



ΠΑΝΕΠΙΣΤΗΜΙΟ
ΠΑΤΡΩΝ
UNIVERSITY OF PATRAS

ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ & ΤΕΧΝΟΛΟΓΙΑΣ ΥΠΟΛΟΓΙΣΤΩΝ

Σχεδίαση και ανάπτυξη μίας εφαρμογής
προγραμματισμού βάρδιας για μεγάλες
επιχειρήσεις

Design and development of a shift scheduling
application for large enterprises

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Χρήστος Κατσανδρής
Αριθμός Μητρώου: 1072755

Επιβλέπων
Χρήστος Σιντόρης, Ε.ΔΙ.Π.

Πάτρα
Ιούλιος 2024

Πανεπιστήμιο Πατρών, Τμήμα Ηλεκτρολόγων Μηχανικών και Τεχνολογίας Υπολογιστών.

©2024 —Με την επιφύλαξη παντός δικαιώματος

Το σύνολο της εργασίας αποτελεί πρωτότυπο έργο, παραχθέν από τον Χρήστο Κατσανδρή, και δεν παραβιάζει δικαιώματα τρίτων καθ' οιονδήποτε τρόπο. Αν η εργασία περιέχει υλικό, το οποίο δεν έχει παραχθεί από τον ίδιο, αυτό είναι ευδιάκριτο και αναφέρεται ρητώς εντός του κειμένου της εργασίας ως προϊόν εργασίας τρίτου, σημειώνοντας με παρομοίως σαφή τρόπο τα στοιχεία ταυτοποίησής του, ενώ παράλληλα βεβαιώνει πως στην περίπτωση χρήσης αυτούσιων γραφικών αναπαραστάσεων, εικόνων, γραφημάτων κ.λπ., έχει λάβει τη χωρίς περιορισμούς άδεια του κατόχου των πνευματικών δικαιωμάτων για την συμπερίληψη και επακόλουθη δημοσίευση του υλικού αυτού.



Χρήστος Κατσανδρής

ΠΙΣΤΟΠΟΙΗΣΗ

Πιστοποιείται ότι η διπλωματική εργασία με θέμα
Σχεδίαση και ανάπτυξη μίας εφαρμογής προγραμματισμού βάρδιας
για μεγάλες επιχειρήσεις
του φοιτητή του τμήματος Ηλεκτρολόγων Μηχανικών & Τεχνολογίας
Υπολογιστών

Χρήστον Κατσανδρή του Δημητρίου

Αριθμός Μητρώου: 1072755

παρουσιάστηκε δημόσια στο τμήμα Ηλεκτρολόγων Μηχανικών &
Τεχνολογίας Υπολογιστών στις

10/7/2024

και εξετάστηκε από την ακόλουθη εξεταστική επιτροπή:

Χρήστος Σιντόρης, Ε.ΔΙ.Π, ΤΗΜ&ΤΥ (επιβλέπων)

Σοφία Δασκαλάκη, Επίκουρη Καθηγήτρια, ΤΗΜ&ΤΥ (μέλος επιτροπής)

Χρήστος Φείδας, Αναπληρωτής Καθηγητής, ΤΗΜ&ΤΥ (μέλος επιτροπής)

Ο Επιβλέπων

Ο Διευθυντής του Τομέα
Ηλεκτρονικής και Υπολογιστών

Χρήστος Σιντόρης
Ε.ΔΙ.Π

Γεώργιος Θεοδωρίδης
Αναπληρωτής Καθηγητής

Σύνοψη

Στην παρούσα διπλωματική εργασία, πραγματοποιείται η πλήρης ανάπτυξη μίας εφαρμογής, με σκοπό τον χρονοπρογραμματισμό της εργασίας σε μεγάλες επιχειρήσεις. Στα πλαίσια της εργασίας αυτής, γίνεται ειδική μνεία στην επιστήμη της Αλληλεπίδρασης Ανθρώπου - Υπολογιστή και, ιδιαίτερα, στους κανόνες που διέπουν τον ανθρωποκεντρικό σχεδιασμό μίας εφαρμογής. Επιπλέον, εφαρμόζεται η τεχνική του Γραμμικού Προγραμματισμού, για την κατάρτιση του αλγορίθμου χρονοπρογραμματισμού εργασίας.

Η έννοια της πλήρους ανάπτυξης της εφαρμογής περιλαμβάνει τόσο την ανάλυση απαιτήσεων και τη σχεδίαση της γραφικής διεπαφής, όσο και την υλοποίησή της σε μία μορφή λειτουργικού προϊόντος, το οποίο θέτει σε προτεραιότητα την καλή εμπειρία χρήστη και την ευχρηστία.

Η καλή εμπειρία χρήστης της εφαρμογής τεκμηριώνεται με μεθόδους αξιολόγησης του ανθρωποκεντρικού σχεδιασμού καθ' όλη τη διάρκεια της ανάπτυξής της. Με την ολοκλήρωση της εργασίας, προκύπτει ένα προϊόν που καθιστά την καθημερινότητα των εργαζομένων μίας μεγάλης επιχείρησης ευκολότερη, αυτοματοποιώντας τις διαδικασίες που αφορούν τον προγραμματισμό της εργασίας, σε έναν κόσμο που μετασχηματίζεται ολοένα και ταχύτερα στην ψηφιακή εποχή.

Λέξεις-κλειδιά: Χρονοπρογραμματισμός βαρδιών, Γραμμικός προγραμματισμός, Αλληλεπίδραση Ανθρώπου - Υπολογιστή, Ανθρωποκεντρικός σχεδιασμός, Εφαρμογές cross-platform, Flutter, Firebase

Abstract

In this diploma thesis, a full-stack application is being designed and developed, with the aim of supporting the work scheduling of large enterprises. In this context, a special mention is made in the science of Human - Computer Interaction and, more specifically, in the rules that govern the human-centered design of an application. Furthermore, it is the Linear Programming technique that is being employed, so as to develop the algorithm of the automatic shift scheduling.

The concept of the full-stack application development includes not only the requirement analysis and the user interface design, but also its implementation to a form of an operating product, that prioritizes the best user experience and usability.

The user experience of the application is documented by evaluation methods of the human-centered design framework, throughout the period of its design. Completing this thesis, results in a product that makes the daily life of a large enterprise's employees easier, by automating the work scheduling processes, in a world that is transforming ever faster to the digital era.

Keywords: Shift scheduling, Linear programming, Human - Computer Interaction, Human-centered design, Cross-platform applications, Flutter, Firebase

Ευχαριστίες

Με την ολοκλήρωση της διπλωματικής μου εργασίας, θα ήθελα να ευχαριστήσω θερμά όλους, όσοι με βοήθησαν, με καθοδήγησαν και με στήριξαν καθ' όλη τη διάρκεια της ακαδημαϊκής μου πορείας μέχρι σήμερα.

Πρώτον, θα ήθελα να ευχαριστήσω τον κο Χρήστο Σιντόρη για τη συνεχή του βοήθεια και ενασχόληση κατά την εκπόνηση της εργασίας μου. Επίσης, θα ήθελα να ευχαριστήσω τον κο Νικόλαο Αβούρη για το εξαιρετικό του έργο, μιας και τα μαθήματά του ήταν πηγή έμπνευσης για εμένα από το πρώτο κιόλας έτος. Ευχαριστώ την κα Σοφία Δασκαλάκη και τον κο Χρήστο Βαλουζή, για τη βοήθειά τους σε θέματα Γραμμικής Βελτιστοποίησης.

Ακόμη, θα ήθελα να ευχαριστήσω τους φίλους και συμφοιτητές μου για τη συνεχή συμπαράσταση. Ιδιαίτερα, ευχαριστώ τον φίλο και εξαιρετικό, πλέον, συνάδελφο, Γεώργιο Τσιάλιο, για την υπέροχη συνεργασία αυτά τα χρόνια, που μου έμαθε να αγαπώ την ομαδικότητα.

Ξεχωριστές ευχαριστίες αξίζει η οικογένειά μου για τη βοήθεια και τη στήριξη που μου προσφέρουν συνεχώς. Ευχαριστώ ιδιαίτερα τη μητέρα μου, Χρύσα, και όλο το προσωπικό του [The Y Hotel](#) για την έμπνευση που μου προσέφεραν επί του θέματος της εργασίας μου.

Περιεχόμενα

1 Εισαγωγή	1
2 Βιβλιογραφική επισκόπηση	3
2.1 Ανάπτυξη cross-platform εφαρμογών	3
2.1.1 Συσκευές, λειτουργικά συστήματα και native apps	3
2.1.2 Εξέλιξη των cross-platform εφαρμογών	6
2.1.3 Flutter: Περιγραφή και χαρακτηριστικά	8
2.2 Ανθρωποκεντρικός σχεδιασμός εφαρμογών	10
2.2.1 Μοντέλα ανάπτυξης λογισμικού	11
2.2.2 Ευχρηστία συστήματος	12
2.2.3 Κανόνες μέτρησης ευχρηστίας	15
2.3 Το πρόβλημα του Βέλτιστου Χρονοπρογραμματισμού Εργασίας	16
2.3.1 Περιγραφή του προβλήματος	16
2.3.2 Θεωρητική μελέτη	17
2.3.3 Εισαγωγή στον γραμμικό προγραμματισμό	18
2.3.4 Σχηματισμός του μοντέλου	19
3 Σχεδίαση της εφαρμογής	25
3.1 Ανάλυση προβλήματος και καθορισμός απαιτήσεων	25
3.1.1 Ανάλυση χρηστών	25
3.1.2 Το πλαίσιο PACT	26
3.1.3 Περσόνες και σενάρια χρήσης	28
3.1.4 Ιστορίες χρηστών	33
3.1.5 Πίνακες απαιτήσεων	34
3.1.6 Ανάλυση εργασιών: Εθνογραφία και Ιεραρχική ανάλυση εργασιών	37
3.2 Μελέτη ανταγωνιστικών λύσεων	40
3.2.1 When I Work	40
3.2.2 Homebase	42
3.2.3 Planday	43
3.2.4 7shifts	43

3.3	Σχεδιαστικές επιλογές	45
3.3.1	Επωνυμία και λογότυπο	45
3.3.2	Χρωματική παλέτα	46
3.3.3	Εικονίδια και γραφικά	47
3.3.4	Γραμματοσειρές	50
3.4	Σχεδίαση σκαριφημάτων και πρωτοτύπων	50
3.4.1	Αρχική οθόνη (Home page)	51
3.4.2	Οθόνη προγράμματος: Επιλογή προβολής (Schedule page: View selection) . .	51
3.4.3	Οθόνη προγράμματος: Ατομική προβολή (Schedule page: Personal view) . .	52
3.4.4	Οθόνη προγράμματος: Ομαδική προβολή (Schedule page: Team view) . . .	53
3.4.5	Φύλλο βάρδιας (Shift sheet)	55
3.4.6	Οδηγός δημιουργίας / επεξεργασίας προγράμματος (Create / Edit schedule wizard)	56
3.4.7	Οθόνη αιτημάτων και Φύλλο αιτήματος (Requests page και Request sheet) . .	60
3.4.8	Οδηγός νέου αιτήματος (New request wizard)	63
3.4.9	Οθόνη ρυθμίσεων (Settings page)	67
3.4.10	Οθόνη τοποθεσιών (Locations page)	67
3.4.11	Οθόνη ομάδων (Teams page)	70
3.4.12	Οθόνη εργαζομένων (Employees page)	70
3.4.13	Οθόνη συμμετοχής εργαζομένου σε ομάδες (Employee team membership page)	74
3.4.14	Οθόνη περιορισμών βαρδιών εργαζομένου (Shift constraints page)	74
3.4.15	Οθόνη προτιμήσεων βαρδιών εργαζομένου (Shift preferences page)	74
3.4.16	Οθόνη προτιμήσεων συναδέλφων εργαζομένου (Colleague preferences page) .	76
3.5	Σύνοψη	76
4	Υλοποίηση της εφαρμογής	79
4.1	Αρχιτεκτονική συστήματος	79
4.2	Η αρχιτεκτονική Model-View-ViewModel (MVVM)	79
4.3	Μοντέλο δεδομένων	82
4.4	Υλοποίηση της βάσης δεδομένων	84
4.5	Τοπική αποθήκευση δεδομένων και μεγάλα δυαδικά αρχεία	86
4.6	Αυθεντικοποίηση χρηστών	86
4.7	Μηχανή αυτόματης ανάθεσης βαρδιών	89
4.7.1	Υλοποίηση αλγορίθμου	89
4.7.2	Υλοποίηση εξυπηρετητή	90
4.8	Σύνοψη	91

