδικός πρόσβασης (Password) και Επιβεβαίωση κωδικού (Confirm password).
3. Ο κωδικός πρέπει να πληροί τα ελάχιστα κριτήρια ασφαλείας (τουλάχιστον 8 χαρακτήρες).
4. Πατήστε «Create account».
5. Αν το email υπάρχει ήδη, εμφανίζεται μήνυμα σφάλματος.
6. Μετά την επιτυχή εγγραφή, εμφανίζεται μήνυμα επιτυχίας και μετά από δύο δευτερόλεπτα ο χρήστης μεταφέρεται στη σελίδα σύνδεσης (Login).

Κατά την υποβολή, ο κωδικός κατακερματίζεται (hashing) στην πλευρά του client μέσω SHA-256 πριν αποσταλεί στον διακομιστή, όπου αποθηκεύεται με bcrypt. Έτσι, ο αρχικός κωδικός δεν εμφανίζεται στα αιτήματα προς τον διακομιστή (βλ. ενότητα 7.4).

A registration form titled "Create your account" with the subtitle "Start organizing your wardrobe in minutes." The form contains five input fields: "First name", "Last name", "Email", "Password", and "Confirm password". Each field is a light gray rounded rectangle. The "Password" and "Confirm password" fields have an eye icon on the right side to toggle visibility. Below the fields is a dark gray rounded button labeled "Create account". At the bottom, there is a link: "Already have an account? [Log in](#)".

Create your account

Start organizing your wardrobe in minutes.

First name

Last name

Email

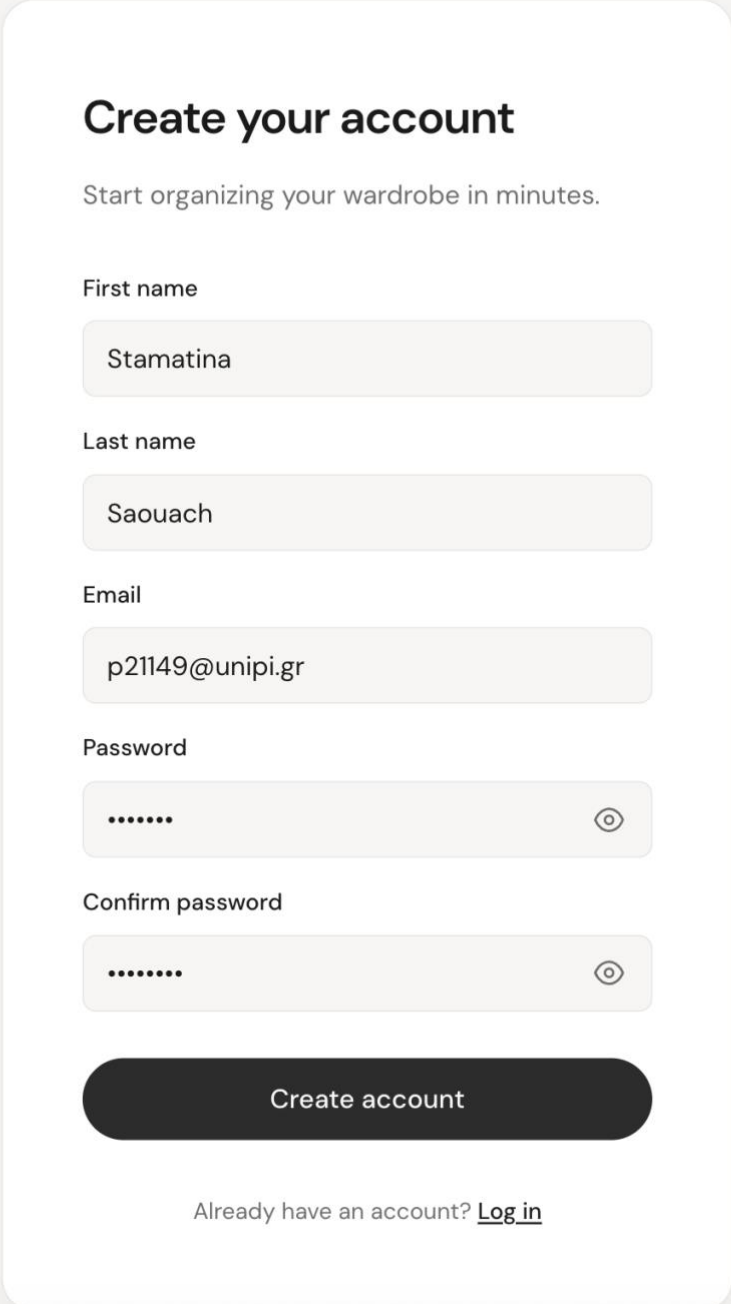
Password

Confirm password

Create account

Already have an account? [Log in](#)

Εικόνα 9.4. Φόρμα εγγραφής (κενή)



Create your account

Start organizing your wardrobe in minutes.

First name

Stamatina

Last name

Saouach

Email

p21149@unipi.gr

Password

.....

Confirm password

.....

Create account

Already have an account? [Log in](#)

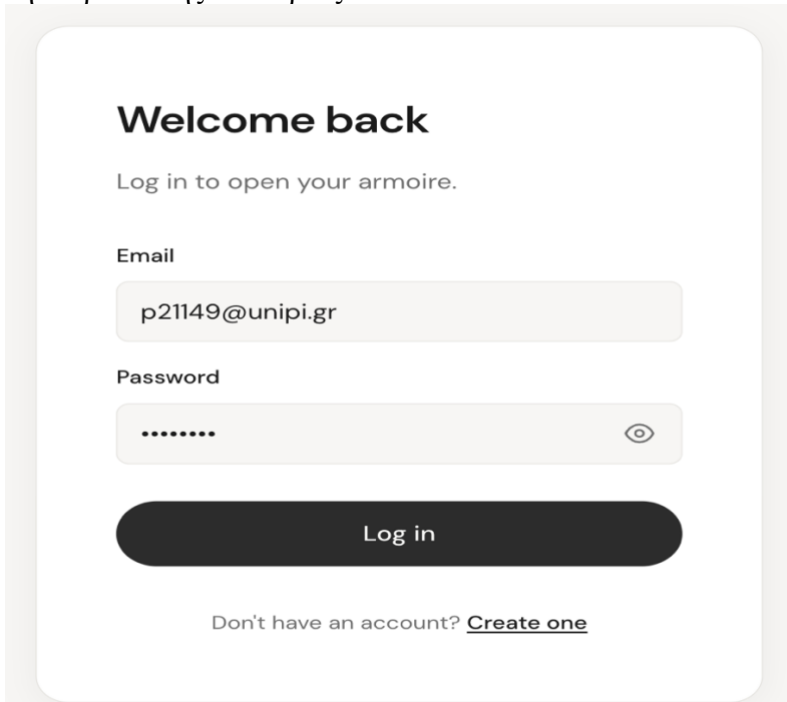
Εικόνα 9.5. Φόρμα εγγραφής (συμπληρωμένη)

9.4 Σύνδεση

1. Από την αρχική σελίδα ή τη γραμμή πλοήγησης, πατήστε «Log in».
2. Εισάγετε Email και Κωδικό πρόσβασης.
3. Πατήστε «Log in».
4. Αν τα στοιχεία είναι σωστά, μεταφέρεστε στο Dashboard.
5. Αν τα στοιχεία είναι λάθος, εμφανίζεται μήνυμα: «Invalid email or password».

Μετά την επιτυχή σύνδεση, ο διακομιστής εκδίδει ένα JWT token που αποθηκεύεται στον browser (sessionStorage). Αυτό το token χρησιμοποιείται αυτόματα σε κάθε

επόμενο αίτημα, χωρίς ο χρήστης να χρειάζεται να εισάγει ξανά τα στοιχεία του κατά τη διάρκεια της συνεδρίας.

A login form titled "Welcome back" with the subtitle "Log in to open your armoire." It features two input fields: "Email" containing "p21149@unipi.gr" and "Password" with masked characters ".....". A "Log in" button is positioned below the fields. At the bottom, there is a link "Don't have an account? [Create one](#)".

Welcome back

Log in to open your armoire.

Email

p21149@unipi.gr

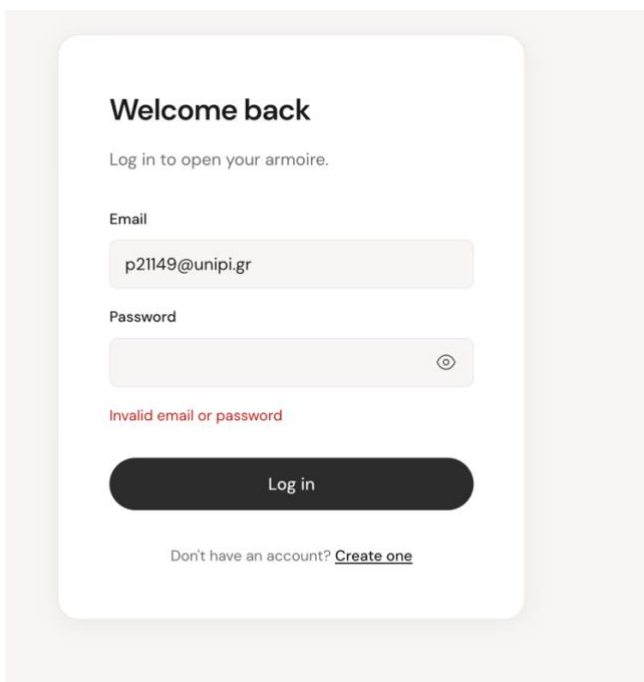
Password

.....

Log in

Don't have an account? [Create one](#)

Εικόνα 9.6. Φόρμα σύνδεσης

The same login form as in Figure 9.6, but with an error message "Invalid email or password" displayed in red text below the password field. The "Email" field still contains "p21149@unipi.gr".

Welcome back

Log in to open your armoire.

Email

p21149@unipi.gr

Password

Invalid email or password

Log in

Don't have an account? [Create one](#)