5 Αξιολόγηση της εφαρμογής	93
5.1 Περί αξιολόγησης ευχρηστίας στον ανθρωποκεντρικό σχεδιασμό	93
5.2 Μέθοδοι που χρησιμοποιήθηκαν	93
5.3 Γνωσιακό περιδιάβασμα (Cognitive walkthrough)	94
5.4 Τα μοντέλα KLM - FLM	98
Δημιουργία προγράμματος εργασίας	100
Υποβολή αιτήματος άδειας	102
Υποβολή αιτήματος ανταλλαγής βαρδιών	103
5.5 Συνεντεύξεις με πραγματικούς χρήστες	103
5.6 Αξιολόγηση αποδοτικότητας αλγορίθμου αυτόματης ανάθεσης βαρδιών	105
6 Συμπεράσματα	109
6.1 Σύνοψη εργασίας	109
6.2 Μελλοντικά βήματα	110
7 Βιβλιογραφία	113
Παράρτημα	119
Πρωτότυπα που επανασχεδιάστηκαν μετά την αξιολόγηση	119
Σενάριο αξιολόγησης εμπειρίας χρήστη με πραγματικούς χρήστες	121

Κατάλογος πινάκων

3.1	Πίνακας λειτουργικών απαιτήσεων	35
3.2	Πίνακας μη λειτουργικών απαιτήσεων	37
5.1	Αποτελέσματα γνωσιακού περιδιαβάσματος για τη δημιουργία προγράμματος εργασίας	95
5.2	Αποτελέσματα γνωσιακού περιδιαβάσματος για τη δημιουργία αιτήματος από εργαζόμενο	97
5.3	Κατηγορίες ενεργειών του μοντέλου KLM	99
5.4	Κατηγορίες ενεργειών του μοντέλου FLM	100
5.5	Αποτελέσματα χρονομέτρησης αλγορίθμου αυτόματης ανάθεσης βαρδιών	105
5.6	Αποτελέσματα αναθέσεων συγκεκριμένου προβλήματος	107
5.7	Αποτελέσματα αναθέσεων συγκεκριμένου προβλήματος με προτιμήσεις συναδέλφων	107

Κατάλογος σχημάτων

2.1 Μερίδιο αγοράς ηλεκτρονικών συσκευών. Δεδομένα Ιανουαρίου 2022 (Kemp 2022)	4
2.2 Μερίδιο αγοράς λειτουργικών συστημάτων. Δεδομένα Νοεμβρίου 2023 (StatCounter GlobalStats 2023)	5
2.3 Αποτελέσματα έρευνας για την ανάπτυξη της δημοτικότητας του Flutter. Δεδομένα πρώτου τριμήνου 2023 (J. Lee 2023)	10
2.4 Το μοντέλο καταρράκτη	11
2.5 Το ελικοειδές μοντέλο	12
2.6 Το αστεροειδές μοντέλο	13
2.7 UX honeycomb	15
3.1 Σχηματική αναπαράσταση των ενδιαφερομένων της εφαρμογής Horario	26
3.2 Σχηματική αναπαράσταση του πλαισίου PACT για την εφαρμογή Horario	28
3.3 Το εικονικό ξενοδοχείο Starlight Hotel	29
3.4 Περσόνα 1: Μαργαρίτα Σταθούλια	30
3.5 Περσόνα 2: Ιωσηφίνα Νικοπολίδη	31
3.6 Περσόνα 3: Αγγελική Ηλιοπούλου	32
3.7 Περσόνα 4: Ηλίας Παλαιολόγος	33
3.8 Ιεραρχική ανάλυση εργασιών για τη δημιουργία προγράμματος εργασίας	39
3.9 Ιεραρχική ανάλυση εργασιών για την προβολή επερχόμενων βαρδιών	40
3.10 Ιεραρχική ανάλυση εργασιών για τη δημιουργία αιτήματος από εργαζόμενο	41
3.11 Η online έκδοση της εφαρμογής When I Work (πηγή: wheniwork.com)	42
3.12 Η εφαρμογή Homebase (πηγή: joinhomebase.com)	43
3.13 Η εφαρμογή Planday (πηγή: planday.com)	44
3.14 Η εφαρμογή 7shifts (πηγή: 7shifts.com)	44
3.15 Μία ενδεικτική εφαρμογή που ακολουθεί τις προδιαγραφές του Material 3 (πηγή: m3.material.io)	45
3.16 Το λογότυπο της επωνυμίας Horario	46
3.17 Το μονόγραμμα της επωνυμίας Horario	46
3.18 Η χρωματική παλέτα της εφαρμογής για ανοιχτόχρωμο θέμα (light theme)	48
3.19 Η χρωματική παλέτα της εφαρμογής για σκουρόχρωμο θέμα (dark theme)	49
3.20 Η Αρχική οθόνη: σκαρίφημα και πρωτότυπα	52

3.21	Η οθόνη επιλογής προβολής για την οθόνη Προγράμματος: σκαρίφημα και πρωτότυπα	53
3.22	Η Ατομική προβολή της οθόνης Προγράμματος: σκαρίφημα και πρωτότυπα	54
3.23	Η Ομαδική προβολή της οθόνης Προγράμματος: σκαρίφημα και πρωτότυπα	54
3.24	Το Φύλλο βάρδιας: σκαρίφημα και πρωτότυπα	55
3.25	Οδηγός προγράμματος: Βήμα 1: σκαρίφημα και πρωτότυπα	57
3.26	Οδηγός προγράμματος: Βήμα 2 (δημιουργία νέου): σκαρίφημα και πρωτότυπα	57
3.27	Οδηγός προγράμματος: Βήμα 2 (επεξεργασία υπάρχοντος): πρωτότυπα	58
3.28	Οδηγός προγράμματος: Βήμα 3: σκαρίφημα και πρωτότυπα	59
3.29	Οδηγός προγράμματος: Βήμα 3 (Φύλλο βάρδιας): σκαρίφημα και πρωτότυπα	59
3.30	Οδηγός προγράμματος: Βήμα 3 (Επιβεβαίωση αυτόματης ανάθεσης): σκαρίφημα και πρωτότυπα	60
3.31	Οδηγός προγράμματος: Βήμα 3 (Επιβεβαίωση δημοσίευσης): σκαρίφημα και πρωτότυπα	61
3.32	Οδηγός προγράμματος: Ολοκλήρωση: σκαρίφημα και πρωτότυπα	61
3.33	Η Οθόνη αιτημάτων: σκαρίφημα και πρωτότυπα	62
3.34	Η Οθόνη αιτημάτων για αιτήματα ομάδας: σκαρίφημα και πρωτότυπα	62
3.35	Το Φύλλο αιτήματος: σκαρίφημα και πρωτότυπα	63
3.36	Το Φύλλο αιτήματος για αίτημα προς έγκριση: σκαρίφημα και πρωτότυπα	64
3.37	Οδηγός αιτήματος: Βήμα 1: σκαρίφημα και πρωτότυπα	64
3.38	Οδηγός αιτήματος: Βήμα 2 (αίτημα άδειας): σκαρίφημα και πρωτότυπα	65
3.39	Οδηγός αιτήματος: Βήματα 2-3 (αίτημα ανταλλαγής βαρδιών): πρωτότυπα	66
3.40	Οδηγός αιτήματος: Βήμα 3/4: πρωτότυπα	66
3.41	Οδηγός αιτήματος: Ολοκλήρωση: σκαρίφημα και πρωτότυπα	67
3.42	Η Οθόνη ρυθμίσεων: σκαρίφημα και πρωτότυπα	68
3.43	Η Οθόνη τοποθεσιών: πρωτότυπα	69
3.44	Η Οθόνη επεξεργασίας τοποθεσίας: πρωτότυπα	69
3.45	Η Οθόνη ομάδων: πρωτότυπα	70
3.46	Η Οθόνη επεξεργασίας ομάδας: πρωτότυπα	71
3.47	Το Φύλλο τύπου βαρδιών: πρωτότυπα	72
3.48	Η Οθόνη εργαζομένων: πρωτότυπα	72
3.49	Η Οθόνη επεξεργασίας εργαζομένου: πρωτότυπα	73
3.50	Η Οθόνη συμμετοχής εργαζομένου σε ομάδες: πρωτότυπα	74
3.51	Η Οθόνη περιορισμών βαρδιών εργαζομένου: πρωτότυπα	75
3.52	Η Οθόνη επεξεργασίας περιορισμού βαρδιών εργαζομένου: πρωτότυπα	75
3.53	Η Οθόνη προτιμήσεων βαρδιών εργαζομένου: σκαρίφημα και πρωτότυπα	76
3.54	Η Οθόνη προτιμήσεων συναδέλφων εργαζομένου: σκαρίφημα και πρωτότυπα	77
4.1	Η πλήρης, υψηλού επιπέδου αρχιτεκτονική της εφαρμογής	80
4.2	Το Διάγραμμα Οντοτήτων-Συσχετίσεων για την εφαρμογή Horario	85

4.3	Η δόμηση των δεδομένων της εφαρμογής Horario στο Firestore	87
4.4	Η δόμηση των δεδομένων της εφαρμογής Horario στο Firestore (συνέχεια)	88
5.1	Μέθοδοι αξιολόγησης της εφαρμογής	94
5.2	Αποτελέσματα χρονομέτρησης αλγορίθμου αυτόματης ανάθεσης βαρδιών	106
7.1	Παλαιό πρωτότυπο της Ομαδικής προβολής του προγράμματος εργασίας	120
7.2	Απορριφθέν πρωτότυπο της Προβολής λίστας του προγράμματος εργασίας	120
7.3	Απορριφθέν πρωτότυπο του Φύλλου προσθήκης βάρδιας στον Οδηγό δημιουργίας / επεξεργασίας προγράμματος εργασίας	121
7.4	Σενάριο αξιολόγησης εμπειρίας χρήστη με πραγματικούς χρήστες	122
7.5	Σενάριο αξιολόγησης εμπειρίας χρήστη με πραγματικούς χρήστες (συνέχεια)	123

1 Εισαγωγή

Η παρούσα διπλωματική εργασία πραγματεύεται τη σχεδίαση και την ανάπτυξη μίας εφαρμογής χρονοπρογραμματισμού βαρδιών για μεγάλες επιχειρήσεις. Ο κύριος σκοπός της εφαρμογής αποτελεί την αυτοματοποίηση της διαδικασίας κατάρτισης του προγράμματος εργασίας, μέσα από μία διεπαφή φιλική προς τον χρήστη, σχεδιασμένη με γνώμονα τις πρακτικές του ανθρωποκεντρικού σχεδιασμού και την ευχρηστία.

Στο Κεφάλαιο 2 γίνεται μία βιβλιογραφική επισκόπηση γύρω από τα θέματα της ανάπτυξης cross-platform εφαρμογών και του ανθρωποκεντρικού σχεδιασμού. Όσον αφορά το πρώτο σκέλος, γίνεται μία έρευνα περί των τάσεων που επικρατούν στη χρήση εφαρμογών και τις συσκευές που χρησιμοποιούνται, και παρουσιάζονται λύσεις που έχουν σχεδιαστεί για ενιαία ανάπτυξη εφαρμογών προς τις περισσότερες πλατφόρμες. Όσον αφορά το δεύτερο σκέλος, παρουσιάζεται η βασική θεωρία του ανθρωποκεντρικού σχεδιασμού, τα μοντέλα ανάπτυξης λογισμικού που ακολουθούντο παραδοσιακά και πώς αυτά έχουν εξελιχθεί σήμερα. Γίνεται επίσης αναφορά σε θέματα περί ευχρηστίας και εμπειρίας χρήστη.

Στο Κεφάλαιο 2 γίνεται, ακόμα, ανάλυση του προβλήματος του χρονοπρογραμματισμού εργασίας από μαθηματικής πλευράς. Αφού παρουσιάστει η βασική θεωρία του γραμμικού προγραμματισμού και τα ζητούμενα του προβλήματος αυτού καθαυτού, σχηματίζεται ένα μαθηματικό μοντέλο επίλυσης του προβλήματος, το οποίο λαμβάνει υπόψη τις παραμέτρους που τίθενται.

Το Κεφάλαιο 3 πραγματεύεται τη σχεδίαση της εφαρμογής. Αφού γίνει ανάλυση των απαιτήσεων αυτής με βάση τις αρχές του ανθρωποκεντρικού σχεδιασμού, μελετώνται ανταγωνιστικές λύσεις και πραγματοποιούνται σχεδιαστικές επιλογές που ακολουθούνται καθ' όλη τη διάρκεια της σχεδίασης και της υλοποίησης. Έπειτα, παρουσιάζονται τα σχέδια της γραφικής διεπαφής, σε μορφή σκαριφημάτων και πρωτότυπων.

Στο Κεφάλαιο 4 αναλύονται θέματα υλοποίησης της εφαρμογής. Αφού πραγματοποιηθούν επιλογές αρχιτεκτονικής λογισμικού, σχηματίζεται το μοντέλο δεδομένων και υλοποιείται η βάση δεδομένων της εφαρμογής. Παράλληλα, αναπτύσσεται η μηχανή αυτόματης ανάθεσης βαρδιών, που υλοποιεί τον μαθηματικό αλγόριθμο χρονοπρογραμματισμού εργασίας.