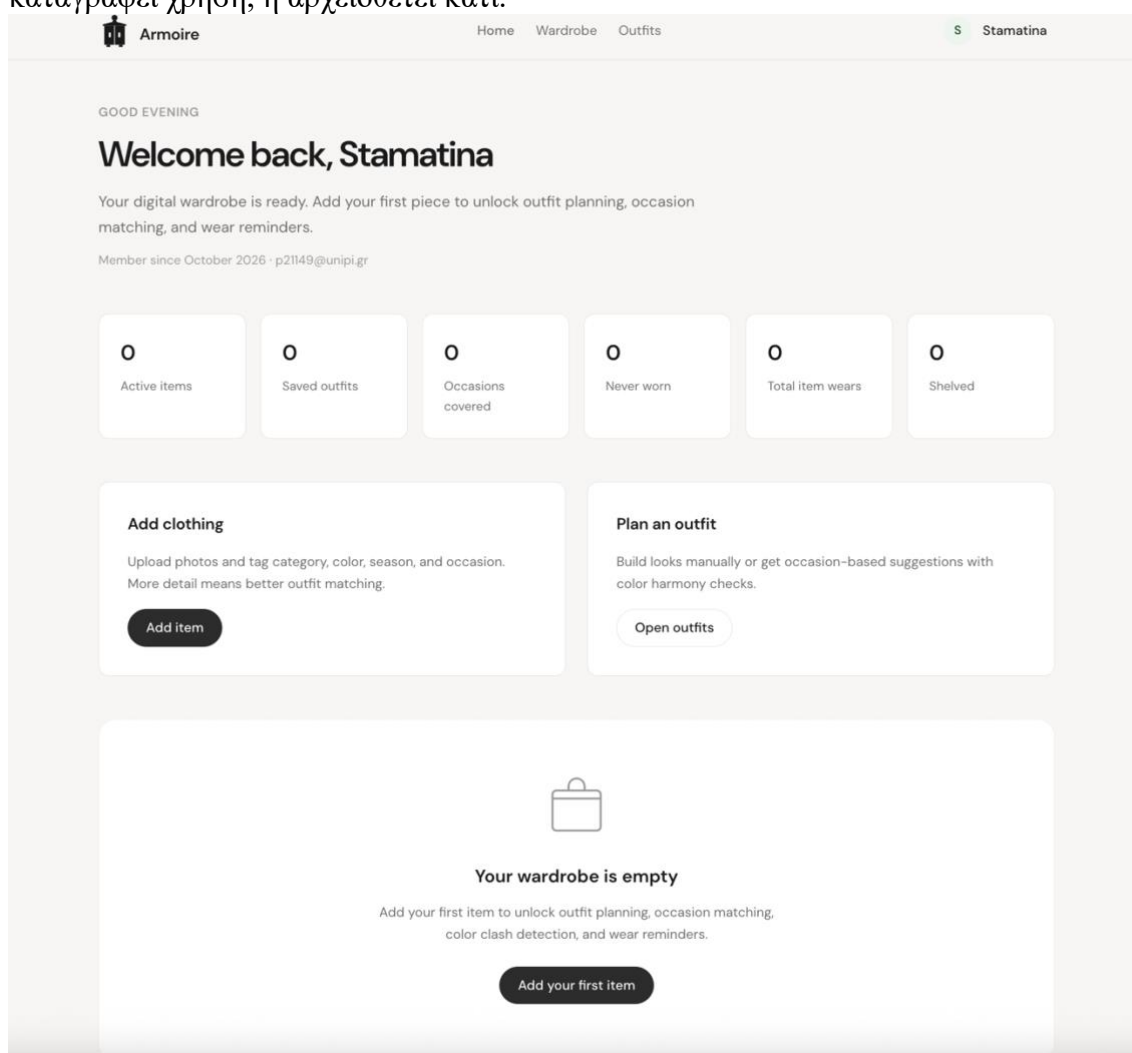
Εικόνα 9.7. Μήνυμα σφάλματος σύνδεσης

9.5 Dashboard (Πίνακας Ελέγχου)

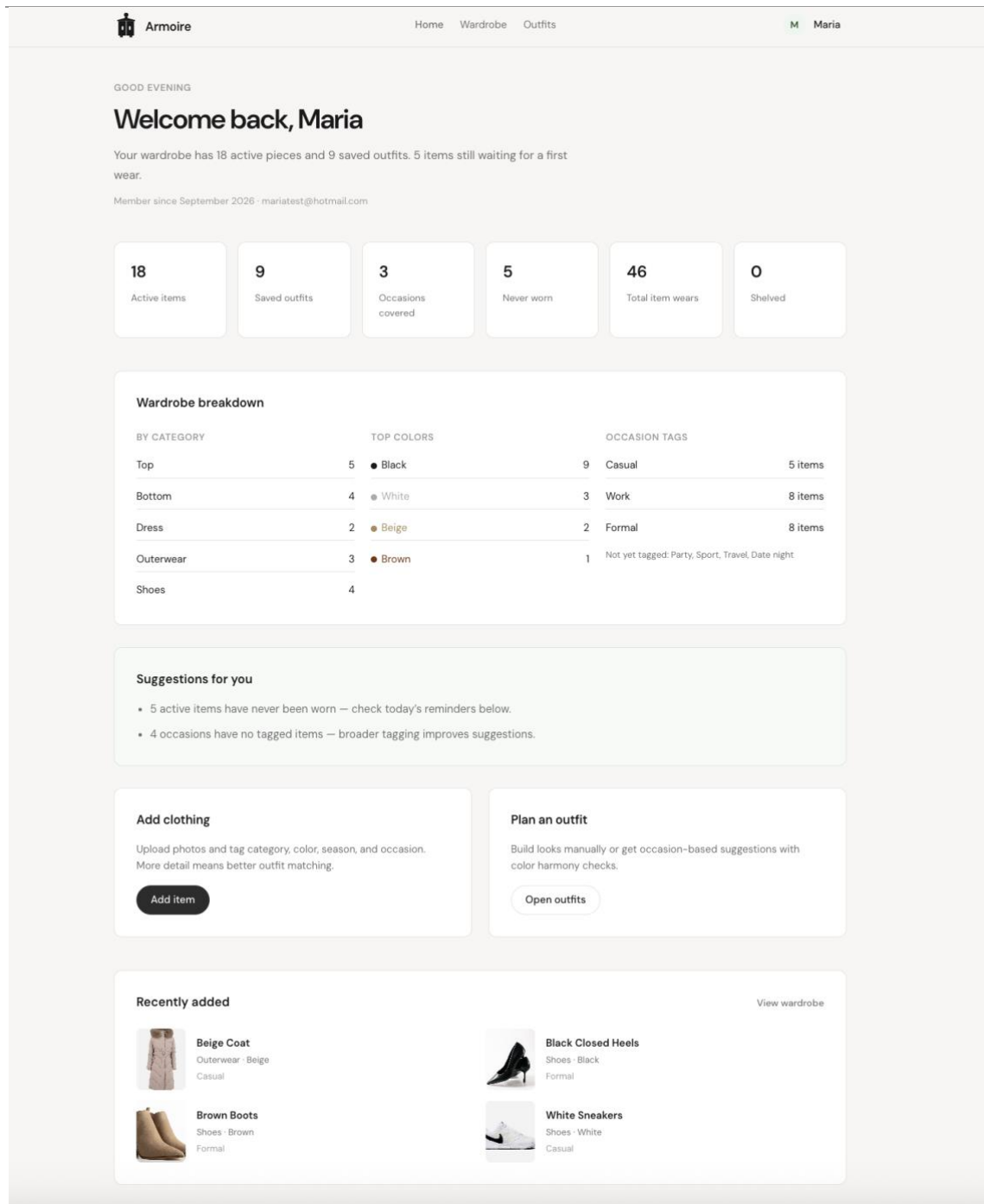
Μετά τη σύνδεση, ο χρήστης μεταφέρεται στο Dashboard, που αποτελεί την κεντρική σελίδα της εφαρμογής (επιλογή «Home» στη γραμμή πλοήγησης). Το Dashboard περιλαμβάνει:

1. Εισαγωγικό μήνυμα: Δυναμικό, προσαρμοσμένο στην κατάσταση της ντουλάπας. Αλλάζει ανάλογα με τον αριθμό ρούχων, συνόλων και περιστάσεων του χρήστη.
2. Στατιστικά ντουλάπας: Αριθμός ενεργών ρούχων, αρχειοθετημένων, συνόλων και συνολικών χρήσεων.
3. Συμβουλές: Δυναμικές, βασισμένες στα δεδομένα (π.χ. «1 active item has never been worn — check today's reminders below.», «No items tagged for work yet — tag pieces to get occasion suggestions.»).
4. Υπενθυμίσεις: Λίστα ρούχων που δεν έχουν χρησιμοποιηθεί πρόσφατα, με ξεχωριστή ένδειξη TRY_TODAY ή SHELVE (εμφανίζονται έως 3 κάθε φορά).

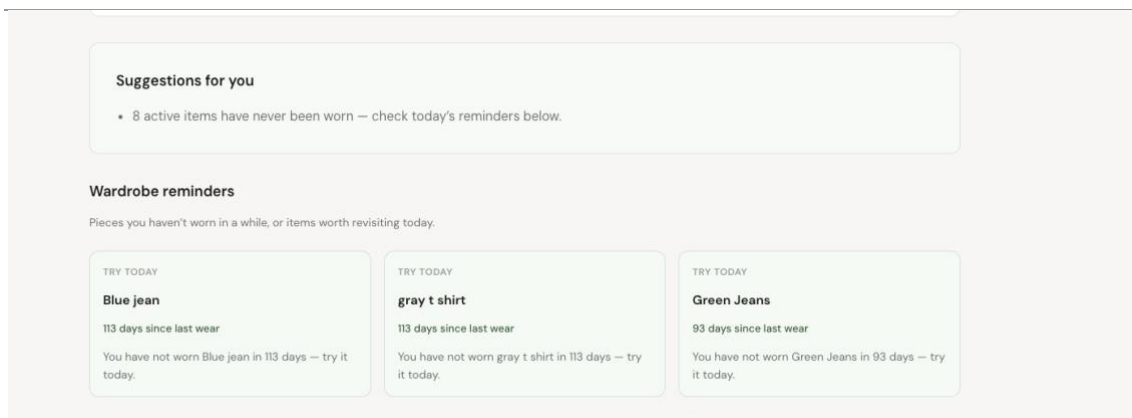
Το Dashboard ενημερώνεται αυτόματα κάθε φορά που ο χρήστης προσθέτει ρούχο, καταγράφει χρήση, ή αρχειοθετεί κάτι.



Εικόνα 9.8. Dashboard νέου χρήστη (κενή γκαρνταρόμπα)



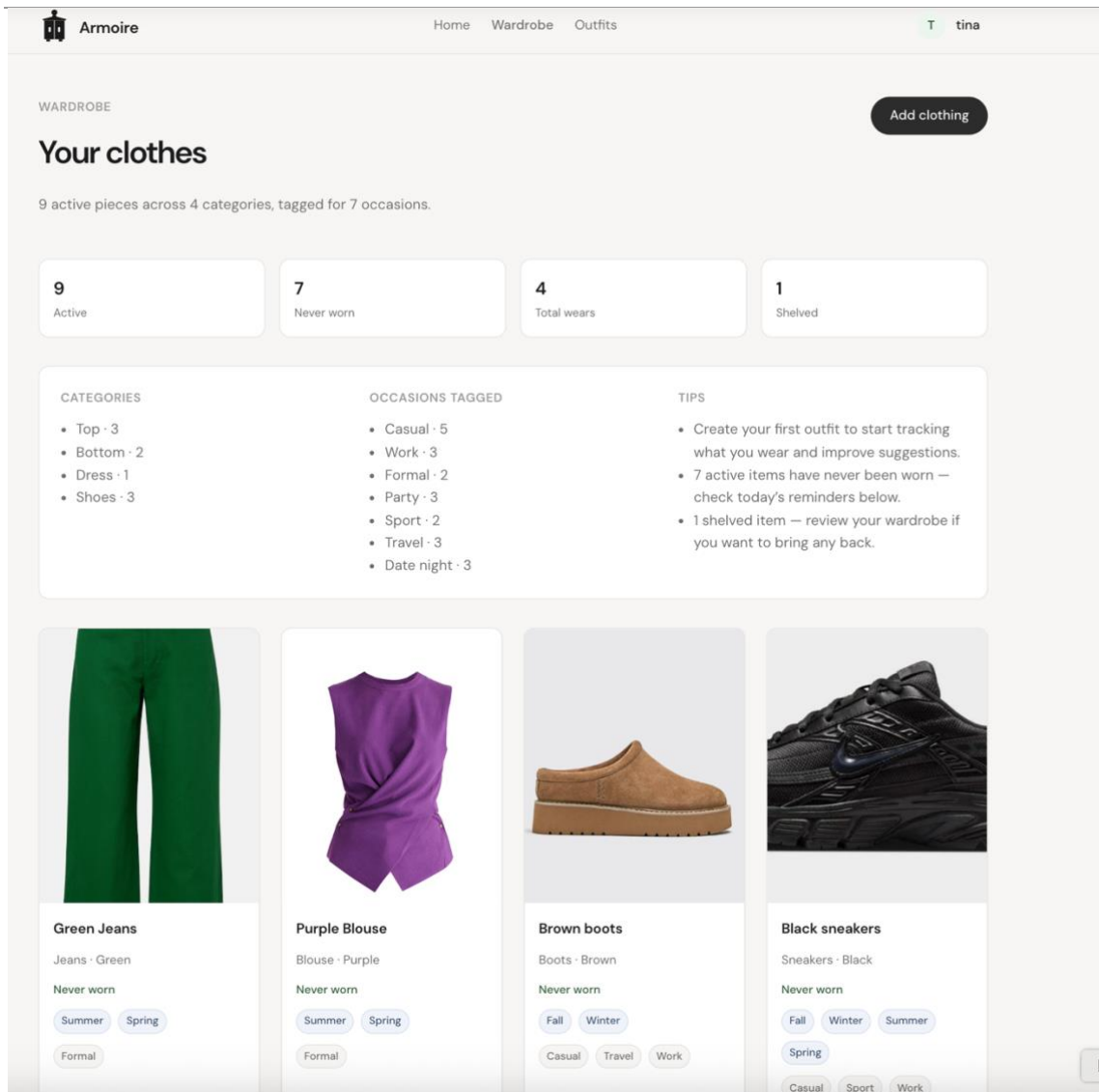
Εικόνα 9.9. Dashboard χρήστη με πλούσια γκαρνταρόμπα



Εικόνα 9.10. Ενότητα υπενθυμίσεων στο Dashboard

9.6 Γκαρνταρόμπα — Προβολή Ρούχων

1. Από τη γραμμή πλοήγησης, πατήστε «Wardrobe».
2. Εμφανίζεται πλέγμα (grid) με όλα τα ενεργά ρούχα σε μορφή καρτών.
3. Κάθε κάρτα εμφανίζει: φωτογραφία, όνομα, κατηγορία και χρώμα, αριθμό χρήσεων (π.χ. «7× total» ή «Never worn»), ετικέτες εποχής και περιστασης, και κουμπί «Remove» για διαγραφή.
4. Στο πάνω μέρος εμφανίζονται τα στατιστικά (Active, Never worn, Total wears, Shelved), η κάλυψη κατηγοριών, οι ετικεταρισμένες περιστάσεις και δυναμικές συμβουλές.
5. Τα αρχειοθετημένα ρούχα δεν εμφανίζονται στο πλέγμα — ο αριθμός τους φαίνεται μόνο στα στατιστικά. Η αρχειοθέτηση γίνεται μέσω των υπενθυμίσεων SHELVE στο Dashboard (βλ. ενότητα 9.12) ή από τη φόρμα επεξεργασίας ρούχου, με το πεδίο Status (Active/Shelved), από όπου ένα ρούχο μπορεί και να επανέλθει στην ενεργή γκαρνταρόμπα.