Στο Κεφάλαιο 5 παρουσιάζονται οι μέθοδοι αξιολόγησης που χρησιμοποιήθηκαν για την εφαρμογή. Αξιολογείται τόσο η ίδια η εφαρμογή από πλευράς εμπειρίας χρήστη, με βάση τις αρχές του ανθρωποκεντρικού σχεδιασμού, αλλά και η αποδοτικότητα του αλγορίθμου χρονοπρογραμματισμού.

2 Βιβλιογραφική επισκόπηση

2.1 Ανάπτυξη cross-platform εφαρμογών

Στη σημερινή εποχή, οι ηλεκτρονικές συσκευές με τις οποίες αλληλεπιδρά κανείς είναι ποικίλες. Από τον κλασικό υπολογιστή, μέχρι τα έξυπνα κινητά (smartphones), τις ταμπλέτες και τις φορετές συσκευές (wearables), η ανάγκη του ανθρώπου να εκτελεί την εργασία του από μία πληθώρα συσκευών ολοένα αυξάνεται. Ανάλογα αυξάνεται και ο φόρτος εργασίας των σχεδιαστών και μηχανικών λογισμικού, οι οποίοι πρέπει να διασφαλίζουν την άψογη λειτουργικότητα των προϊόντων τους σε όλα τα περιβάλλοντα.

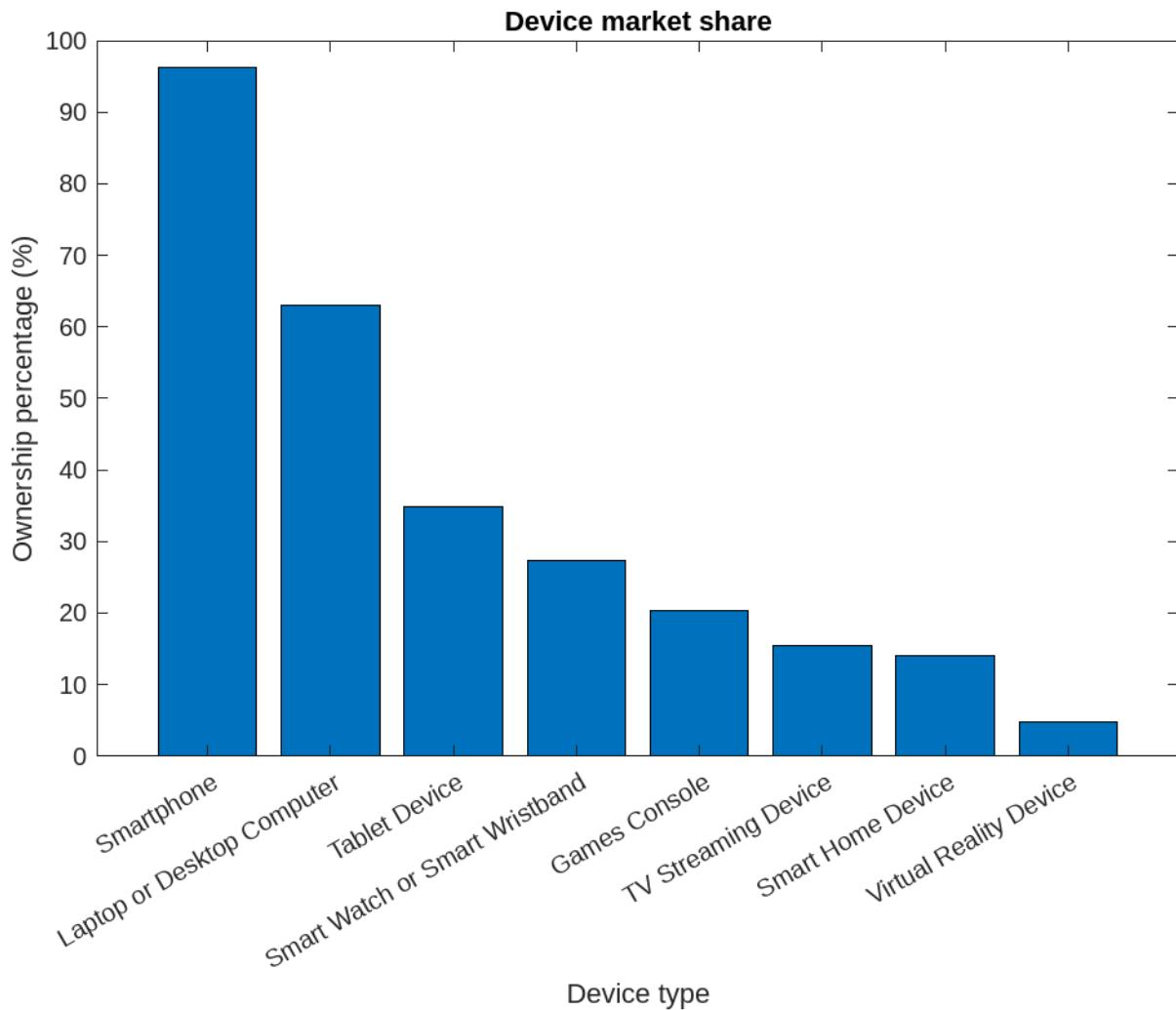
Στην παρούσα ενότητα, θα αναλυθεί η αναγκαιότητα ανάπτυξης εφαρμογών για πολλαπλές πλατφόρμες και λειτουργικά συστήματα και θα παρουσιαστούν λύσεις που έχουν προταθεί για διευκόλυνση του έργου των μηχανικών λογισμικού. Επίσης, θα παρουσιαστεί το Flutter, ένα προγραμματιστικό περιβάλλον για ανάπτυξη εφαρμογών, οι οποίες καλύπτουν μία πληθώρα συστημάτων.

2.1.1 Συσκευές, λειτουργικά συστήματα και native apps

Τα τελευταία χρόνια, το μερίδιο χρήσης ηλεκτρονικών συσκευών από τους καταναλωτές γίνεται ολοένα πιο ευρύ. Σύμφωνα με δεδομένα του DataReportal (Kemp 2022), οι πιο δημοφιλείς τύποι συσκευών είναι τα έξυπνα κινητά (smartphones), οι ηλεκτρονικοί υπολογιστές (laptops ή desktop computers), οι ταμπλέτες, τα έξυπνα ρολόγια, οι παιχνιδομηχανές, οι συσκευές ροής δεδομένων τηλεοπτικού περιεχομένου (TV streaming), οι συσκευές έξυπνου σπιτιού (smart home) και οι συσκευές εικονικής πραγματικότητας, με ποσοστά που φαίνονται στην Εικόνα 2.1.

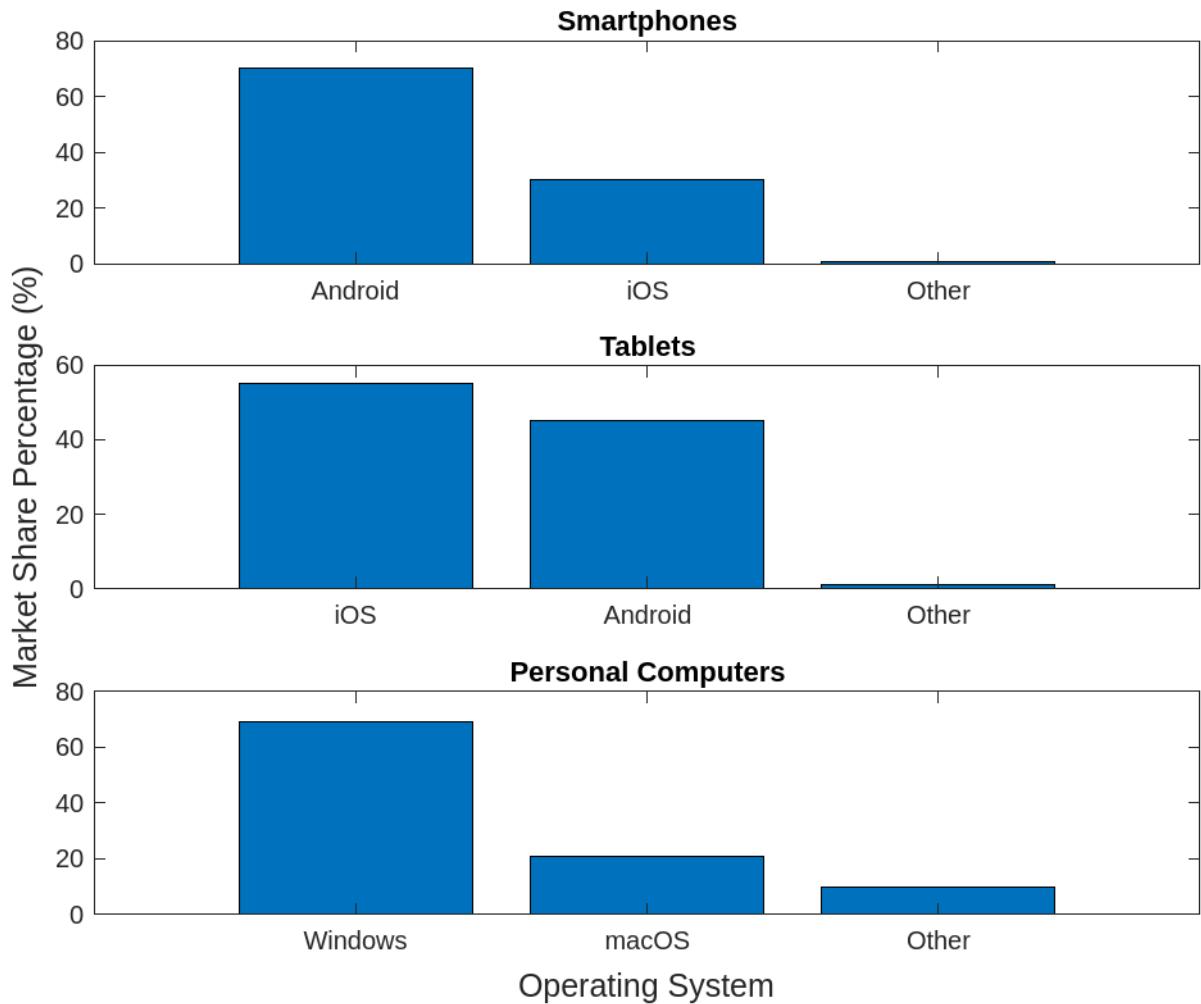
Μοιρασμένη είναι επίσης η αγορά των χρησιμοποιούμενων λειτουργικών συστημάτων ανά συσκευή. Σύμφωνα με τα (StatCounter GlobalStats 2023) και (Statista Research Department 2023), περίπου το 70% των smartphones διαθέτει λειτουργικό σύστημα Android, ενώ περίπου το υπόλοιπο 30% λειτουργεί με Apple iOS. Στις ταμπλέτες το χάσμα είναι μικρότερο, όπου προηγείται το λειτουργικό σύστημα iOS με περίπου 55%, ενώ ακολουθεί το Android με ποσοστό περίπου 45%. Τέλος, στον τομέα των ηλεκτρονικών υπολογιστών, κυριαρχούν τα Microsoft Windows με ποσοστό περίπου 69%, ακολουθούν οι εκδόσεις της Apple macOS με περίπου 21%, ενώ το υπόλοιπο 10% σχηματίζουν άλλα λειτουργικά

2.1 Ανάπτυξη cross-platform εφαρμογών



Εικόνα 2.1. Μερίδιο αγοράς ηλεκτρονικών συσκευών. Δεδομένα Ιανουαρίου 2022 (Kemp 2022).

συστήματα, όπως διανομές Linux, Chrome OS, κ.ά. Τα αντίστοιχα ποσοστά φαίνονται στην Εικόνα 2.2.



Εικόνα 2.2. Μερίδιο αγοράς λειτουργικών συστημάτων. Δεδομένα Νοεμβρίου 2023 (StatCounter GlobalStats 2023)

Από τα προηγούμενα, μπορεί κανείς να συμπεράνε ότι οι σχεδιαστές και οι προγραμματιστές μίας εφαρμογής¹ πρέπει, κατά τη διάρκεια σχεδίασης, ανάπτυξης και υποστήριξης, να λαμβάνουν υπόψη όλα τα διαφορετικά περιβάλλοντα χρήσης του προϊόντος τους.