Εικόνα 9.11. Η γκαρνταρόμπα σε μορφή πλέγματος με κάρτες ρούχων

9.7 Προσθήκη Νέου Ρούχου

1. Από τη σελίδα Γκαρνταρόμπας, πατήστε «Add item».
2. Ανεβάστε μια φωτογραφία του ρούχου (υποστηρίζονται JPEG, PNG, WebP, μέγιστο 5MB).
3. Συμπληρώστε το όνομα (π.χ. «Μαύρο πουκάμισο»).
4. Επιλέξτε κατηγορία (Top, Bottom, Dress, Shoes, Outerwear, Accessory).
5. Επιλέξτε υποκατηγορία (π.χ. T-shirt, Shirt, Blouse για Top).
6. Επιλέξτε κύριο χρώμα (υποχρεωτικό) και δευτερεύον χρώμα (προαιρετικό).
7. Επιλέξτε εποχές (Spring, Summer, Fall, Winter — τουλάχιστον μία, μπορείτε πολλαπλές).
8. Επιλέξτε περιστάσεις (Casual, Work, Formal, Party, Sport, Date Night, Travel — τουλάχιστον μία, μπορείτε πολλαπλές).
9. Προαιρετικά, προσθέστε σημειώσεις.
10. Πατήστε «Save item».

Σημαντικό: Η υποκατηγορία πρέπει να αντιστοιχεί στην κατηγορία — π.χ. δεν μπορείτε να επιλέξετε υποκατηγορία «Sneakers» (παπούτσια) αν η κατηγορία είναι «Top».

Επίσης, κύριο και δευτερεύον χρώμα δεν μπορεί να είναι ίδια. Αν κάποιος κανόνας παραβιαστεί, εμφανίζεται μήνυμα σφάλματος.

Η σωστή σήμανση εποχών και περιστάσεων είναι κρίσιμη για τη λειτουργία του συστήματος: ρούχα χωρίς ετικέτα περιστάσης δεν θα εμφανιστούν στις αντίστοιχες προτάσεις.

WARDROBE

Back to wardrobe

Add clothing

Tag each piece with category, color, seasons, and occasions so Armoire can recommend outfits later.

Photo

Drop a photo here or browse
JPEG, PNG, or WebP up to 5 MB

Choose photo

Name

Blue oxford shirt

Details

Category: Top Type: Select type

Primary color: Black Secondary color: None

Seasons

Spring Summer Fall Winter

Occasions

These tags help match outfits to events like work, travel, or date night.

Casual Work Formal Party Sport

Travel Date night

Notes

Optional details like fit, fabric, or where you usually wear it.

Save item

Εικόνα 9.12. Φόρμα προσθήκης ρούχου (κενή)

WARDROBE

Back to wardrobe

Add clothing

Tag each piece with category, color, seasons, and occasions so Armoire can recommend outfits later.

Photo



Change photo Remove

Name

Black Shirt

Details

Category: Top

Type: Blouse

Primary color: Black

Secondary color: None

Seasons

Spring Summer Fall Winter

Occasions

These tags help match outfits to events like work, travel, or date night.

Casual Work Formal Party Sport

Travel Date night

Notes

Optional details like fit, fabric, or where you usually wear it.

Εικόνα 9.13. Φόρμα προσθήκης ρούχου (συμπληρωμένη, με φωτογραφία)

Add clothing

Tag each piece with category, color, seasons, and occasions so Armoire can recommend outfits later.

Photo



Change photo Remove

Name

Black Shirt

Details

Category: Top

Type: Blouse

Primary color: Black

Secondary color: Black

Seasons

Spring Summer Fall Winter

Occasions

These tags help match outfits to events like work, travel, or date night.

Casual Work Formal Party Sport

Travel Date night

Notes

Optional details like fit, fabric, or where you usually wear it.

Secondary color must differ from the primary color.

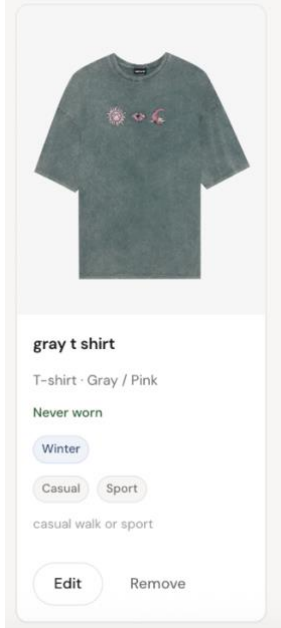
Save item

Εικόνα 9.14. Φόρμα προσθήκης ρούχου με μήνυμα σφάλματος

9.8 Επεξεργασία Ρούχου

- Πατήστε το κουμπί «Edit» στην κάρτα ενός ρούχου για να ανοίξετε τη φόρμα επεξεργασίας.
- Τα πεδία εμφανίζονται προσυμπληρωμένα με τα τρέχοντα στοιχεία του ρούχου (όνομα, κατηγορία, χρώμα, περιστάσεις, εποχές, κατάσταση, σημειώσεις)· μπορείτε επίσης να αντικαταστήσετε τη φωτογραφία.
- Τροποποιήστε όποιο πεδίο επιθυμείτε και πατήστε «Save changes» για αποθήκευση.

Η φόρμα επεξεργασίας είναι ίδια με τη φόρμα προσθήκης, με τη διαφορά ότι τα πεδία είναι ήδη συμπληρωμένα.



Εικόνα 9.15. Κάρτα ρούχου με κουμπί «Edit»

Update tags, photo, or status. Occasion tags affect which outfits Armoire can suggest.

Current photo



Replace photo (optional)

Drop a photo here or browse

JPEG, PNG, or WebP up to 5 MB

Choose photo

Name

gray t shirt

Status

Active

Details

Category: Top

Type: T-shirt

Primary color: Gray

Secondary color: Pink

Seasons

Spring Summer Fall Winter

Occasions

Casual Work Formal Party Sport

Travel Date night

Notes

casual walk or sport

Save changes

Εικόνα 9.16. Φόρμα επεξεργασίας με προσυμπληρωμένα πεδία**9.9 Δημιουργία Συνόλου (Outfit) Χειροκίνητα**

1. Από τη γραμμή πλοήγησης, πατήστε «Outfits».
2. Πατήστε «Create outfit».
3. Δώστε όνομα στο σύνολο (π.χ. «Monday Work Look»).
4. Επιλέξτε περίσταση.
5. Επιλέξτε ρούχα από τη γκαρνταρόμπα. Πρέπει να επιλεγούν τουλάχιστον δύο ρούχα και να υπάρχει:
 - Ένα φόρεμα Ή ένα top + ένα bottom (όχι και τα δύο).
 - Παπούτσια και πανωφόρι είναι προαιρετικά.
6. Αν επιλεγεί αρχειοθετημένο ρούχο, η αποθήκευση απορρίπτεται με μήνυμα σφάλματος («Shelved items cannot be used in outfits»).
7. Πατήστε «Save outfit».

Μετά την αποθήκευση, το σύνολο εμφανίζεται στη λίστα Outfits με μικρογραφίες των ρούχων που περιέχει.

OUTFITS

Back to outfits

Create outfit

Choose a dress or a top-and-bottom combo, then add shoes or outerwear if you like.

Outfit name

Weekend brunch

Occasion

Casual

Notes

Optional styling notes.

Pick items

Selected: 0

Top



Purple Blouse



Black top



gray t shirt

Bottom



Green Jeans



Blue Jeans



Blue jean

Dress



Black dress

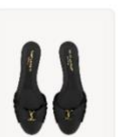
Shoes



Brown boots



Black sneakers



Black sandals

Save outfit

Εικόνα 9.17. Φόρμα δημιουργίας συνόλου: επιλογή ρούχων

Armoire: Διαδίκτυακή Εφαρμογή Καταλογοποίησης Γκαρνταρόμπας και Προτάσεων Ρουχισμού και Εμφάνισεων

85

Outfit name

outfit for school

Occasion

Casual

Notes

Optional styling notes.

Pick items

Selected: 3

Top

Purple Blouse

Black top

gray t shirt

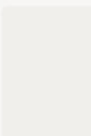
Bottom

Green Jeans

Blue Jeans

Blue jean

Dress

Black dress

Shoes

Brown boots

Black sneakers

Black sandals

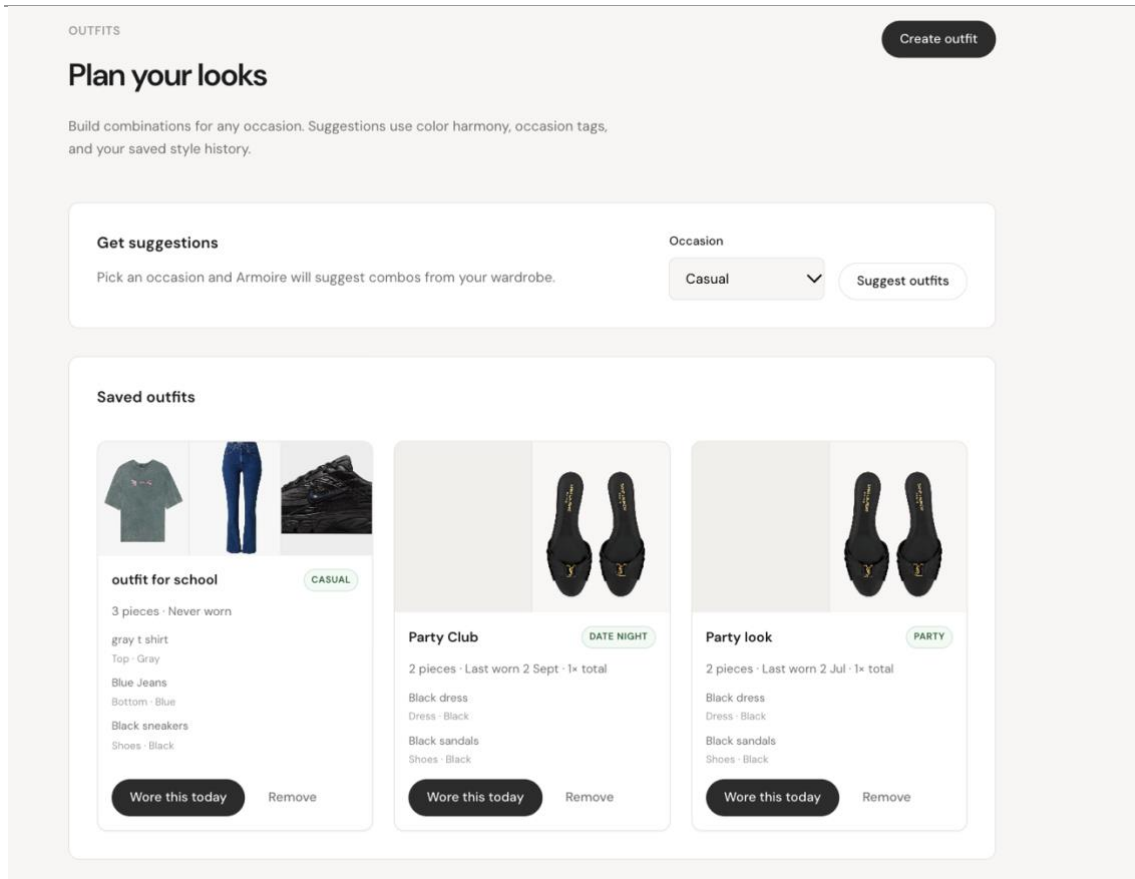
Shelved items cannot be used in outfits

Save outfit

Εικόνα 9.18. Μήνυμα σφάλματος για αρχειοθετημένο ρούχο

Armoire: Διαδικτυακή Εφαρμογή Καταλογοποίησης Γκαρνταρόμπας και Προτάσεων Ρουχισμού και Εμφάνισεων