Επομένως, η σχεδίαση και η ανάπτυξη εφαρμογών έχει διαφοροποιηθεί πλέον σε σύγκριση με το παρελθόν, αφού θα πρέπει να υποστηρίζει πολλαπλές πλατφόρμες, στάνταρ, πρωτόκολλα και

¹Σύμφωνα με την εγκυκλοπαίδεια Britannica (Gregersen 2023), εφαρμογή (application) είναι το λογισμικό εκείνο που έχει σχεδιαστεί ώστε να διαχειρίζεται συγκεκριμένες εργασίες των χρηστών. Τέτοιο λογισμικό οδηγεί τον υπολογιστή να εκτελεί εντολές που προέρχονται από τον χρήστη και ενδέχεται να περιλαμβάνει ένα πρόγραμμα διαχείρισης δεδομένων για λογαριασμό του χρήστη.

2.1 Ανάπτυξη cross-platform εφαρμογών

τεχνολογίες δικτύου. Επιπλέον, δεδομένου ότι τη μεγαλύτερη μερίδα της αγοράς ηλεκτρονικών συσκευών κατέχουν τα smartphones και οι ταμπλέτες, οι ίδιοι πρέπει να λαμβάνουν υπόψη τις περιορισμένες δυνατότητες που συχνά διαθέτουν οι συσκευές αυτές, αν και λόγω της τεχνολογικής εξέλιξης, αυτές συνεχώς αυξάνονται (Delia κ.ά. 2015).

Ο ιδανικότερος τρόπος για να υποστηρίξει κανείς όλα τα επιμέρους λειτουργικά συστήματα είναι με την ανάπτυξη μίας ξεχωριστής εφαρμογής για το καθένα. Οι λεγόμενες native εφαρμογές προσφέρουν την πλουσιότερη εμπειρία χρήσης (user experience), αφού έχουν γρήγορη απόδοση και αισθητική σύμφωνη με το υπόλοιπο σύστημα, ενώ ταυτόχρονα διαχειρίζονται με τον καλύτερο δυνατό τρόπο τους διαθέσιμους πόρους, έχοντας πλήρη πρόσβαση στο υποκείμενο υλικό (Xanthopoulos και Xinogalos 2013).

Από την άλλη, αυτό εισάγει μία σημαντική δυσκολία για τους μηχανικούς λογισμικού, αφού για κάθε λειτουργικό σύστημα θα πρέπει να χρησιμοποιηθεί διαφορετική γλώσσα προγραμματισμού (για παράδειγμα, Java ή Kotlin για το Android, Swift για το iOS κ.λπ.), ενώ επιπλέον τα διαθέσιμα APIs² και SDKs³ είναι διαφορετικά. Έτσι, είτε απαιτείται γνώση πολλών διαφορετικών προγραμματιστικών περιβαλλόντων, είτε η συνεργασία πολλών προγραμματιστών σε μία ομάδα, γεγονός που οδηγεί σε αυξημένο κόστος παραγωγής.

Με τις native εφαρμογές, λοιπόν, παρατηρείται ένα φαινόμενο γνωστό ως τμηματοποίηση (fragmentation), επειδή για το ίδιο προϊόν απαιτούνται πολλαπλά εκτελέσιμα αρχεία. Εξάλλου, η προγραμματιστική αρχή write-once, run-anywhere (WORA), που εισήχθη ως όραμα της Java, δεν μπορεί να εφαρμοστεί με αυτόν τον τρόπο (Gavalas και Economou 2011). Έτσι, άρχισαν σταδιακά να γίνονται προσπάθειες προς μία νέα κατεύθυνση της μηχανικής λογισμικού, που κατέληξε στις σημερινές cross-platform λύσεις.

2.1.2 Εξέλιξη των cross-platform εφαρμογών

Πλέον, το ενδιαφέρον των μηχανικών λογισμικού έχει στραφεί στις cross-platform λύσεις. Ο στόχος αυτής της προσέγγισης είναι η κατάκτηση επιδόσεων εφαρμίλλων των native εφαρμογών και η βελτιστοποίηση του λόγου κόστους - κέρδους, ενώ ο κώδικας γράφεται μία φορά και τρέχει σε όσες περισσότερες πλατφόρμες αυτό είναι δυνατό (Xanthopoulos και Xinogalos 2013) (Delia κ.ά. 2015).

Από τις πρώτες λύσεις που δόθηκαν στο πρόβλημα, ήταν η ανάπτυξη εφαρμογών ιστού (web applications), οι οποίες φιλοξενούνται σε έναν εξυπηρετητή και τρέχουν στις συσκευές των χρηστών

²Ενα API (Application Programming Interface) είναι ένα σύνολο κανόνων που επιτρέπουν σε διαφορετικές εφαρμογές να επικοινωνούν μεταξύ τους. Λειτουργεί ως ενδιάμεσο επίπεδο που χειρίζεται τη μεταφορά δεδομένων μεταξύ συστημάτων, επιτρέποντας στις εταιρείες να καθιστούν διαθέσιμα τα δεδομένα και τις λειτουργίες των εφαρμογών τους σε τρίτους (IBM Topics, χ.χ.).

³Ενα SDK (Software Development Kit) είναι ένα σύνολο εργαλείων που διατίθεται από κατασκευαστές υλικού και λογισμικού και βοηθά τους προγραμματιστές να κατασκευάζουν εφαρμογές για συγκεκριμένες πλατφόρμες (Yasar και Rosencrance 2022).

μέσω ενός φυλλομετρητή. Πράγματι, οι εφαρμογές ιστού είναι ανεξάρτητες λογισμικού, αφού ο φυλλομετρητής λειτουργεί ως ένα αφαιρετικό στρώμα μεταξύ της εφαρμογής και του λειτουργικού συστήματος του πελάτη. Ωστόσο, προσφέρουν φτωχότερη εμπειρία χρήσης, αφού για την εκτέλεση μίας τέτοιας εφαρμογής απαιτείται συνεχής σύνδεση στο Διαδίκτυο και η λήψη του κώδικα από τον εξυπηρετητή στον πελάτη κάθε φορά που ανανεώνεται η σελίδα, οδηγεί σε καθυστερήσεις στην απόδοση. Επιπλέον, ο φυλλομετρητής δεν επιτρέπει τη χρήση όλων των πόρων του συστήματος (όπως κάμερα, GPS, επιταχυνσιόμετρο), ενώ δεν είναι δυνατή ούτε η εκτέλεση της εφαρμογής στο παρασκήνιο, που ενδεχομένως να ήταν επιθυμητή για τη λήψη ενημερώσεων από τον χρήστη. Συμπεραίνει κανείς, επομένως, πως οι εφαρμογές ιστού αποτελούν μία εξυπηρετική λύση για τους προγραμματιστές, αλλά συχνά προκαλούν τη δυσαρέσκεια των πελατών, λόγω της κακής εμπειρίας χρήσης (Delia κ.ά. 2015).

Εξέλιξη των εφαρμογών ιστού αποτέλεσαν οι υβριδικές εφαρμογές (hybrid apps). Οι εφαρμογές αυτές εγκαθίστανται στη συσκευή, όπως μία native εφαρμογή, ωστόσο περιλαμβάνουν ένα web container, μέσα στο οποίο εκτελείται η web εφαρμογή. Οι υβριδικές εφαρμογές επιτρέπουν την επαναχρησιμοποίηση του κώδικα, όπως οι web εφαρμογές, και επιπλέον προσφέρουν περισσότερη λειτουργικότητα, αφού μπορούν να επικοινωνούν με τους πόρους της συσκευής μέσω API. Εξάλλου, η εγκατάσταση και η διανομή της εφαρμογής από πλατφόρμες, όπως το Google Play Store και το App Store, επιτρέπει γρήγορες αναβαθμίσεις και διορθώσεις σφαλμάτων (Delia κ.ά. 2015) (Xanthopoulos και Xinogalos 2013) (Shah, Sinha, και Mishra 2019). Παρ' όλα αυτά, τέτοιες λύσεις χαρακτηρίζονται από φτωχή εμπειρία χρήσης, αφού συνήθως η γραφική διεπαφή τους δεν περιλαμβάνει native στοιχεία, ενώ οι χρόνοι εκτέλεσης είναι αργοί, λόγω της συνεχούς επικοινωνίας με τον εξυπηρετητή (Mascetti κ.ά. 2021). Μία τέτοια λύση αποτελεί το Apache Cordova, παλαιότερα γνωστό ως PhoneGap, το οποίο χρησιμοποιεί WebView για τον σχεδιασμό της διεπαφής και Plugins για την επικοινωνία με την υποκείμενη συσκευή (Shah, Sinha, και Mishra 2019).

Πλέον, λόγω των μειονεκτημάτων που προαναφέρθηκαν, το ενδιαφέρον αρχίζει να απομακρύνεται από τις web-based εφαρμογές. Στη θέση τους, έδαφος κερδίζουν οι διερμηνευόμενες εφαρμογές (interpreted apps) και οι εφαρμογές που προκύπτουν από παράλληλη μεταγλώττιση (cross-compilation). Οι μεν διερμηνευόμενες εφαρμογές προσφέρουν μία native γραφική διεπαφή, η οποία προκύπτει από διερμήνευση κατά τον χρόνο εκτέλεσης (runtime) και είναι ανεξάρτητες πλατφόρμας (Delia κ.ά. 2015). Χαρακτηριστικό παράδειγμα αποτελεί το React Native, ένα cross-platform development framework που χρησιμοποιεί μηχανή JavaScript. Αρκετά γνωστό σε αυτή την κατηγορία, ήταν παλαιότερα επίσης το Appcelerator Titanium.

Αντίθετα, οι cross-compiled εφαρμογές μεταφράζονται πλήρως σε native κώδικα κατά τη διάρκεια της μεταγλώττισης (compile-time), προσφέροντας έτσι τις υψηλότερες επιδόσεις σε σύγκριση με όλα τα υπόλοιπα frameworks. Οι εφαρμογές αυτές είναι συνήθως widget-based, δηλαδή όλα τα στοιχεία είναι widgets⁴, που λειτουργούν ως δομικοί λίθοι για το χτίσιμο της διεπαφής (Delia κ.ά. 2015) (Shah,

⁴Widget ονομάζεται οτιδήποτε μπορεί να οριστεί ως δομικό στοιχείο, στοιχείο καθορισμού στυλ ή χαρακτηριστικό

2.1 Ανάπτυξη cross-platform εφαρμογών

Sinha, και Mishra 2019). Ένα από τα πιο γνωστά cross-compilation frameworks είναι το Flutter, που θα παρουσιαστεί στην Ενότητα 2.1.3.

Πολλές εργασίες έχουν αφοσιωθεί στη σύγκριση και την αξιολόγηση των διαφόρων cross-platform development frameworks. Οι κ.κ. (Xanthopoulos και Xinogalos 2013) καταλήγουν στο συμπέρασμα ότι δεν υπάρχει εν γένει βέλτιστη λύση, αλλά η επιλογή framework πρέπει να παίρνεται ξεχωριστά για κάθε project, με βάση πολλούς παράγοντες, όπως:

- την διάθεση της εφαρμογής σε σουίτες όπως Google Play Store και το App Store,
- τις υποστηριζόμενες τεχνολογίες,
- την ανάγκη για πρόσβαση στο υποκείμενο υλικό και τα δεδομένα,
- την αισθητική της γραφικής διεπαφής και
- τις επιδόσεις όπως τις αντιλαμβάνεται ο χρήστης.

Εξάλλου, όπως είναι κατανοητό, υπάρχουν πολλά cross-platform development frameworks, λίγα από τα οποία έγιναν ευρέως αποδεκτά, λόγω της κακής επίδοσης και σταθερότητας. Από το 2020 και έπειτα, το Flutter, το React Native και το Xamarin άρχισαν να κερδίζουν έδαφος στον χώρο, λόγω καλύτερων επιδόσεων (You και Hu 2021). Οι κ.κ. (Mascetti κ.ά. 2021) ερεύνησαν την προσβασιμότητα των εφαρμογών που χτίζονται με cross-platform development frameworks και κατέληξαν στο συμπέρασμα ότι αρκετά από τα native APIs προσβασιμότητας δεν είναι διαθέσιμα μέσω τέτοιων frameworks. Οι προγραμματιστές εξακολουθούν να μπορούν να τα χρησιμοποιούν, αλλά μόνο μέσω κώδικα που εξαρτάται από την εκάστοτε πλατφόρμα.

Εν κατακλείδι, μπορεί κανείς από τα παραπάνω να συμπεράνει, πως το Flutter είναι μία από τις καλύτερες επιλογές για cross-platform development framework. Για αυτόν τον λόγο, θα χρησιμοποιηθεί στην παρούσα εργασία για την κατασκευή της εφαρμογής, και παρουσιάζεται στην επόμενη ενότητα.