86

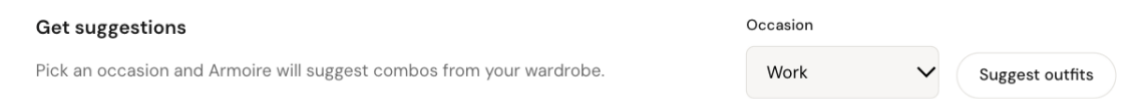


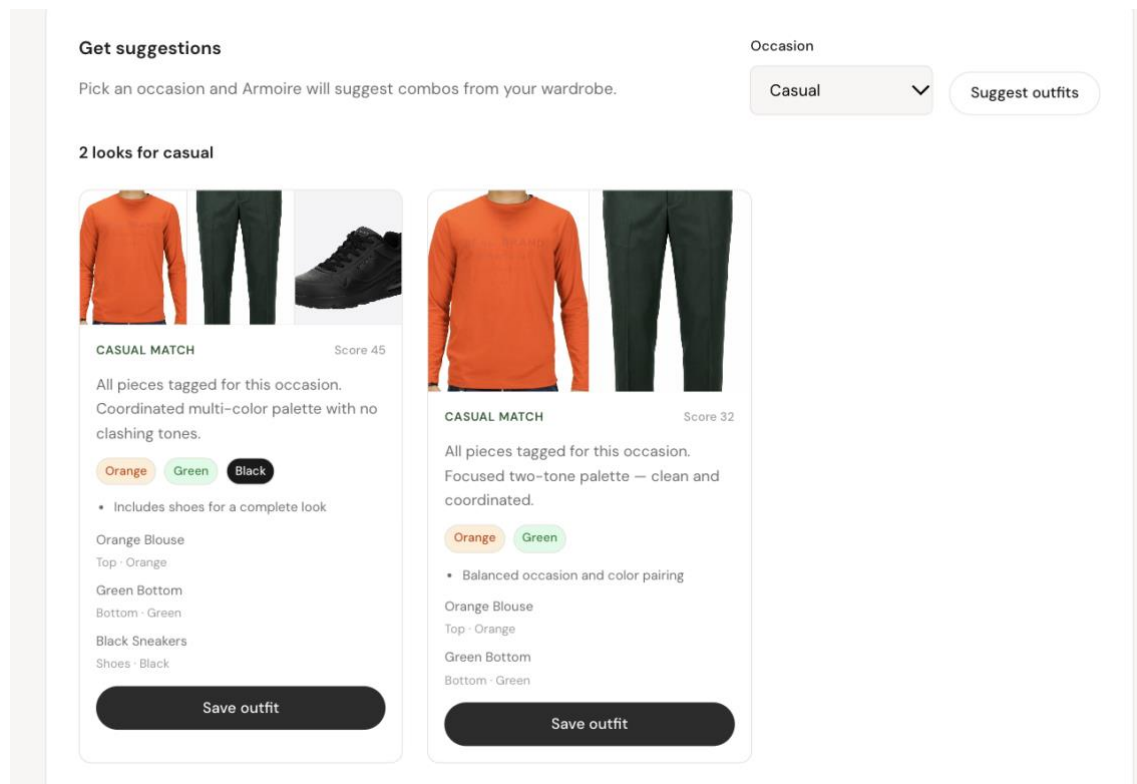
Εικόνα 9.19. Λίστα αποθηκευμένων συνόλων

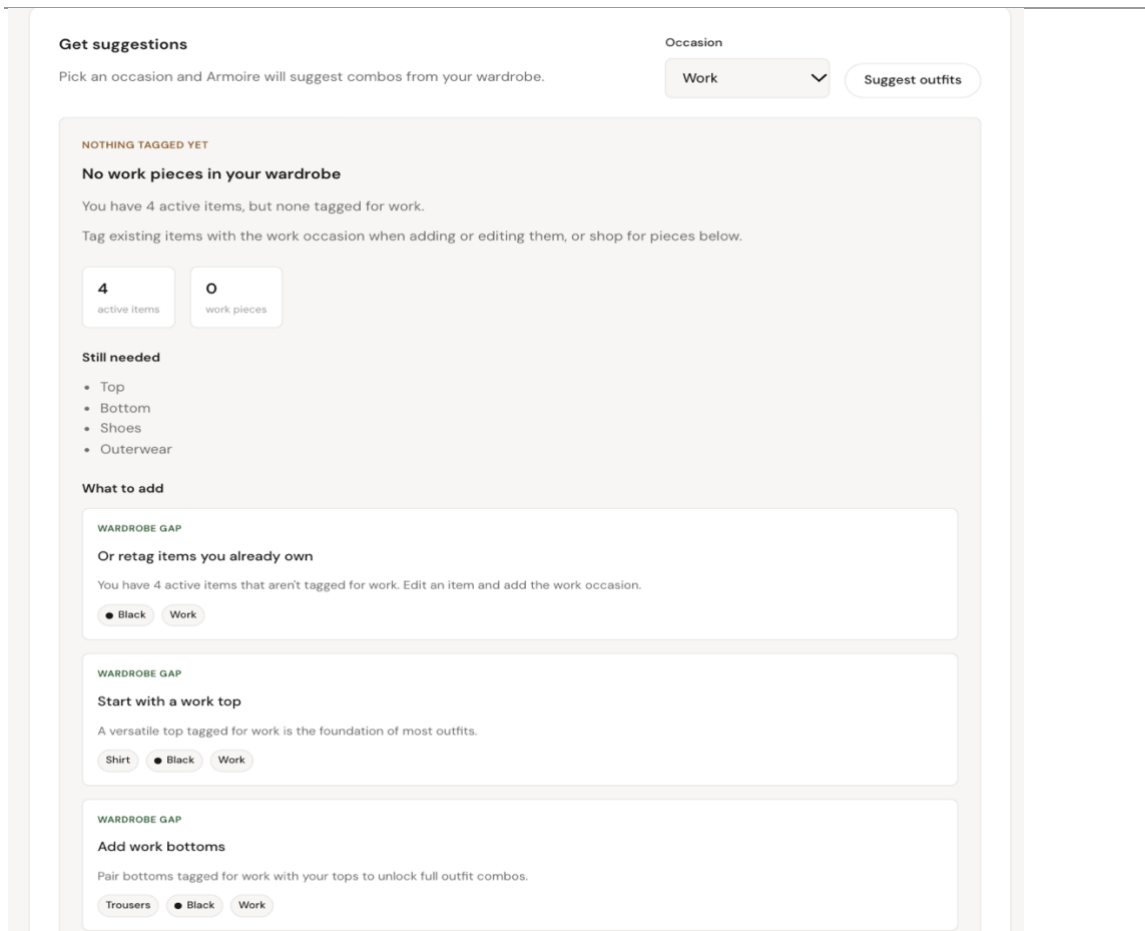
9.10 Λήψη Προτάσεων Εμφάνισης

1. Από τη γραμμή πλοήγησης, πατήστε «Outfits»: οι προτάσεις βρίσκονται στην ενότητα «Get suggestions» της σελίδας «Plan your looks».
2. Επιλέξτε μία περίπτωση από τη λίστα (π.χ. Casual, Work, Formal).
3. Πατήστε «Suggest outfits».
4. Εμφανίζονται κάρτες προτάσεων, κάθε μία με:
 - Τα ρούχα του συνόλου (με φωτογραφίες).
 - Βαθμολογία (score).
 - Χρωματική παλέτα.
 - Εξήγηση (π.χ. «All-neutral palette — versatile and easy to accessorize»).
 - Ένδειξη preference boost αν ταιριάζει με το ιστορικό του χρήστη.
5. Για να αποθηκεύσετε μια πρόταση, πατήστε «Save outfit».
6. Για να καταγράψετε ότι τη φορέσατε, αφού την αποθηκεύσετε, πατήστε «Wore this today» στη λίστα αποθηκευμένων συνόλων.

Αν δεν υπάρχουν αρκετά ρούχα ή κατάλληλα ρούχα, αντί για κάρτες προτάσεων εμφανίζεται ένα διαγνωστικό μήνυμα που εξηγεί γιατί δεν βρέθηκαν προτάσεις και τι μπορεί να κάνει ο χρήστης (π.χ. «You have 4 active items, but none tagged for work.»). Ταντόχρονα εμφανίζονται προτάσεις αγορών.



Εικόνα 9.20. Επιλογή περίπτωσης για προτάσεις**Εικόνα 9.21. Κάρτες προτάσεων με βαθμολογία και επεξήγηση**

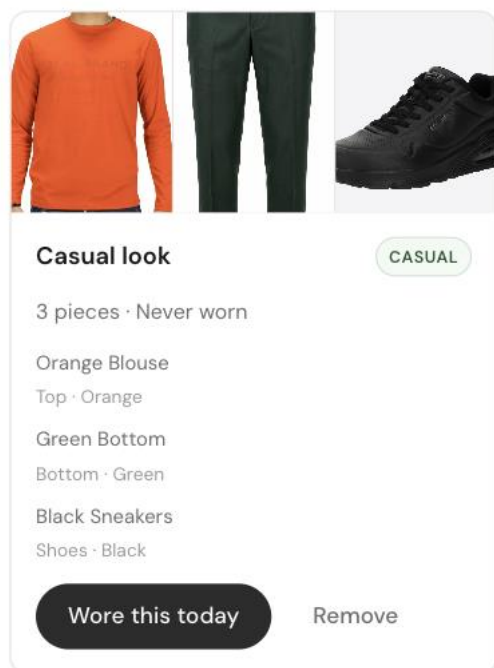


Εικόνα 9.22. Κενή κατάσταση: μήνυμα «No work pieces in your wardrobe» με προτάσεις αγορών

9.11 Καταγραφή Χρήσης (Wear Tracking)

1. Μεταβείτε στη σελίδα Outfits (λίστα αποθηκευμένων συνόλων).
2. Πατήστε «Wore this today» στο σύνολο που φοράτε σήμερα.
3. Η εφαρμογή ενημερώνει αυτόματα:
 - Τον μετρητή χρήσεων (wear count) κάθε ρούχου στο σύνολο.
 - Την ημερομηνία τελευταίας χρήσης (last worn at).
 - Τα στατιστικά στο Dashboard.
4. Η καταγραφή χρήσης επηρεάζει:
 - Τις υπενθυμίσεις (ρούχα που φοράτε δεν θα εμφανίζονται ως TRY_TODAY).
 - Το προφίλ προτιμήσεων (ρούχα που φοράτε συχνά βαρύνουν περισσότερο στις μελλοντικές προτάσεις).

Σημείωση: Η καταγραφή γίνεται σε επίπεδο συνόλου, αλλά διαδίδεται (cascade) σε κάθε μεμονωμένο ρούχο που περιέχει το σύνολο.