2.1.3 Flutter: Περιγραφή και χαρακτηριστικά

Το Flutter είναι ένα software development kit (SDK), που δημιουργήθηκε από την Google για την ανάπτυξη εφαρμογών υψηλής πιστότητας και απόδοσης. Αρχικά, σχεδιάστηκε για να εξυπηρετήσει εφαρμογές για κινητές συσκευές, στοχεύοντας τις πλατφόρμες Android και iOS. Ωστόσο, γρήγορα επεκτάθηκε και σε άλλες πλατφόρμες, ώσπου πλέον υποστηρίζει εφαρμογές ιστού, υπολογιστή (εφαρμογές για Windows και macOS), αλλά και ενσωματωμένα (embedded) συστήματα (Copperchips 2023) (‘Flutter Official Website’ , χ.χ.).

Η ταχύτητα του Flutter έγκειται στο γεγονός ότι η γραφική διεπαφή σχεδιάζεται σε έναν καμβά από το ίδιο το Flutter, χωρίς να βασίζεται στην υποκείμενη πλατφόρμα. Έτσι, δεν υπάρχει ανάγκη για

διάταξης. Συνήθως διακρίνονται σε stateless widgets, τα οποία δεν ανταποκρίνονται σε ενέργειες του χρήστη, και σε stateful widgets, τα οποία μπορούν να αλλάξουν την κατάστασή τους δυναμικά (Shah, Sinha, και Mishra 2019).

κάποια γέφυρα με το λειτουργικό σύστημα, όπως με το React Native, ούτε απαιτούνται εναλλαγές περιβάλλοντος (context switches) για την επίτευξη αυτού του σκοπού (Aguinis 2019).

Το Flutter βασίζεται στη γλώσσα προγραμματισμού Dart, μία αντικειμενοστραφής γλώσσα που σχεδιάστηκε έτσι, ώστε να είναι φιλική προς τον προγραμματιστή, εύκολη στην εκμάθηση και γρήγορη στην εκτέλεση, χωρίς να στερείται καμία λειτουργικότητα από άλλες αντικειμενοστραφείς γλώσσες, όπως η Java, η JavaScript ή η C#. Ένα σημαντικό πλεονέκτημα της Dart αποτελεί το γεγονός ότι χρησιμοποιεί και Just-in-Time (JIT) μεταγλωττιζόμενο κώδικα, επιτρέποντας έτσι γρήγορη σχεδίαση και αποσφαλμάτωση (debugging), με το λεγόμενο Hot Reload, το οποίο επιτρέπει στον προγραμματιστή να βλέπει τις αλλαγές που κάνει στον κώδικα, χωρίς καμία καθυστέρηση. Ταυτόχρονα όμως, όταν η εφαρμογή είναι έτοιμη για διάθεση, ο κώδικας είναι Ahead-of-Time (AoT) μεταγλωττιζόμενος, ούτως ώστε να προσφέρει ταχύτατες επιδόσεις κατά τη χρήση. Για τη σχεδίαση της γραφικής διεπαφής, το Flutter χρησιμοποιεί τη μηχανή σχεδίασης (rendering engine) Skia, η οποία είναι κατά βάση γραμμένη σε C++ (Aguinis 2019) (You και Hu 2021) (Shah, Sinha, και Mishra 2019).

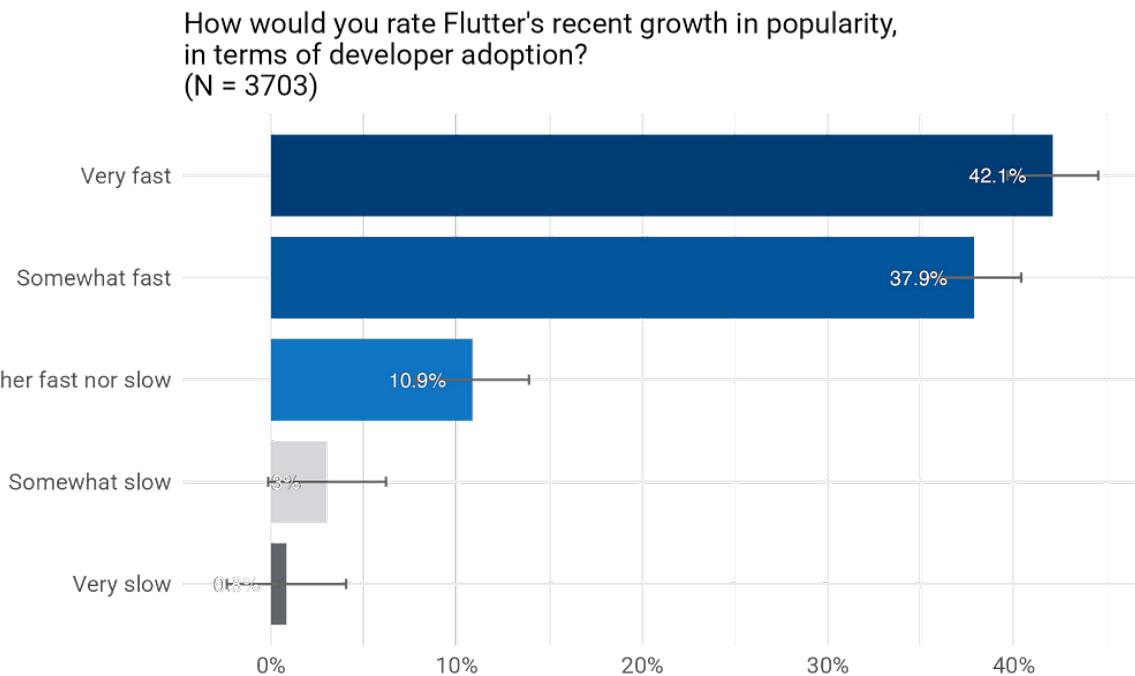
Το Flutter, λοιπόν, χαρακτηρίζεται από μία πληθώρα πλεονεκτημάτων, όπως:

- Υποστήριξη πολλών πλατφορμών από μία βάση κώδικα και ως εκ τούτου προσαρμοστικότητα σε πολλαπλά μεγέθη οθονών
- Εύκολος προγραμματισμός, αποσφαλμάτωση και δοκιμές
- Παροχή πληθώρας έτοιμων widgets, αλλά και εύκολη προσαρμογή νέων
- Επιδόσεις εφάμιλλες των native εφαρμογών
- Ανοιχτού κώδικα, δωρεάν
- Υποστήριξη από τη Google

Στον αντίποδα, τα εκτελέσιμα αρχεία που προκύπτουν για κάθε πλατφόρμα τείνουν να είναι μεγαλύτερα από τις υπόλοιπες λύσεις (με βέλτιστο μέγεθος αυτό των native εφαρμογών) (Copperchips 2023) (Keshav 2021) (You και Hu 2021).

Οι παραπάνω λόγοι κάνουν το Flutter μία από τις πιο αγαπητές λύσεις στο κοινό των προγραμματιστών αυτή τη στιγμή, όπως δείχνουν στατιστικές αναλύσεις (J. Lee 2023). Σε έρευνα που πραγματοποιήθηκε το πρώτο τρίμηνο του 2023, περισσότεροι από 13.000 ερωτηθέντες απάντησαν κατά 93% θετικά ως προς το Flutter, με το 55% αυτών να είναι πολύ ευχαριστημένοι. Ενδεικτικά, στην Εικόνα 2.3, φαίνεται η άποψη του κοινού για την ταχύτητα ανάπτυξης της δημοτικότητας του Flutter. Ο ενδιαφερόμενος αναγνώστης μπορεί να διαβάσει περισσότερα αποτελέσματα στο (J. Lee 2023).

2.2 Ανθρωποκεντρικός σχεδιασμός εφαρμογών



Εικόνα 2.3. Αποτελέσματα έρευνας για την ανάπτυξη της δημοτικότητας του Flutter. Δεδομένα πρώτου τριμήνου 2023 (J. Lee 2023).

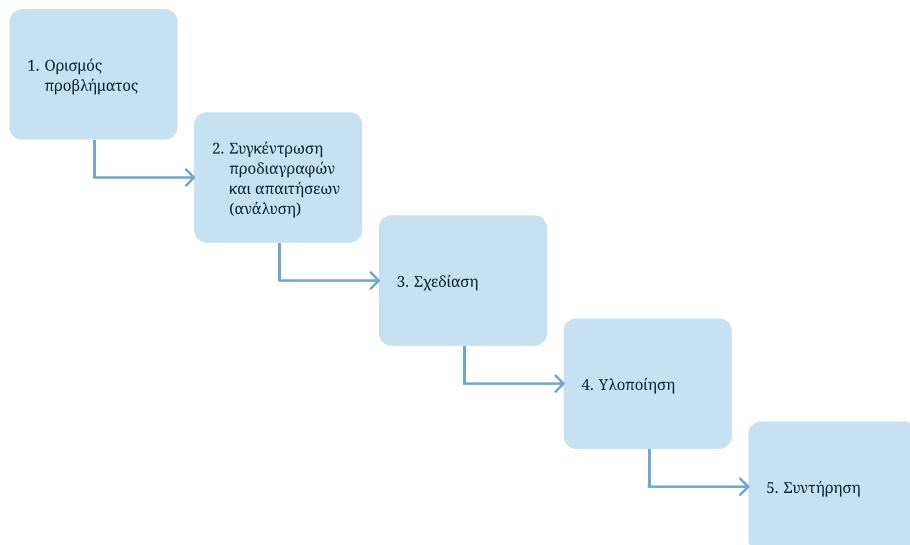
2.2 Ανθρωποκεντρικός σχεδιασμός εφαρμογών

Η σχεδίαση διαδραστικών συστημάτων (interactive systems design) είναι μία επιστήμη, η οποία έκανε την εμφάνισή της τη δεκαετία του 1970. Είναι μία διαδικασία επίλυσης προβλημάτων, με σκοπό την παραγωγή όχι μόνο της ενδιάμεσης αναπαράστασης (σχεδίου), αλλά και του ίδιου του διαδραστικού μέσου, κατ' αναλογία με άλλους κλάδους της μηχανικής. Η διαδικασία αυτή κατευθύνεται από τους στόχους των χρηστών και τον τρόπο που αυτοί πρόκειται να χρησιμοποιήσουν τη λύση του προβλήματος (Αβιούρης κ.ά. 2018).

Ως ανθρωποκεντρικός (human-centered) ή χρηστοκεντρικός σχεδιασμός (user-centered design) ορίζονται οι μέθοδοι ανάπτυξης διαδραστικών συστημάτων που έχουν αναπτυχθεί στο πλαίσιο της επιστήμης της αλληλεπίδρασης ανθρώπου - υπολογιστή (human - computer interaction) και οι οποίες δίνουν έμφαση στη μελέτη του ανθρώπου - χρήστη και τοποθετούν τον ίδιο και τις ανάγκες του στο κέντρο της διαδικασίας σχεδίασης του προϊόντος, τόσο κατά την ανάλυση, τη σχεδίαση και την αξιολόγησή του.

2.2.1 Μοντέλα ανάπτυξης λογισμικού

Με την πρόοδο του χρόνου και την έντονη ανάπτυξη της συγκεκριμένης επιστήμης, έχουν δημιουργηθεί διαφορετικά πρότυπα που καθοδηγούν τη σχεδίαση ενός τέτοιου συστήματος, τα οποία ονομάζονται μοντέλα ανάπτυξης ή κύκλος ζωής συστημάτων λογισμικού. Το παραδοσιακό μοντέλο ανάπτυξης λογισμικού είναι το **μοντέλο καταρράκτη** (waterfall model), το οποίο συνιστά με ακολουθία διαδοχικών βημάτων (Εικόνα 2.4). Αν και το μοντέλο αυτό περιγράφει σαφώς τις φάσεις ανάπτυξης ενός προϊόντος, εντούτοις, το βασικότερο πρόβλημα είναι η αδυναμία λεπτομερούς προδιαγραφής ενός προϊόντος πριν την υλοποίησή του (Αβούρης κ.ά. 2018).