Εικόνα 9.23. Κουμπί «Wore this today» σε κάρτα συνόλου

9.12 Υπενθυμίσεις και Αρχαιοθέτηση μέσω Dashboard

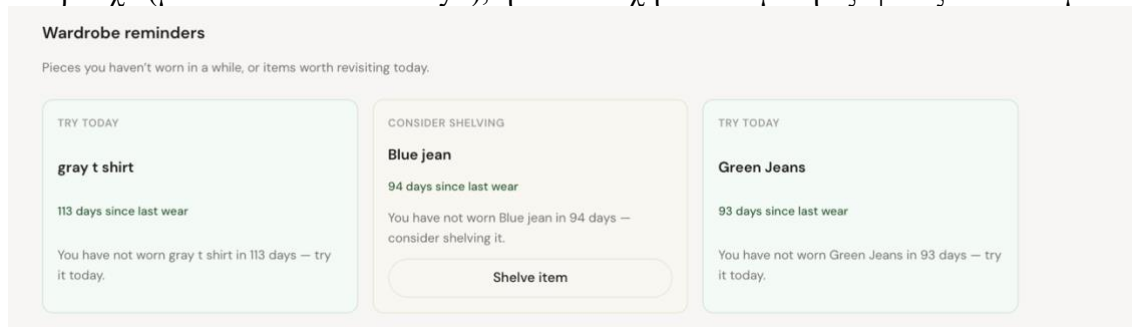
Στο κάτω μέρος του Dashboard εμφανίζονται υπενθυμίσεις δύο τύπων:

- TRY_TODAY (πορτοκαλί): Ρούχα που δεν έχετε φορέσει εδώ και 30+ ημέρες, ή ρούχα που δεν έχετε φορέσει ποτέ μετά από 14+ ημέρες. Η εφαρμογή σας προτρέπει να τα δοκιμάσετε σήμερα.
- SHELVE (κόκκινο): Ρούχα που δεν έχετε φορέσει εδώ και 90+ ημέρες. Η εφαρμογή σας προτρέπει να τα αρχαιοθετήσετε.

Κάθε υπενθύμιση εμφανίζει:

1. Φωτογραφία και όνομα ρούχου.
2. Πόσες ημέρες έχουν περάσει από την τελευταία χρήση.
3. Κουμπί για αρχαιοθέτηση (στις SHELVE υπενθυμίσεις).

Οι υπενθυμίσεις ταξινομούνται με τα πιο αδρανή ρούχα πρώτα. Αν ο χρήστης φορέσει ένα ρούχο (μέσω «Wore this today»), η αντίστοιχη υπενθύμιση εξαφανίζεται αυτόματα.



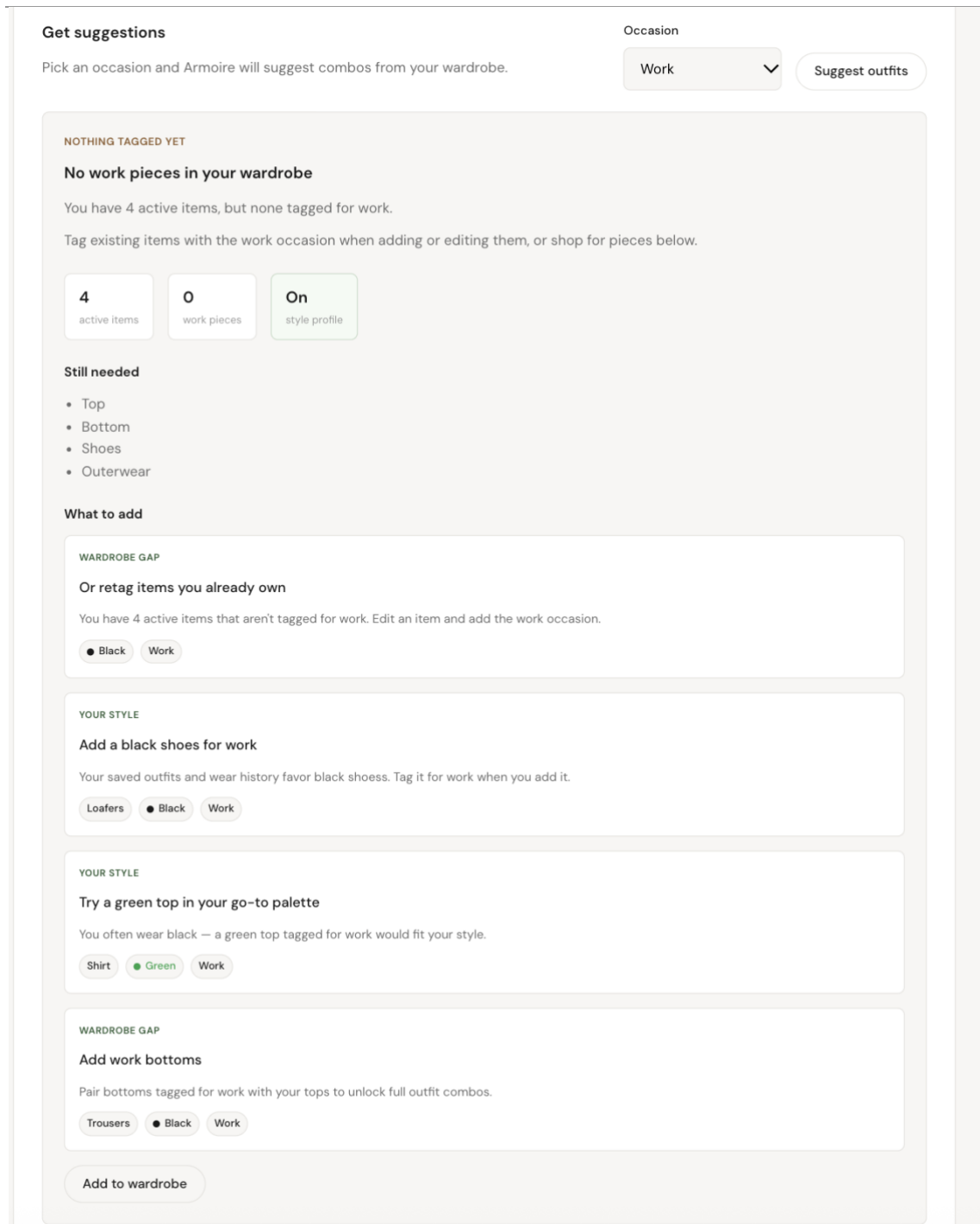
Εικόνα 9.24. Υπενθυμίσεις TRY_TODAY και SHELVE στο Dashboard

9.13 Προτάσεις Αγορών (Purchase Suggestions)

Όταν η μηχανή προτάσεων δεν μπορεί να σχηματίσει σύνολα (λόγω ελλιπούς ντουλάπας ή χρωματικών clashes), εμφανίζονται αυτόματα προτάσεις αγορών δύο τύπων:

1. **Wardrobe Gap:** Εντοπίζει τις κατηγορίες που λείπουν για τη συγκεκριμένη περίπτωση. Παραδείγματα: «Or retag items you already own — You have 4 active items that aren't tagged for work. Edit an item and add the work occasion.» και «Add work bottoms — Pair bottoms tagged for work with your tops to unlock full outfit combos.» Κάθε κάρτα εμφανίζει chips με τον προτεινόμενο τύπο, χρώμα και περίσταση (π.χ. Trousers, Black, Work).
2. **Your Style:** Εμφανίζεται μόνο αν ο χρήστης έχει προφίλ προτιμήσεων (τουλάχιστον μία αποθηκευμένη εμφάνιση ή καταγραφή χρήσης). Βασίζεται στο αποθηκευμένο προφίλ στυλ και προτείνει ρούχα που ταιριάζουν στις προτιμήσεις του. Παραδείγματα: «Add a black shoes for work — Your saved outfits and wear history favor black shoes.» και «Try a green top in your go-to palette — You often wear black — a green top tagged for work would fit your style.»

Οι προτάσεις αγορών δεν περιέχουν συνδέσμους σε καταστήματα — αποτελούν καθοδήγηση για τον χρήστη ώστε να εμπλουτίσει τη ντουλάπα του στοχευμένα. Στο κάτω μέρος εμφανίζεται κουμπί «Add to wardrobe» που οδηγεί απευθείας στη σελίδα προσθήκης ρούχου.



Εικόνα 9.25. Προτάσεις αγορών (wardrobe gap και retagging)

9.14 Ρυθμίσεις Λογαριασμού

1. Πατήστε το εικονίδιο προφίλ (αρχικό και όνομα) στη γραμμή πλοήγησης.
2. Επιλέξτε «Settings».

Από τη σελίδα ρυθμίσεων μπορείτε:

Αλλαγή στοιχείων:

1. Αλλάξτε το όνομα ή το επώνυμο (το email δεν μπορεί να αλλάξει στην τρέχουσα έκδοση).

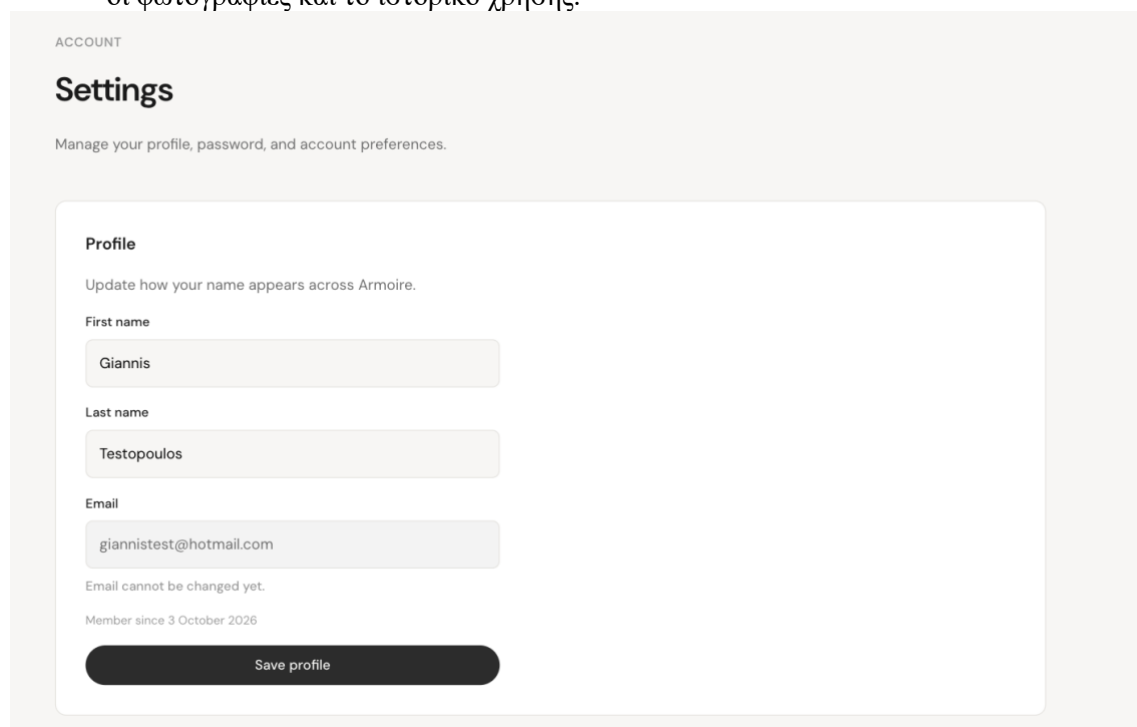
2. Πατήστε «Save profile».

Αλλαγή κωδικού:

1. Εισάγετε τον τρέχοντα κωδικό.
2. Εισάγετε τον νέο κωδικό.
3. Ο νέος κωδικός πρέπει να διαφέρει από τον τρέχοντα.
4. Πατήστε «Change password».

Διαγραφή λογαριασμού:

1. Πατήστε «Delete account».
2. Εμφανίζεται μήνυμα επιβεβαίωσης μέσα στη σελίδα· πατήστε «Yes» για να συνεχίσετε — η ενέργεια είναι μη αναστρέψιμη.
3. Με την επιβεβαίωση, διαγράφονται οριστικά: ο λογαριασμός, όλα τα ρούχα, τα σύνολα, οι φωτογραφίες και το ιστορικό χρήσης.



ACCOUNT

Settings

Manage your profile, password, and account preferences.

Profile

Update how your name appears across Armoire.

First name

Giannis

Last name

Testopoulos

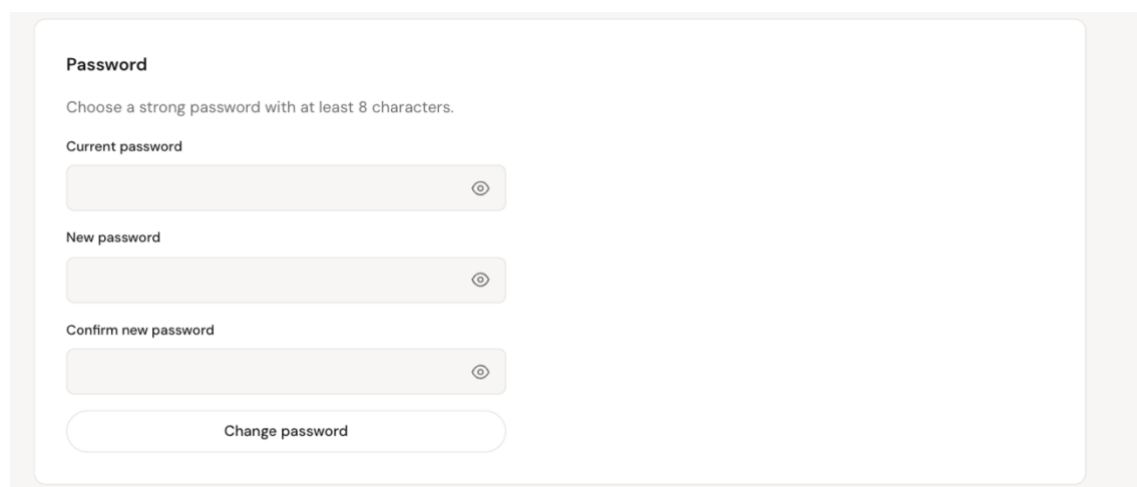
Email

giannistest@hotmail.com

Email cannot be changed yet.

Member since 3 October 2026

Save profile

Εικόνα 9.26. Ρυθμίσεις: στοιχεία λογαριασμού

Password

Choose a strong password with at least 8 characters.

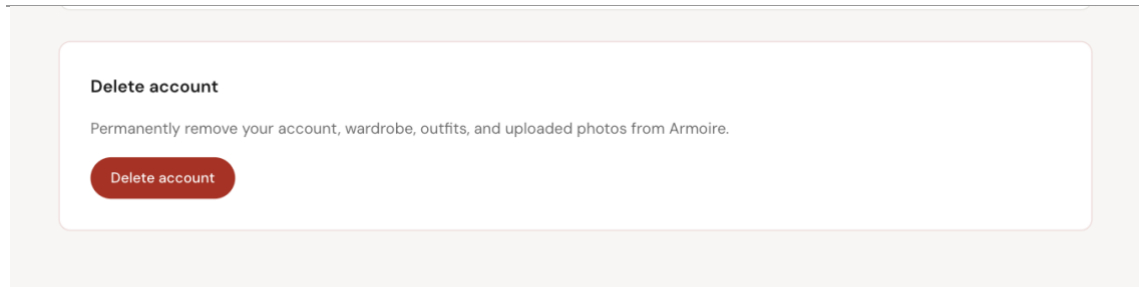
Current password

New password

Confirm new password

Change password

Εικόνα 9.27. Ρυθμίσεις: αλλαγή κωδικού



Εικόνα 9.28. Ρυθμίσεις: διαγραφή λογαριασμού

9.15 Αποσύνδεση

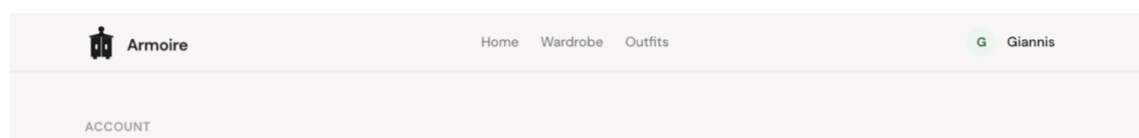
1. Πατήστε το εικονίδιο προφίλ (αρχικό και όνομα).
2. Πατήστε «Log out».
3. Το JWT token διαγράφεται από τον browser (sessionStorage).
4. Μεταφέρεστε στη σελίδα σύνδεσης (Login).

Σημείωση: Επειδή το token αποθηκεύεται στο sessionStorage, διαγράφεται αυτόματα όταν κλείσει η καρτέλα ή ο browser. Σε κάθε περίπτωση, η συνεδρία λήγει μετά από 12 ώρες (εκπνοή JWT token).

9.16 Γραμμή Πλοήγησης

Η γραμμή πλοήγησης (navigation bar) είναι πάντα ορατή στο πάνω μέρος της σελίδας και περιλαμβάνει (σε μικρές οθόνες εμφανίζεται ως αναδιπλούμενο μενού):

1. **Home: Επιστροφή στο Dashboard.**
2. **Wardrobe:** Προβολή και διαχείριση ρούχων.
3. **Outfits: Προβολή αποθηκευμένων συνόλων, δημιουργία νέων και λήψη εξατομικευμένων προτάσεων.**
4. **Εικονίδιο προφίλ:** Πρόσβαση σε ρυθμίσεις (Settings) και αποσύνδεση (Log out).



Εικόνα 9.29. Γραμμή πλοήγησης

10. Οφέλη που Αναμένουμε από την Προτεινόμενη Λύση

Η λύση αυτή αναμένεται να προσφέρει τα εξής οφέλη:

- Λιγότερα ξεχασμένα ρούχα: οι υπενθυμίσεις βοηθούν τον χρήστη να θυμηθεί και να ξαναφορέσει ρούχα που έχει.
- Πιο γρήγορες αποφάσεις: οι προτάσεις ανά περίπτωση γλιτώνουν χρόνο κάθε μέρα.
- Καλύτερη αισθητική: ο έλεγχος χρωματικής αρμονίας αποτρέπει άσχημους συνδυασμούς.
- Εξατομίκευση χωρίς κόπο: το σύστημα μαθαίνει τις προτιμήσεις του χρήστη μόνο του, χωρίς ερωτηματολόγια.
- Πιο σωστές αγορές: οι προτάσεις αγοράς βοηθούν τον χρήστη να αγοράζει ρούχα που πραγματικά χρειάζεται.
- Εύκολη πρόσβαση: επειδή είναι διαδικτυακή εφαρμογή, δουλεύει σε οποιαδήποτε συσκευή, χωρίς εγκατάσταση και χωρίς κόστος.
- Διαφάνεια: επειδή οι προτάσεις βασίζονται σε κατανοητούς κανόνες, ο χρήστης καταλαβαίνει γιατί προτείνεται κάθε εμφάνιση — κάτι που δεν συμβαίνει σε εφαρμογές που χρησιμοποιούν «μαύρα κουτιά» τεχνητής νοημοσύνης.

11. Μελλοντικές Επεκτάσεις και Βελτιώσεις

Υπάρχουν τρεις βασικές κατευθύνσεις για το μέλλον του Armoire. Πρώτον, η αυτόματη αναγνώριση ρούχων από φωτογραφία: το σύστημα θα μπορεί να αναγνωρίζει αυτόματα τι είδος ρούχου βλέπει σε μια φωτογραφία και να βρίσκει παρόμοια ρούχα, μειώνοντας την ανάγκη ο χρήστης να βάζει ετικέτες με το χέρι. Δεύτερον, ένας πιο έξυπνος τρόπος βαθμολόγησης προτάσεων, που θα προσαρμόζεται καλύτερα στον κάθε χρήστη ξεχωριστά, αντί να χρησιμοποιεί την ίδια φόρμουλα για όλους. Τρίτον, ένα πιο πλήρες dashboard με γραφήματα για το πώς αλλάζουν οι συνήθειες του χρήστη με τον καιρό, καθώς και προτάσεις βασισμένες στις προτιμήσεις άλλων χρηστών με παρόμοιο στυλ.

12. Συμπεράσματα

Σκοπός αυτής της πτυχιακής εργασίας ήταν να σχεδιαστεί και να υλοποιηθεί το Armoire, μια διαδικτυακή εφαρμογή που κάνει τη διαχείριση γκαρνταρόμπας πιο εύκολη και πιο οργανωμένη. Η εργασία περιλάμβανε βιβλιογραφική έρευνα, ανάλυση απαιτήσεων, σχεδιασμό αρχιτεκτονικής και βάσης δεδομένων, υλοποίηση ενός αλγορίθμου πρότασης εμφανίσεων, και τεκμηρίωση με παραδείγματα χρήσης και εγχειρίδιο.

Οι στόχοι της εργασίας επιτεύχθηκαν σε μεγάλο βαθμό: η καταλογοποίηση ρούχων, η δημιουργία εμφανίσεων, οι προτάσεις ανά περίπτωση, η παρακολούθηση φοράς, οι υπενθυμίσεις, οι προτάσεις αγοράς και η διαχείριση λογαριασμού είναι όλα υλοποιημένα (Κεφάλαια 7–9). Το πιο σημαντικό στοιχείο της εργασίας είναι ότι το Armoire συνδυάζει, σε μία δωρεάν εφαρμογή, απλούς και κατανοητούς κανόνες με μάθηση από τις προτιμήσεις του χρήστη — κάτι που, όπως δείχθηκε στο Κεφάλαιο 5, δεν προσφέρει καμία από τις ανταγωνιστικές εφαρμογές που εξετάστηκαν.

Υπήρχαν όμως και κάποιοι περιορισμοί. Πρώτον, το σύστημα βασίζεται εξ ολοκλήρου στις ετικέτες που βάζει ο χρήστης· αν ένα ρούχο δεν επισημανθεί σωστά, δεν θα εμφανιστεί ποτέ στις προτάσεις για τη σχετική περίπτωση. Μια μελλοντική αυτόματη αναγνώριση εικόνας θα μπορούσε να λύσει αυτό το πρόβλημα. Δεύτερον, οι κανόνες χρωματικής αρμονίας είναι χτισμένοι με το χέρι και όχι μαθημένοι από δεδομένα· οι Li et al. [28] δείχνουν ότι η αντίληψη των ανθρώπων για τα χρώματα δεν είναι πάντα ίδια με την κλασική θεωρία, οπότε αυτοί οι κανόνες θα μπορούσαν στο μέλλον να βελτιωθούν. Τρίτον, δεν πραγματοποιήθηκε αξιολόγηση ευχρηστίας με χρήστες· μια τέτοια αξιολόγηση θα ήταν χρήσιμη πριν από ευρύτερη κυκλοφορία της εφαρμογής.

Οι μελλοντικές επεκτάσεις, όπως αναλύονται στο Κεφάλαιο 11, περιλαμβάνουν: αυτόματη αναγνώριση ρούχων από φωτογραφία, πιο έξυπνη βαθμολόγηση προτάσεων ανά χρήστη, και ένα πιο πλήρες dashboard με στατιστικά. Συνολικά, το Armoire προσφέρει ήδη μια ολοκληρωμένη και κατανοητή λύση στο πρόβλημα που περιγράφηκε στο Κεφάλαιο 4, με σαφή κατεύθυνση για το πώς μπορεί να βελτιωθεί στο μέλλον.