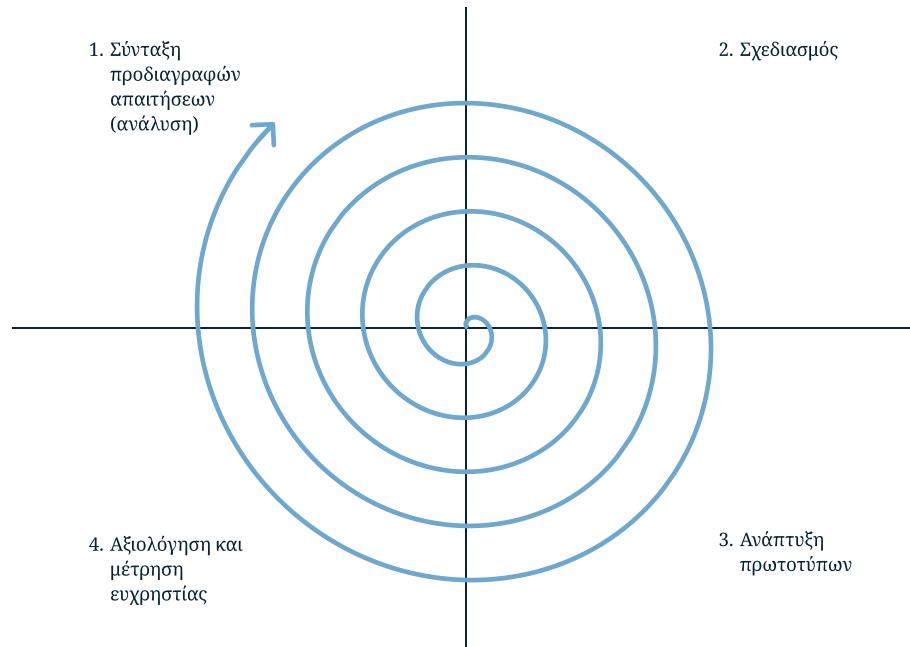


Εικόνα 2.4. Το μοντέλο καταρράκτη

Το **ελικοειδές μοντέλο** (spiral model) αντιμετωπίζει τα προβλήματα του μοντέλου καταρράκτη, επαναλαμβάνοντας τα επιμέρους στάδιά του, σε μία διαδικασία διαδοχικών βελτιώσεων του αρχικού πρωτότυπου (Εικόνα 2.5). Κάθε νέο πρωτότυπο που δημιουργείται, αξιολογείται με βάση τις αρχικές απαιτήσεις, και βελτιώνεται διαρκώς, με έναν αυξανόμενο βαθμό λεπτομέρειας (Αβούρης κ.ά. 2018).

Στο **αστεροειδές μοντέλο** (star model), αντίθετα με τα δύο προαναφερθέντα, δεν είναι απαραίτητη η αυστηρή ακολουθία των επιμέρους βημάτων (Εικόνα 2.6). Η διαδικασία σχεδίασης περιστρέφεται γύρω από την αξιολόγηση του συστήματος, ως κεντρική δραστηριότητα του μοντέλου, ενώ οι επιμέρους φάσεις (ανάλυση, σχεδίαση, υλοποίηση) μπορούν να ακολουθηθούν με οποιαδήποτε σειρά και από οποιοδήποτε σημείο έναρξης, αρκεί να συμπληρώνονται από μία φάση αξιολόγησης, είτε μέσω χρηστών του συστήματος, είτε μέσω ειδικών.

Ο ενδιαφερόμενος γύρω από τον ανθρωποκεντρικό σχεδιασμό εφαρμογών αναγνώστης μπορεί να ανατρέξει σε σχετικά συγγράμματα, όπως το (Αβούρης κ.ά. 2018) για περισσότερες λεπτομέρειες, εις βάθος ανάλυση των προαναφερθέντων και παρουσίαση επιπλέον μοντέλων σχεδιασμού. Καθ' όλη την



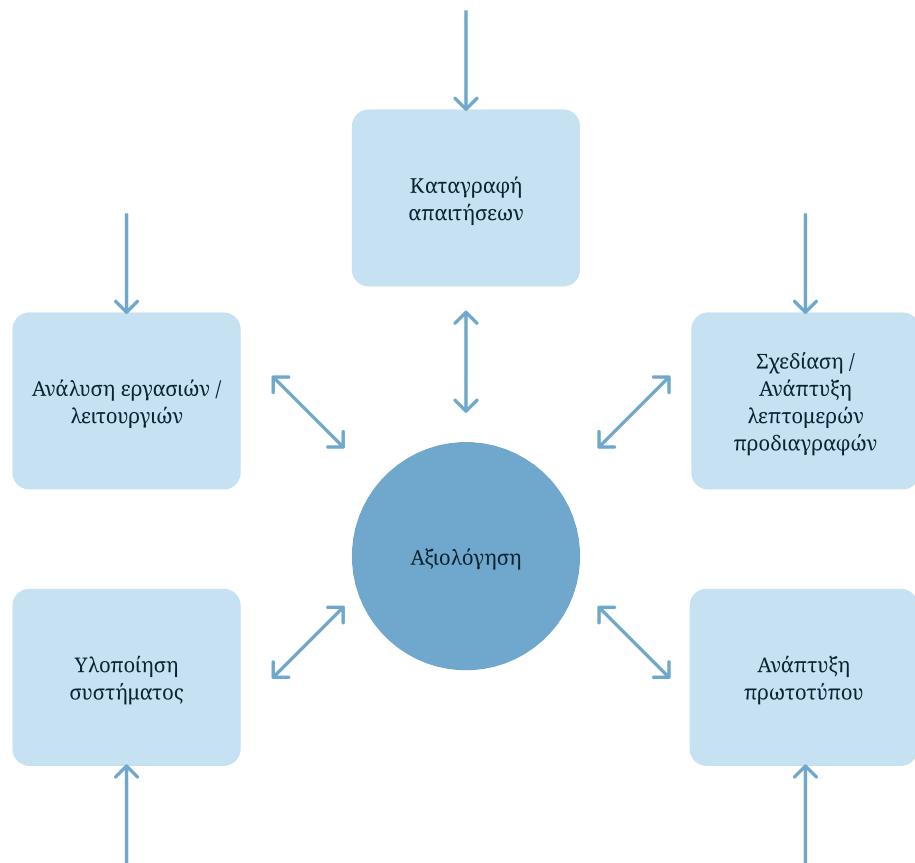
Εικόνα 2.5. Το ελικοειδές μοντέλο

έκταση της παρούσας εργασίας, εφαρμόζονται οι τεχνικές και οι πρακτικές του ανθρωποκεντρικού σχεδιασμού, τόσο κατά την ανάλυση του προβλήματος που η εφαρμογή καλείται να επιλύσει (βλ. Ενότητα 3.1), τη σχεδίαση της εφαρμογής καθεαυτής (βλ. Ενότητες 3.3 και 3.4), την υλοποίησή της (βλ. Κεφάλαιο 4) και την αξιολόγησή της (βλ. Κεφάλαιο 5).

2.2.2 Ευχρηστία συστήματος

Όπως αναφέρθηκε προηγουμένως, αρχή του ανθρωποκεντρικού σχεδιασμού είναι η διαδικασία σχεδίασης συστημάτων που τοποθετεί στο κέντρο τον άνθρωπο - χρήστη και τις ανάγκες του. Με άλλα λόγια, σκοπός αυτής της διαδικασίας είναι η σχεδίαση ενός συστήματος φιλικού προς τους χρήστες, δηλαδή ένα σύστημα το οποίο αυτοί θα μπορούν να χρησιμοποιούν με ευκολία. Ένα τέτοιο σύστημα ονομάζεται εύχρηστο. Για να ορίσουμε την **ευχρηστία** (usability) ενός συστήματος, χρειάζεται να γνωρίζουμε 3 δεδομένα (Αβιούρης κ.ά. 2018):

- Τους χρήστες (users), οι οποίοι χρησιμοποιούν το σύστημα. Πρέπει να ορίσουμε το προφίλ των συγκεκριμένων ανθρώπων, καταγράφοντας χαρακτηριστικά τους, όπως το επίπεδο γνώσεών τους γύρω από υπολογιστές.
- Τις εργασίες (tasks), τις οποίες οι άνθρωποι επιτελούν ή θα ήθελαν να επιτελέσουν μέσα από το σύστημα. Επομένως, μας ενδιαφέρουν οι στόχοι τους, δηλαδή οι λόγοι για τους οποίους



Εικόνα 2.6. Το αστεροειδές μοντέλο

2.2 Ανθρωποκεντρικός σχεδιασμός εφαρμογών

χρησιμοποιούν το σύστημα.

- **Το πλαίσιο (context)**, δηλαδή τις συνθήκες κάτω από τις οποίες χρησιμοποιούν οι χρήστες το σύστημα. Για παράδειγμα, το περιβάλλον αποτελεί σημαντικό φυσικό πλαίσιο. Αν ο χώρος στον οποίο γίνεται συνήθως η χρήση μίας εφαρμογής είναι θορυβώδης (π.χ. ένα εργοστάσιο), τότε οι ηχητικές ειδοποιήσεις (*sound notifications*) δεν αποτελούν καλή λύση για την εφαρμογή, καθώς ενδέχεται οι χρήστες να μην τις αντιλαμβάνονται.

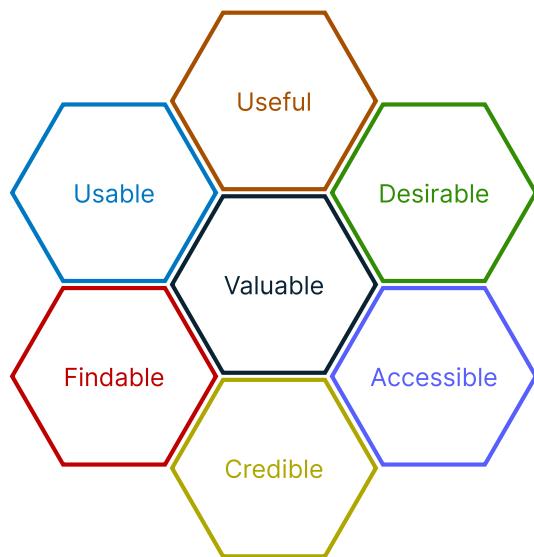
Τότε, είμαστε σε θέση να αποφανθούμε για την ευχρηστία μίας εφαρμογής σε 3 επίπεδα (Αβούρης κ.ά. 2018):

- **Αποτελεσματικότητα (effectiveness)**, δηλαδή καταγράφουμε το πόσο καλά πετυχαίνει τις εργασίες των χρηστών.
- **Αποδοτικότητα (efficiency)** με την οποία γίνονται οι εργασίες αυτές. Συνήθως, η μετρική που μας ενδιαφέρει είναι ο χρόνος εκπλήρωσης των εργασιών. Έτσι, μπορούμε εύκολα να μετρήσουμε την αποδοτικότητα ενός συστήματος, χρησιμοποιώντας ένα χρονόμετρο.
- **Υποκειμενική ικανοποίηση (user satisfaction)**, δηλαδή την τελική εντύπωση που αφήνει το σύστημα στους χρήστες. Ο μόνος τρόπος για να αξιολογήσουμε την υποκειμενική ικανοποίηση των χρηστών είναι να λάβουμε ανάδραση (*feedback*) από αυτούς, μέσω μεθόδων όπως ερωτηματολόγια, συνεντεύξεις, ομάδες εστίασης κ.λπ.

Ο στόχος του ανθρωποκεντρικού σχεδιασμού, ωστόσο, δεν περιορίζεται μονάχα στην επίτευξη της ευχρηστίας του συστήματος, αλλά στην προσφορά μιας καθ' όλα θετικής εμπειρίας χρήστη (*User Experience*). Η εμπειρία χρήστη περιλαμβάνει ως υποσύνολό της την ευχρηστία, μαζί με τις ακόλουθες έννοιες (Αβούρης κ.ά. 2018):

- **Ευρεσιμότητα (findability)**: η ευκολία εύρεσης λειτουργιών της εφαρμογής από τους χρήστες.
- **Προσβασιμότητα (accessibility)**: η δυνατότητα χρήσης της εφαρμογής από όλους τους ανθρώπους, συμπεριλαμβανομένων αυτών με ειδικές ανάγκες, όπως προβλήματα όρασης.
- **Αξιοπιστία (credibility)**: ορθότητα του περιεχομένου και των πληροφοριών που προσφέρει η εφαρμογή.
- **Αισθητική (desirability)**: οπτικός σχεδιασμός που αναδεικνύει την εφαρμογή και την κάνει ελκυστική προς τους χρήστες.
- **Χρησιμότητα (usefulness)**: σχετίζεται με το πόσο ωφέλιμες είναι οι λειτουργίες της εφαρμογής για τους χρήστες.

Ο παραπάνω ορισμός αποτυπώνεται εν συντομίᾳ στο UX **honeycomb**, ένα σχεδιάγραμμα που παρουσιάστηκε από τον Peter Morville το 2004 (Morville, χ.χ.) και παριστάνεται στην Εικόνα 2.7.