Πίνακας Ορολογίας

Ξενόγλωσσος Όρος	Ελληνικός Όρος
Wardrobe	Γκαρνταρόμπα
Outfit	Εμφάνιση / Σύνολο ρούχων
Recommender system	Σύστημα συστάσεων
Cosine similarity	Ομοιότητα συνημιτόνου
Jaccard similarity	Ομοιότητα Jaccard
Color harmony	Χρωματική αρμονία
Embedding	Διανυσματική αναπαράσταση
Token (JWT)	Διακριτικό πρόσβασης
Backend	Εξυπηρετητής / Σύστημα υποστήριξης
Frontend	Πελατευόμενο περιβάλλον / Διεπαφή χρήστη
Responsive design	Προσαρμοστικός σχεδιασμός
Empty state	Κατάσταση κενού περιεχομένου
Nudge	Υπενθύμιση (ώθηση)
Highlight	Επισήμανση
Badge	Σήμα / ένδειξη
Chips	Ετικέτες (chips)
Pipeline	Αλυσίδα επεξεργασίας
Cold start	Ψυχρή εκκίνηση
Feature vector	Διάνυσμα χαρακτηριστικών
Sparse vector	Αραιό διάνυσμα
Content-based filtering	Φιλτράρισμα βάσει περιεχομένου
Endpoint	Τελικό σημείο (API)
Hashing	Κατακερματισμός
Path traversal	Διάσχιση διαδρομών
Use case	Περίπτωση χρήσης
Sequence diagram	Διάγραμμα ακολουθίας
Heuristic evaluation	Ευρετική αξιολόγηση
Shelve	Αρχειοθέτηση
Dashboard	Πίνακας εποπτείας

Πίνακας Συντμήσεων – Αρκτικόλεξων – Ακρωνυμίων

Σύντμηση	Πλήρης Απόδοση
API	Application Programming Interface
REST	Representational State Transfer
JWT	JSON Web Token
SPA	Single Page Application
JPA	Jakarta Persistence API
SQL	Structured Query Language
CLIP	Contrastive Language-Image Pre-training
FAISS	Facebook AI Similarity Search
TF	Term Frequency
HCI	Human-Computer Interaction
JSON	JavaScript Object Notation
ERD	Entity-Relationship Diagram
AI	Artificial Intelligence
UI	User Interface
UX	User Experience
IoT	Internet of Things
RFID	Radio-Frequency Identification
GPT	Generative Pre-trained Transformer
TOPSIS	Technique for Order of Preference by Similarity to Ideal Solution
ITS	Intelligent Tutoring System
GDPR	General Data Protection Regulation
HTTP	HyperText Transfer Protocol
HTTPS	HyperText Transfer Protocol Secure
TLS	Transport Layer Security
SHA	Secure Hash Algorithm
HMAC	Hash-based Message Authentication Code
CSRF	Cross-Site Request Forgery
UUID	Universally Unique Identifier
JPEG	Joint Photographic Experts Group
PNG	Portable Network Graphics
3NF	Third Normal Form
PWA	Progressive Web Application
LSTM	Long Short-Term Memory

Βιβλιογραφία

Οι βιβλιογραφικές αναφορές παρατίθενται με αριθμούς εντός αγκύλης, με βάση τη σειρά εμφάνισής τους στο κείμενο, σύμφωνα με τον Κανονισμό Εκπόνησης Πτυχιικής Εργασίας του Τμήματος.

- [1] Land, R., Crnkovic, I., & Wallin, C. (2003). Software systems integration and architectural analysis — A case study. In Proceedings of the 19th IEEE International Conference on Software Maintenance (ICSM). IEEE.
- [2] Shibl, M. A., Helal, I. M. A., & Mazen, S. A. (2022). System integration for large-scale software projects: Models, approaches, and challenges. In Proceedings of the International Conference on Emerging Technologies and Intelligent Systems. Springer.
- [3] Gupta, S., Modgil, S., Bhushan, B., Sivarajah, U., & Banerjee, S. (2023). Design and implementation of an IIoT driven information system: A case study. *Information Systems Frontiers*, 27(2), 523–537.
- [4] Chukwumanya, O. E., Onwurah, U. O., & Ihueze, C. C. (2024). Development of a web-based inventory management system for small businesses. *International Journal of Industrial and Production Engineering*, 2(2), 1–12.
- [5] Tanaman, M. T., Lloyd, J., Baylosis, A., Abiles, B. J. A., Catungal, M. L. P., & Encarnacion, P. C. (2023). Web-based inventory management system. *International Journal of Science and Applied Information Technology*, 12(1), 1–8.
- [6] Desai, A. (2025). Enhancing inventory management with progressive web applications (PWAs): A scalable solution for small and large enterprises (arXiv:2506.11011) [Preprint]. arXiv.
- [7] Golmohammadi, A., Zhang, M., & Arcuri, A. (2023). Testing RESTful APIs: A survey. *ACM Transactions on Software Engineering and Methodology*, 33(1), Article 27.
- [8] Rodríguez, C., Baez, M., Daniel, F., Casati, F., Trabucco, J. C., Canali, L., & Percannella, G. (2016). REST APIs: A large-scale analysis of compliance with principles and best practices. In Proceedings of the International Conference on Web Engineering (ICWE) (pp. 21–39). Springer.
- [9] Gavrilă, V., Băjenaru, L., & Dobre, C. (2019). Modern single page application architecture: A case study. *Studies in Informatics and Control*, 28(2), 231–238.
- [10] Mukhiya, S. K., & Hung, H. K. (2019). An architectural style for single page scalable modern web application. *International Journal of Recent Research Aspects*, 6(2), 45–50.
- [11] Prayogi, A. A., Niswar, M., Indrabayu, & Rijal, M. (2020). Design and implementation of REST API for academic information system. *IOP Conference Series: Materials Science and Engineering*, 875, Article 012047.
- [12] Goh, K. N., Chen, Y. Y., & Lin, E. S. (2011). Developing a smart wardrobe system. In Proceedings of the IEEE Consumer Communications and Networking Conference (CCNC). IEEE.

- [13] Indrawan, M., Loke, S. W., Ling, S., & Samara, F. (2007). Unobtrusive user profiling: The use of RFID to create a smart wardrobe. In Proceedings of the 1st International Workshop on RFID Technology (RFID-TCAC 2007). SciTePress.
- [14] Prakasa, F. B. P., Maiga, J., & Suyoto. (2019). IoT-based smart and healthy wardrobe system. In Proceedings of the International Conference of Artificial Intelligence and Information Technology (ICAIIIT). IEEE.
- [15] Gaikwad, P. P., Gabhane, J. P., & Golait, S. S. (2015). A survey based on smart homes system using Internet-of-Things. In Proceedings of the International Conference on Computation of Power, Energy, Information and Communication (ICCPEIC) (pp. 330–335). IEEE.
- [16] Alfantoukh, R., Almutairi, H., Alsufaian, F., Alswed, M., & Alkaldi, W. (2025). Dithar: IoT-enabled smart wardrobe for real-time clothing management. In Proceedings of the 2nd International Conference on Artificial Intelligence, Digital Media Technology and Interaction Design (ICADI '25). ACM.
- [17] Cheng, W.-H., Song, S., Chen, C.-Y., Hidayati, S. C., & Liu, J. (2021). Fashion meets computer vision: A survey. *ACM Computing Surveys*, 54(4), 1–41.
- [18] Jiang, G., & Macintyre, L. (2025). Wardrobe management apps and their unintended benefits for fashion sustainability and well-being: Insights from user reviews. *Sustainability*, 17(9), 4159.
- [19] Whering the future: A case study exploration of how the wardrobe app is enabling sustainable fashion choices. *International Journal of Sustainable Fashion & Textiles*, 2026.
- [20] Singh, A., et al. (2024). A deep learning-powered wardrobe companion app for outfit recommendations. *AIP Conference Proceedings*, 3237(1), 020010.
- [21] Patel, R., et al. (2025). A user-centric approach to revolutionizing wardrobe management: Integrating connected sensors and AI-driven outfit suggestions. In *Lecture Notes in Networks and Systems*. Springer.
- [22] Chérie: A proposed design for a mobile application with AI outfit assistance and 3D virtual wardrobe. In Proceedings of the 2023 9th International Conference on Industrial and Business Engineering (ICIBE) (pp. 1–6). ACM.
- [23] Jiang, G., & Macintyre, L. (2024). Consumer perceptions of app functions designed to reduce unnecessary fashion purchases. In *Lecture Notes in Networks and Systems*. Springer.
- [24] Adomavicius, G., & Tuzhilin, A. (2005). Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering*, 17(6), 734–749.
- [25] Salton, G., & Buckley, C. (1988). Term-weighting approaches in automatic text retrieval. *Information Processing & Management*, 24(5), 513–523.
- [26] Wang, J., & Dong, Y. (2020). Measurement of text similarity: A survey. *Information*, 11(9), Article 421.
- [27] Bag, S., Kumar, S. K., & Tiwari, M. K. (2019). An efficient recommendation generation using relevant Jaccard similarity. *Information Sciences*, 483, 53–64.

- [28] Li, H. C., Wang, L. K., & Chang, Y. K. (2025). Establishing colour harmony evaluation and recommendation model for clothing colour matching based on machine learning and deep learning. *Fashion and Textiles*, 12, Article 27.
- [29] Han, X., Wu, Z., Jiang, Y.-G., & Davis, L. S. (2017). Learning fashion compatibility with bidirectional LSTMs. In *Proceedings of the 25th ACM International Conference on Multimedia (MM '17)* (pp. 1078–1086). ACM.
- [30] Vasileva, M. I., Plummer, B. A., Dusad, K., Rajpal, S., Kumar, R., & Forsyth, D. (2018). Learning type-aware embeddings for fashion compatibility. In *Computer Vision – ECCV 2018* (pp. 390–405). Springer.
- [31] Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., Krueger, G., & Sutskever, I. (2021). Learning transferable visual models from natural language supervision. In *Proceedings of the 38th International Conference on Machine Learning (ICML)*, PMLR 139, 8748–8763.
- [32] Johnson, J., Douze, M., & Jégou, H. (2021). Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*, 7(3), 535–547.
- [33] Mohammadi, S. O., & Kalhor, A. (2021). Smart fashion: A review of AI applications in the fashion & apparel industry (arXiv:2111.00905) [Preprint]. arXiv.
- [34] Ding, Y., Lai, Z., Mok, P. Y., & Chua, T.-S. (2023). Computational technologies for fashion recommendation: A survey. *ACM Computing Surveys*, 56(5), 1–45.
- [35] Chrysafiadi, K. (2024, August). Incorporating Fuzzy Logic in Intelligent Software Systems. In *International Conference on Artificial Intelligence-empowered Software Engineering* (pp. 470–475). Cham: Springer Nature Switzerland.
- [36] Chrysafiadi, K., Tsichrintzi, E. A., & Sakkopoulos, E. (2024). A smart home fuzzy logic-based cooking fire prevention system addressing human errors due to distraction or minor medical memory lapses. *Procedia Computer Science*, 246, 3390–3399.
- [37] Panagoulas, D. P., Sotiropoulos, D., Chrysafiadi, K., Tsichrintzi, E. A., Sakkopoulos, E., Tsihrintzis, G. A., & Virvou, M. (2024, July). LYRICEL: a rule-augmented artificial intelligence-empowered cultural E-learning with GPT and machine learning for song lyrics. In *2024 15th International Conference on Information, Intelligence, Systems & Applications (IISA)* (pp. 1–5). IEEE.
- [38] Vali-Sarafoglou, E., & Chrysafiadi, K. (2024, July). TopMoviePicks: a personalized movie recommendation system based on TOPSIS. In *2024 15th International Conference on Information, Intelligence, Systems & Applications (IISA)* (pp. 1–8). IEEE.
- [39] Chrysafiadi, K., Kontogianni, A., Virvou, M., & Alepis, E. (2025). Enhancing user experience in smart tourism via fuzzy logic-based personalization. *Mathematics*, 13(5), 846.
- [40] Kontogianni, A., Chrysafiadi, K., Virvou, M., & Alepis, E. (2026). From Wikidata to Smart Tourism: A Reproducible Pipeline Based on AI and Fuzzy Logic for Interpretable Multi-Category Classification of Points of Interest. *Mathematics*, 14(12), 2227.