Εικόνα 2.7. UX honeycomb

2.2.3 Κανόνες μέτρησης ευχρηστίας

Το 1994 ο Nielsen πρότεινε ένα σύνολο από γενικούς κανόνες, με σκοπό τη μέτρηση της ευχρηστίας μίας εφαρμογής. Οι κανόνες αυτοί χρησιμοποιούνται κυρίως κατά τη διαδικασία της ευρετικής αξιολόγησης (heuristic evaluation) και αναφέρονται παρακάτω (Nielsen 1994):

1. Παροχή ανάδρασης της κατάστασης του συστήματος
2. Χρήση κατανοητής προς τους χρήστες γλώσσας
3. Παροχή εύκολων και σαφών εξόδων διαφυγής
4. Διατήρηση συνέπειας και συνέχειας, χρήση standards
5. Σχεδιασμός για αποτροπή σφαλμάτων χρήστη
6. Ελαχιστοποίηση μνημονικού φορτίου χρήστη
7. Προσαρμοστικότητα, παροχή συντομεύσεων
8. Μινιμαλισμός, αποφυγή περιττών στοιχείων
9. Παροχή σαφών μηνυμάτων λάθους
10. Επαρκής υποστήριξη - Βοήθεια και εγχειρίδια

2.3 Το πρόβλημα του Βέλτιστου Χρονοπρογραμματισμού Εργασίας

Η κατάστρωση του ωρολογίου προγράμματος εργασίας των εργαζομένων μίας επιχείρησης αποτελεί συχνά ένα περίπλοκο πρόβλημα. Η φύση πολλών επιχειρήσεων απαιτεί τη διαρκή στελέχωσή τους από προσωπικό, ακόμα και καθ' όλη τη διάρκεια του εικοσιτετραώρου, με τρόπο που μεταβάλλεται από ημέρα σε ημέρα, με βάση μία πληθώρα παραγόντων.

Στην παρούσα ενότητα, θα αναλυθεί το συγκεκριμένο πρόβλημα, θα εξεταστεί ο αντικειμενικός του στόχος, θα μελετηθεί η υπάρχουσα βιβλιογραφία και θα επιχειρηθεί η κατάστρωση μίας λύσης του.

2.3.1 Περιγραφή του προβλήματος

Ως “Πρόβλημα Βέλτιστου Χρονοπρογραμματισμού Εργασίας”, γνωστό στη διεθνή βιβλιογραφία ως “Nurse Scheduling Problem”, ορίζεται το πρόβλημα βελτιστοποίησης, το οποίο έχει στόχο την εύρεση του βέλτιστου ωρολογίου προγράμματος εργασίας των εργαζομένων σε μία επιχείρηση. Το βέλτιστο αυτό ωρολόγιο πρόγραμμα καλύπτει πλήρως τις ανάγκες της επιχείρησης σε προσωπικό, χωρίς να απασχολεί περισσότερους υπαλλήλους από όσους πραγματικά χρειάζονται, ενώ ταυτόχρονα ικανοποιεί τις προτιμήσεις κάθε εργαζόμενου, στον βαθμό που αυτό είναι δυνατό.

Ο χρονοπρογραμματισμός εργασίας είναι ένα σύνθετο πρόβλημα, που χρειάζεται να επιλύεται σε τακτική βάση από τον υπεύθυνο προσωπικού της επιχείρησης, συνήθως ανά εβδομάδα ή μήνα. Οι στόχοι του είναι συνήθως πολλαπλοί και αντιφατικοί, όπως για παράδειγμα η ελαχιστοποίηση του κόστους για την επιχείρηση και η μεγιστοποίηση της ικανοποίησης των εργαζομένων (Legrain, Bouarab, και Lahrichi 2014).

Μία συνήθης τακτική κατάστρωσης του ωρολογίου προγράμματος είναι το κυκλικό πρόγραμμα (cyclic roster), σύμφωνα με το οποίο η ανάθεση του προσωπικού σε βάρδιες⁵ επαναλαμβάνεται εκ περιτροπής ανά δεδομένο χρονικό διάστημα (Rosenblom και Goertzen 1987). Αν και ο συγκεκριμένος τύπος προγράμματος εξασφαλίζει τη δικαιοσύνη στην κατανομή των βαρδιών μεταξύ των εργαζομένων, εντούτοις δε λαμβάνει υπόψη τις προτιμήσεις του καθενός, αφού στηρίζεται στην επαναληψιμότητα (Choy και Cheong 2012). Αντίθετα, τα μη κυκλικά προγράμματα (non-cyclic rosters) δεν επαναλαμβάνονται, αλλά δημιουργούνται κάθε φορά εκ νέου, λαμβάνοντας υπόψη όλους τους περιορισμούς που έχουν τεθεί.

Το πρόβλημα βρίσκει εφαρμογή σε οποιαδήποτε επιχείρηση οι εργαζόμενοι δουλεύουν με βάρδιες. Το παράδειγμα ενός νοσοκομείου ή μίας κλινικής προτιμάται συνήθως από τους ερευνητές, λόγω των

⁵Εργασία κατά βάρδιες ορίζεται ως η μέθοδος οργάνωσης της ομαδικής εργασίας, κατά την οποία οι εργαζόμενοι διαδέχονται ο ένας τον άλλο στις ίδιες θέσεις εργασίας με ορισμένο ρυθμό, περιλαμβανομένου του ρυθμού περιτροπής. Αυτή μπορεί να είναι συνεχής ή όχι, επομένως οι εργαζόμενοι ενδέχεται να επιτελούν μία εργασία σε διαφορετικές ώρες, σε μία δεδομένη περίοδο ημερών ή εβδομάδων (άρθρο 2 Π.Δ. 88/1999) (Κέντρο Πληροφόρησης Εργαζομένων και Ανέργων, χ.χ.).

κρίσιμων περιορισμών που θέτει το επάγγελμα. Ωστόσο, αυτό μπορεί να επεκταθεί σε τουριστικά καταλύματα, εργοστάσια, εστιατόρια, υπηρεσίες εξυπηρέτησης πελατών κ.λπ. Στα πλαίσια της παρούσας εργασίας, η μελέτη θα επικεντρωθεί γύρω από το παράδειγμα ενός ξενοδοχείου.

2.3.2 Θεωρητική μελέτη

Το πρόβλημα του βέλτιστου χρονοπρογραμματισμού εργασίας έχει μελετηθεί εκτενώς και έχει αποδειχθεί ότι πρόκειται για πρόβλημα πολυπλοκότητας NP-hard (Jafari και Salmasi 2015). Στο (Van den Bergh κ.ά. 2013) γίνεται αναλυτική παρουσίαση του προβλήματος και των διαφόρων λύσεων που έχουν προταθεί. Τεχνικές που χρησιμοποιούνται περιλαμβάνουν μεταξύ άλλων γραμμικό (linear), ακέραιο (integer) και μικτό προγραμματισμό (mixed integer programming) (Jaumard, Semet, και Vovor 1998), προγραμματισμό στόχων (goal programming) (Berrada, Ferland, και Michelon 1996) και προγραμματισμό περιορισμών (constraint programming) (Trilling, Guinet, και Magny 2006). Άλλες έρευνες κάνουν χρήση μετευρετικών μεθόδων (meta-heuristic methods), οπως γενετικούς αλγορίθμους (genetic algorithms) (Aickelin και Dowsland 2004), αναζήτηση ταμπού (tabu search) (Valouxis και Housos 2000) κ.λπ.

Στην παρούσα εργασία, θα δοθεί έμφαση στη δημιουργία μη κυκλικών προγραμμάτων, με χρήση ακέραιου προγραμματισμού (integer programming). Η διαδικασία αυτή υπόκειται σε διάφορους περιορισμούς. Οι περιορισμοί αυτοί ονομάζονται ισχυροί (hard constraints), με την έννοια ότι εάν κάποιος από αυτούς παραβιάζεται, ότι το αποτέλεσμα δεν αποτελεί εφικτή λύση του προβλήματος. Αρχικά, πρέπει να καλύπτεται ο ελάχιστος αριθμός προσωπικού σε κάθε βάρδια για την ομάδα. Ο αριθμός αυτός δεν είναι απαραίτητα σταθερός, αλλά μεταβάλλεται ανάλογα με την ημέρα και τη βάρδια. Επιπλέον, πρέπει να ικανοποιούνται οι όροι που αντιστοιχούν στον φόρτο εργασίας κάθε εργαζόμενου, όπως καθορίζεται από τη σύμβαση εργασίας. Αυτό συνήθως περιλαμβάνει το πλήθος των βαρδιών ή την κατηγορία τους (για παράδειγμα, νυχτερινές βάρδιες). Ακόμη, η νομοθεσία ορίζει περιορισμούς για κάθε εργαζόμενο, όπως τον μέγιστο αριθμό συνεχόμενων ημερών εργασίας ή η ελάχιστη χρονική διάρκεια μεταξύ δύο βαρδιών.

Παράλληλα με τους ισχυρούς περιορισμούς, το πρόβλημα χαρακτηρίζεται επίσης από τους χαλαρούς περιορισμούς (soft constraints), οι οποίοι επιτρέπεται να παραβιαστούν από μία λύση του, αν και δεν είναι επιθυμητό. Σε αυτούς συγκαταλέγονται, για παράδειγμα, αιτήσεις του προσωπικού για ημέρες άδειας, προτιμήσεις βάρδιες ή προτιμήσεις μεταξύ συναδέλφων. Σε μοντέλα γραμμικού προγραμματισμού, οι χαλαροί περιορισμοί εκφράζονται συνήθως με τη μορφή ποινών στην αντικειμενική συνάρτηση (Choy και Cheong 2012). Επίσης, μπορούμε να εκφράσουμε την ποιότητα μίας εφικτής λύσης, με τον βαθμό στον οποίο αυτή παραβιάζει τους χαλαρούς περιορισμούς (Kumar, Nagalakshmi, και Kumaraguru 2014).

2.3.3 Εισαγωγή στον γραμμικό προγραμματισμό

Στην παράγραφο αυτή, θα οριστούν οι βασικές έννοιες του γραμμικού προγραμματισμού, με σκοπό την αξιοποίησή τους στην παράγραφο 2.3.4. Ο ενδιαφερόμενος αναγνώστης μπορεί να ανατρέξει σε σχετικά συγγράμματα, όπως το (Σίσκος 2000).

Ο γραμμικός προγραμματισμός είναι ένα μαθηματικό μοντέλο, με στόχο τη βέλτιστη προσέγγιση προβλημάτων κατανομής περιορισμένων πόρων ή μέσων σε εναλλακτικές και ανταγωνιστικές μεταξύ τους δραστηριότητες κατά τον καλύτερο δυνατό τρόπο. Είναι ίσως το δημοφιλέστερο μοντέλο στον χώρο της επιχειρησιακής έρευνας και της διοικητικής επιστήμης.

Με τον όρο βέλτιστοποίηση, εννοούμε τη μεγιστοποίηση ή ελαχιστοποίηση αγνώστων πραγματικών μεταβλητών (μεταβλητές απόφασης) που περιέχονται σε μία ή περισσότερες γραμμικές συναρτήσεις (κριτήρια βέλτιστοποίησης) και οριοθετούνται από γραμμικούς περιορισμούς (εξισώσεις και ανισώσεις). Το κριτήριο με το οποίο πραγματοποιείται η βέλτιστοποίηση ονομάζεται αντικειμενική συνάρτηση.

Εκφράζοντας μαθηματικά τα παραπάνω, ένα πρόβλημα γραμμικού προγραμματισμού έχει στόχο την εύρεση των τιμών των μεταβλητών απόφασης x_1, x_2, \dots, x_l έτσι, ώστε να βέλτιστοποιείται (μεγιστοποιείται ή ελαχιστοποιείται) η αντικειμενική συνάρτηση:

$$z = c_1 x_1 + c_2 x_2 + \dots + c_l x_l$$

υπό τους γραμμικούς περιορισμούς:

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \dots + a_{1l}x_l &\leq \hat{\eta} = \hat{\eta} \geq b_1 \\ a_{21}x_1 + a_{22}x_2 + \dots + a_{2l}x_l &\leq \hat{\eta} = \hat{\eta} \geq b_2 \\ &\vdots \\ a_{m1}x_1 + a_{m2}x_2 + \dots + a_{ml}x_l &\leq \hat{\eta} = \hat{\eta} \geq b_m \\ x_1 \geq 0, x_2 \geq 0, \dots, x_l \geq 0 \end{aligned}$$

όπου $c_{ij}, i = 1, 2, \dots, m$ και $j = 1, 2, \dots, l$ είναι γνωστοί πραγματικοί συντελεστές.

Συνηθίζεται το παραπάνω πρόβλημα να γράφεται συνοπτικά, με τη βοήθεια μητρών, ως εξής:

Να βρεθεί το διάνυσμα \mathbf{x} για το οποίο έχουμε:

$$[\max] \quad \hat{\eta} \quad [\min] \quad \mathbf{z} = \mathbf{c}^T \mathbf{x}$$

υπό τους περιορισμούς:

$$A\mathbf{x} \leq \mathbf{b}$$

$$\mathbf{x} \geq 0$$

όπου A είναι η μήτρα των συντελεστών a_{ij} και έχει διαστάσεις $m \times n$, ενώ $\mathbf{b}, \mathbf{c}, \mathbf{x}$ είναι διανύσματα διαστάσεων $m \times 1, l \times 1, l \times 1$ αντίστοιχα.

2.3.4 Σχηματισμός του μοντέλου

Θα καταστρώσουμε τώρα τις εξισώσεις που σχηματίζουν το μοντέλο ακέραιου προγραμματισμού.