- [41] Kaplan, K. (2021). Designing empty states in complex applications: 3 guidelines. Nielsen Norman Group. <https://www.nngroup.com/articles/empty-state-interface-design/>
- [42] Lindgaard, G., Fernandes, G., Dudek, C., & Brown, J. (2006). Attention web designers: You have 50 milliseconds to make a good first impression! *Behaviour & Information Technology*, 25(2), 115–126.
- [43] Nakamura, W. T., Ahmed, I., Redmiles, D., Oliveira, E., Fernandes, D., de Oliveira, E. H. T., & Conte, T. (2021). Are UX evaluation methods providing the same big picture? *Sensors*, 21(10), Article 3480.
- [44] Perrig, S. A. C., Aeschbach, L. F., Scharowski, N., von Felten, N., Opwis, K., & Brühlmann, F. (2024). Measurement practices in user experience (UX) research: A systematic quantitative literature review. *Frontiers in Computer Science*, 6, Article 1368860.
- [45] Sulikowski, P., Kucznerowicz, M., Bąk, I., Romanowski, A., & Zdziebko, T. (2022). Online store aesthetics impact efficacy of product recommendations and highlighting. *Sensors*, 22(23), Article 9186.
- [46] Aayesha, Afzaal, M., & Neidhardt, J. (2024). User experience of recommender system: A user study of social-aware fashion recommendations system. In *Adjunct Proceedings of the 32nd ACM Conference on User Modeling, Adaptation and Personalization (UMAP Adjunct '24)*. ACM.
- [47] Brusilovsky, P. (2001). Adaptive hypermedia. *User Modeling and User-Adapted Interaction*, 11(1–2), 87–110.
- [48] Miraz, M. H., Ali, M., & Excell, P. S. (2021). Adaptive user interfaces and universal usability through plasticity of user interface design. *Computer Science Review*, 40, Article 100363.
- [49] Purificato, E., Boratto, L., & De Luca, E. W. (2024). User modeling and user profiling: A comprehensive survey (arXiv:2402.09660) [Preprint]. arXiv.
- [50] Ricci, F., Rokach, L., & Shapira, B. (Eds.). (2022). *Recommender systems handbook* (3rd ed.). Springer.
- [51] Sayed, W. S., Noeman, A. M., Abdellatif, A., Abdelrazek, M., Badawy, M. G., Hamed, A., & El-Tantawy, S. (2022). AI-based adaptive personalized content presentation and exercises navigation for an effective and engaging e-learning platform. *Multimedia Tools and Applications*, 82, 3303–3333.
- [52] Chrysafiadi, K., Virvou, M., Tsihrintzis, G. A., & Hatzilygeroudis, I. (2023). Evaluating the user's experience, adaptivity and learning outcomes of a fuzzy-based intelligent tutoring system for computer programming for academic students in Greece. *Education and Information Technologies*, 28(6), 6453–6483.
- [53] Chrysafiadi, K., Papadimitriou, S., Kamitsios, M., & Virvou, M. (2024, August). User Experience in Serious Games: Adaptivity in the Plot and Environment of the Game. In *International Conference on Artificial Intelligence-empowered Software Engineering* (pp. 215–230). Cham: Springer Nature Switzerland.
- [54] Papadimitriou, S., Chrysafiadi, K., & Virvou, M. (2023, July). Adaptive quizzes using fuzzy genetic algorithm. In *2023 14th International Conference on Information, Intelligence, Systems & Applications (IISA)* (pp. 1–8). IEEE.

- [55] Chrysafiadi, K., Papadimitriou, S., & Virvou, M. (2022). Cognitive-based adaptive scenarios in educational games using fuzzy reasoning. *Knowledge-Based Systems*, 250, 109111.
- [56] Chrysafiadi, K., Virvou, M., & Tsihrintzis, G. A. (2022). A fuzzy-based mechanism for automatic personalized assessment in an e-learning system for computer programming. *Intelligent Decision Technologies*, 16(4), 699–714.
- [57] Chrysafiadi, K., Virvou, M., Tsihrintzis, G. A., & Hatzilygeroudis, I. (2023). An adaptive learning environment for programming based on fuzzy logic and machine learning. *International Journal on Artificial Intelligence Tools*, 32(05), 2360011.
- [58] Tibensky, P., & Kompan, M. (2024). Context-aware adaptive personalised recommendation: A meta-hybrid (arXiv:2410.13374) [Preprint]. arXiv.
- [59] Virvou, M. (2023). Artificial Intelligence and User Experience in reciprocity: Contributions and state of the art. *Intelligent Decision Technologies*, 17(1), 73–125.
- [60] Chrysafiadi, K., & Virvou, M. (2021, July). Evaluating the user experience of a fuzzy-based Intelligent Tutoring System. In 2021 12th International Conference On Information, Intelligence, Systems & Applications (IISA) (pp. 1–7). IEEE.
- [61] Mouggiakou, E., Papadimitriou, S., Chrysafiadi, K., & Virvou, M. (2025). Artificial intelligence in educational games and consent under general data protection regulation. *Intelligent Decision Technologies*, 19(2), 670–686.
- [62] Nielsen, J., & Landauer, T. K. (1993). A mathematical model of the finding of usability problems. In *Proceedings of INTERACT '93 and CHI '93* (pp. 206–213). ACM.
- [63] Nielsen, J. (1994). Enhancing the explanatory power of usability heuristics. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '94)* (pp. 152–158). ACM.
- [64] MacKenzie, I. S. (1992). Fitts' law as a research and design tool in human-computer interaction. *Human-Computer Interaction*, 7(1), 91–139.
- [65] Khajouei, R., Zahiri Esfahani, M., & Jahani, Y. (2017). Comparison of heuristic and cognitive walkthrough usability evaluation methods for evaluating health information systems. *Journal of the American Medical Informatics Association*, 24(e1), e55–e60.

Παραρτήματα

Το Παράρτημα Α αναπαράγει τα scripts μετάπτωσης της βάσης δεδομένων (SQL) που αναφέρονται στο Κεφάλαιο 7. Το Παράρτημα Β αναπαράγει την αναφορά REST API του Κεφαλαίου 7 σε έναν ενοποιημένο πίνακα για γρήγορη αναφορά.

Παράρτημα Α — Scripts Μετάπτωσης Βάσης Δεδομένων (SQL)

001_users.sql

```
CREATE TABLE IF NOT EXISTS users (  
    id BIGINT AUTO_INCREMENT PRIMARY KEY,  
    email VARCHAR(255) NOT NULL,  
    password_hash VARCHAR(255) NOT NULL,  
    first_name VARCHAR(100) NOT NULL,  
    last_name VARCHAR(100) NOT NULL,  
    created_at TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,  
    updated_at TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE  
CURRENT_TIMESTAMP,  
    CONSTRAINT uk_users_email UNIQUE (email)  
);
```

002_clothing_items.sql

```
CREATE TABLE IF NOT EXISTS clothing_items (  
    id BIGINT AUTO_INCREMENT PRIMARY KEY,  
    user_id BIGINT NOT NULL,  
    name VARCHAR(255) NOT NULL,  
    category VARCHAR(50) NOT NULL,  
    subcategory VARCHAR(50) NULL,  
    primary_color VARCHAR(50) NOT NULL,  
    secondary_color VARCHAR(50) NULL,  
    image_path VARCHAR(500) NOT NULL,  
    status VARCHAR(20) NOT NULL DEFAULT 'ACTIVE',  
    wear_count INT NOT NULL DEFAULT 0,  
    last_worn_at TIMESTAMP NULL,  
    notes VARCHAR(1000) NULL,  
    created_at TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,  
    updated_at TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE  
CURRENT_TIMESTAMP,  
    CONSTRAINT fk_clothing_items_user FOREIGN KEY (user_id) REFERENCES users (id),  
    INDEX idx_clothing_items_user (user_id),  
    INDEX idx_clothing_items_category (category),  
    INDEX idx_clothing_items_status (status)  
);  
  
CREATE TABLE IF NOT EXISTS clothing_item_seasons (  
    clothing_item_id BIGINT NOT NULL,  
    season VARCHAR(20) NOT NULL,  
    CONSTRAINT fk_clothing_item_seasons_item FOREIGN KEY (clothing_item_id) REFERENCES  
clothing_items (id) ON DELETE CASCADE,  
    PRIMARY KEY (clothing_item_id, season)  
);  
  
CREATE TABLE IF NOT EXISTS clothing_item_occasions (  
    clothing_item_id BIGINT NOT NULL,  
    occasion VARCHAR(30) NOT NULL,  
    CONSTRAINT fk_clothing_item_occasions_item FOREIGN KEY (clothing_item_id) REFERENCES  
clothing_items (id) ON DELETE CASCADE,  
    PRIMARY KEY (clothing_item_id, occasion)  
);
```

003_outfits.sql

```
CREATE TABLE IF NOT EXISTS outfits (  
    id BIGINT AUTO_INCREMENT PRIMARY KEY,  
    user_id BIGINT NOT NULL,  
    name VARCHAR(255) NOT NULL,  
    occasion VARCHAR(30) NOT NULL,  
    notes VARCHAR(1000) NULL,  
    created_at TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,  
    updated_at TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE  
CURRENT_TIMESTAMP,  
    CONSTRAINT fk_outfits_user FOREIGN KEY (user_id) REFERENCES users (id),  
    INDEX idx_outfits_user (user_id),  
    INDEX idx_outfits_occasion (occasion)  
);  
  
CREATE TABLE IF NOT EXISTS outfit_items (  

```

```
    outfit_id          BIGINT NOT NULL,
    clothing_item_id   BIGINT NOT NULL,
    sort_order         INT NOT NULL,
    CONSTRAINT fk_outfit_items_outfit FOREIGN KEY (outfit_id) REFERENCES outfits (id) ON
DELETE CASCADE,
    CONSTRAINT fk_outfit_items_clothing FOREIGN KEY (clothing_item_id) REFERENCES
clothing_items (id),
    PRIMARY KEY (outfit_id, clothing_item_id)
);
```

004_outfit_wear_tracking.sql

```
ALTER TABLE outfits
    ADD COLUMN wear_count    INT NOT NULL DEFAULT 0,
    ADD COLUMN last_worn_at  TIMESTAMP NULL;
```

Παράρτημα Β — Ενοποιημένη Αναφορά REST API

Μέθοδος	Διαδρομή	Πιστοποίηση	Περιγραφή
GET	/api/v1/info	Όχι	Όνομα και έκδοση εφαρμογής
POST	/api/v1/auth/signup	Όχι	Δημιουργία λογαριασμού
POST	/api/v1/auth/login	Όχι	Σύνδεση· επιστρέφει bearer JWT + χρήστη
GET	/api/v1/auth/user/{userId}	Ναι	Προφίλ χρήστη (μόνο του ίδιου χρήστη)
PATCH	/api/v1/auth/profile	Ναι	Ενημέρωση ονόματος/επωνύμου
PUT	/api/v1/auth/password	Ναι	Αλλαγή κωδικού
DELETE	/api/v1/auth/account	Ναι	Διαγραφή λογαριασμού και όλων των δεδομένων
GET	/api/v1/wardrobe/items	Ναι	Λίστα ρούχων
GET	/api/v1/wardrobe/items/{id}	Ναι	Ανάκτηση στοιχείων ρούχου
POST	/api/v1/wardrobe/items	Ναι	Ανέβασμα ρούχου
PUT	/api/v1/wardrobe/items/{id}	Ναι	Επεξεργασία ρούχου (στοιχεία και προαιρετικά νέα φωτογραφία)
GET	/api/v1/wardrobe/items/{id}/image	Ναι	Ανάκτηση φωτογραφίας ρούχου
DELETE	/api/v1/wardrobe/items/{id}	Ναι	Διαγραφή ρούχου
POST	/api/v1/wardrobe/items/{id}/shelve	Ναι	Αρχειοθέτηση αχρησιμοποίητου ρούχου
GET	/api/v1/wardrobe/nudges	Ναι	Υπενθυμίσεις αφόρετων ρούχων
GET	/api/v1/outfits	Ναι	Λίστα αποθηκευμένων εμφανίσεων
GET	/api/v1/outfits/{id}	Ναι	Ανάκτηση αποθηκευμένης εμφάνισης
POST	/api/v1/outfits	Ναι	Δημιουργία εμφάνισης από αναγνωριστικά ρούχων
GET	/api/v1/outfits/suggestions?occasion=	Ναι	Λήψη προτάσεων ή κενής κατάστασης
POST	/api/v1/outfits/{id}/wear	Ναι	Καταγραφή ότι η εμφάνιση φορέθηκε σήμερα
DELETE	/api/v1/outfits/{id}	Ναι	Διαγραφή εμφάνισης
GET	/actuator/health	Όχι	Έλεγχος υγείας συστήματος
GET	/actuator/info	Όχι	Πληροφορίες εφαρμογής (Actuator)