Σταθερές και μεταβλητές

Θα χρειαστούμε αρχικά κάποιες σταθερές για το πρόβλημα. Έστω ότι θα σχηματίσουμε το ωρολόγιο πρόγραμμα για έναν ορίζοντα D ημερών (π.χ. δύο εβδομάδες). Έστω ότι κάθε ημέρα χωρίζεται σε S βάρδιες (π.χ. τρεις, πρωινή, απογευματινή και νυχτερινή). Μπορούμε με όμοιο τρόπο να χωρίσουμε κάθε ημέρα σε διαφορετικό πλήθος βάρδιών, ωστόσο χωρίς βλάβη της γενικότητας, θα υποθέσουμε ότι κάθε ημέρα χωρίζεται στις ίδιες S βάρδιες. Έστω επίσης ότι η επιχείρηση στελεχώνεται από N εργαζόμενους.

Οι μεταβλητές απόφασης που θα χρησιμοποιήσουμε είναι οι εξής:

$$X_{d,s,n}$$

όπου d είναι μία ημέρα που ανήκει στον ορίζοντα προγραμματισμού, δηλαδή $d \in D$, s είναι μία βάρδια της ημέρας d , δηλαδή $s \in S$ (ή $s \in S_d$, αν κάθε ημέρα χωρίζεται σε διαφορετικές βάρδιες) και n είναι ένας εργαζόμενος της επιχείρησης, δηλαδή $n \in N$. Η κάθε μεταβλητή απόφασης είναι δυαδική και παίρνει τις τιμές 0 και 1, σύμφωνα με τον ακόλουθο κανόνα:

$$X_{d,s,n} = \begin{cases} 0 & \text{ο εργαζόμενος } n \text{ δεν δουλεύει στη βάρδια } s \text{ της ημέρας } d \\ 1 & \text{ο εργαζόμενος } n \text{ δουλεύει στη βάρδια } s \text{ της ημέρας } d \end{cases}$$

Ο κάθε εργαζόμενος μπορεί να έχει τη δική του σειρά προτίμησης για τις βάρδιες κάθε ημέρας του ορίζοντα προγραμματισμού. Μπορούμε να μοντελοποιήσουμε αυτή τη σειρά προτίμησης, αντιστοιχίζοντας κάθε βάρδια της ημέρας με έναν ακέραιο συντελεστή:

$$P_{d,s,n} \in \{-1, 1, \dots, |S|\}$$

2.3 Το πρόβλημα του Βέλτιστου Χρονοπρογραμματισμού Εργασίας

όπου $|S|$ είναι το πλήθος των βαρδιών S και $P_{d,s,n}$ είναι ο συντελεστής προτίμησης του εργαζόμενου n για τη βάρδια s της ημέρας d . Ο συντελεστής $P_{d,s,n}$ μπορεί να πάρει και την αρνητική τιμή -1 , ώστε να εκφράσει με τη μορφή χαλαρού περιορισμού (ή, ισοδύναμα, κόστους στην αντικειμενική συνάρτηση) τη δυσαρέσκεια του συγκεκριμένου εργαζόμενου για τη συγκεκριμένη βάρδια.

Ο κάθε εργαζόμενος έχει δικαίωμα να αιτηθεί άδειες κάποιες ημέρες του ορίζοντα προγραμματισμού. Μπορούμε να μοντελοποιήσουμε αυτή την αίτηση, με την ακόλουθη μεταβλητή:

$$L_{d,n} = \begin{cases} 0 & \text{ο εργαζόμενος } n \text{ έχει αιτηθεί άδεια την ημέρα } d \\ 1 & \text{ο εργαζόμενος } n \text{ δεν έχει αιτηθεί άδεια την ημέρα } d \end{cases}$$

Η κάθε βάρδια της κάθε ημέρας του ορίζοντα προγραμματισμού έχει διαφορετικές απαιτήσεις σε προσωπικό. Μπορούμε να μοντελοποιήσουμε αυτή τη ζήτηση, με την ακόλουθη μεταβλητή:

$$D_{d,s}$$

όπου $D_{d,s}$ είναι ένας ακέραιος αριθμός που εκφράζει το πόσοι εργαζόμενοι πρέπει να εργάζονται στη βάρδια s της ημέρας d .

Αντικειμενική συνάρτηση

Η αντικειμενική συνάρτηση μπορεί να εκφραστεί είτε ως συνάρτηση των προτιμήσεων των εργαζομένων —και άρα ο αντικειμενικός σκοπός να είναι η μεγιστοποίησή της—, είτε ως συνάρτηση του κόστους της επιχείρησης —και άρα ο αντικειμενικός σκοπός να είναι η ελαχιστοποίησή της—. Θα εξετάσουμε το πρόβλημα μεγιστοποίησης της χρησιμότητας.

$$\max Z = \sum_d \sum_s \sum_n P_{d,s,n} X_{d,s,n}$$

Αθροίζουμε τα γινόμενα της κάθε μεταβλητής απόφασης με τον αντίστοιχο συντελεστή προτίμησης του εργαζόμενου, για κάθε ημέρα του ορίζοντα προγραμματισμού, κάθε βάρδια και κάθε εργαζόμενο.

Περιορισμοί

Θα προχωρήσουμε τώρα με την έκφραση των περιορισμών του προβλήματος.

Κάθε εργαζόμενος πρέπει να αντιστοιχίζεται το πολύ σε μία βάρδια κάθε ημέρα, δηλαδή:

$$\sum_s X_{d,s,n} \leq 1, \forall n, d$$

Εάν ένας εργαζόμενος έχει αιτηθεί άδεια μία συγκεκριμένη ημέρα, δε θα πρέπει να αντιστοιχιστεί σε καμία βάρδια της συγκεκριμένης ημέρας, δηλαδή:

$$\sum_s X_{d,s,n} \leq L_{d,n}, \forall n, d$$

Κάθε εργαζόμενος πρέπει να εργάζεται το πολύ $MaxS$ ημέρες στον ορίζοντα προγραμματισμού (έστω 5 ημέρες την εβδομάδα), δηλαδή:

$$\sum_d \sum_s X_{d,s,n} \leq MaxS, \forall n$$

Μετά από $MaxCD$ συνεχόμενες ημέρες εργασίας, κάθε εργαζόμενος δικαιούται μία ημέρα ξεκούρασης, δηλαδή:

$$\sum_{d=i}^{i+MaxCD-1} \sum_s X_{d,s,n} \leq MaxCD, \forall n, i = \{1, \dots, MaxCD - 1\}$$

Κάθε εργαζόμενος απαγορεύεται να αντιστοιχίζεται σε μία πρωινή βάρδια, εάν δούλευε την αμέσως προηγούμενη νυχτερινή, δηλαδή:

$$X_{i,s_{Night},n} + X_{i+1,s_{Morning},n} \leq 1, \forall n, i = \{1, \dots, |D| - 1\}$$

όπου s_{Night} και $s_{Morning}$ η νυχτερινή και η πρωινή βάρδια αντίστοιχα.

Τέλος, η κάθε βάρδια θα πρέπει να στελεχώνεται από τον κατάλληλο αριθμό εργαζομένων, δηλαδή:

$$\sum_n X_{d,s,n} = D_{d,s}, \forall d, s$$

Προτιμήσεις συναδέλφων

Συχνά, στο εργασιακό περιβάλλον αναπτύσσονται σχέσεις φιλίας, με αποτέλεσμα κάποιοι εργαζόμενοι να προτιμούν να δουλεύουν μαζί με κάποιον συνάδελφό τους σε μία βάρδια. Ομοίως, μπορεί να συμβαίνει και το αντίθετο, δηλαδή ένος εργαζόμενος να μην επιθυμεί να δουλεύει μαζί με κάποιον συγκεκριμένο. Αυτές είναι δύο παράμετροι που μπορούν να ενσωματωθούν στο μοντέλο.

Προκειμένου να εισαχθεί η έννοια της κοινής εκτέλεσης μίας βάρδιας από δύο εργαζόμενους, θα πρέπει να συνδυάσουμε δύο μεταβλητές απόφασης με το λογικό “και” (logical AND). Μία πρώτη προσέγγιση, ωστόσο λανθασμένη, θα ήταν ο πολλαπλασιασμός των δύο αντίστοιχων μεταβλητών

2.3 Το πρόβλημα του Βέλτιστου Χρονοπρογραμματισμού Εργασίας

απόφασης, οπότε το γινόμενό τους θα ήταν 1, αν και οι δύο εργαζόμενοι εκτελούν μαζί την ίδια βάρδια και 0 σε κάθε άλλη περίπτωση. Η λύση αυτή όμως είναι λανθασμένη, γιατί ο πολλαπλασιασμός δύο μεταβλητών απόφασης κάνει το μοντέλο μη γραμμικό (Brown και Dell 2007).

Για τον λόγο αυτό, ορίζουμε επιπλέον μεταβλητές απόφασης, μία για κάθε ζεύγος εργαζομένων. Καθώς αυτό μπορεί να οδηγήσει σε σπατάλη μνήμης όταν οι καθορισμένες προτιμήσεις είναι λίγες (ως προς το σύνολο των εργαζομένων) –συνολικά θα χρειάζονταν $N^2 \times S \times D$ τέτοιες μεταβλητές–, θα ορίσουμε μόνο όσες χρειάζονται, μία για κάθε προτίμηση κάθε εργαζόμενου, θετική ή αρνητική, για κάθε βάρδια, κάθε ημέρας:

$$Y_{d,s,n_1,n_2}$$

όπου d είναι μία ημέρα που ανήκει στον ορίζοντα προγραμματισμού, δηλαδή $d \in D$, s είναι μία βάρδια της ημέρας d , δηλαδή $s \in S$ (ή $s \in S_d$, αν κάθε ημέρα χωρίζεται σε διαφορετικές βάρδιες) και n_1, n_2 είναι δύο εργαζόμενοι της επιχείρησης, δηλαδή $n_1, n_2 \in N$. Αυτές οι μεταβλητές απόφασης είναι επίσης δυαδικές και παίρνουν τις τιμές 0 και 1, σύμφωνα με τον ακόλουθο κανόνα:

$$Y_{d,s,n_1,n_2} = \begin{cases} 0 & \text{οι εργαζόμενοι } n_1 \text{ και } n_2 \text{ δεν δουλεύουν μαζί στη βάρδια } s \text{ της ημέρας } d \\ 1 & \text{οι εργαζόμενοι } n_1 \text{ και } n_2 \text{ δουλεύουν μαζί στη βάρδια } s \text{ της ημέρας } d \end{cases}$$

Θα πρέπει βέβαια να ορίσουμε και τους –ισχυρούς–περιορισμούς που επιβάλλουν την ακόλουθη σχέση λογικού “και” μεταξύ των τριών μεταβλητών:

$$Y_{d,s,n_1,n_2} = X_{d,s,n_1} \wedge X_{d,s,n_2}$$

Οι περιορισμοί αυτοί είναι οι ακόλουθοι (Brown και Dell 2007):

$$\begin{aligned} Y_{d,s,n_1,n_2} &\leq X_{d,s,n_1} \\ Y_{d,s,n_1,n_2} &\leq X_{d,s,n_2} \\ Y_{d,s,n_1,n_2} &\geq X_{d,s,n_1} + X_{d,s,n_2} - 1 \end{aligned}$$

Οι περιορισμοί προτίμησης συναδέλφων ανήκουν στην κατηγορία των χαλαρών περιορισμών, αφού εκφράζουν προτίμηση και μπορούν να παραβιαστούν από την εφικτή λύση, αν και δεν είναι προτιμητέο. Γι' αυτό, θα προσαρτηθούν στην αντικειμενική συνάρτηση ως εξής:

$$\max Z = \sum_d \sum_s \sum_n P_{d,s,n} X_{d,s,n} + \sum_d \sum_s \sum_n \sum_{n_{\text{liked}}} Y_{d,s,n,n_{\text{liked}}} + \sum_d \sum_s \sum_n \sum_{n_{\text{disliked}}} Y_{d,s,n,n_{\text{disliked}}}$$

2.3 Το πρόβλημα του Βέλτιστου Χρονοπρογραμματισμού Εργασίας

δηλαδή η κοινή αντιστοίχιση ενός εργαζομένου με έναν προσφιλή του αντιστοιχεί σε αντικειμενικό όφελος, ενώ η κοινή αντιστοίχιση με έναν μη προσφιλή του αντιστοιχεί σε αντικειμενικό κόστος.

3 Σχεδίαση της εφαρμογής

3.1 Ανάλυση προβλήματος και καθορισμός απαιτήσεων

Κατά τη φάση της σχεδίασης ενός προϊόντος, γενικότερα, ή μίας εφαρμογής, ειδικότερα, είναι υψηλής σημασίας η ανάλυση του προβλήματος που αυτή καλείται να λύσει και η καταγραφή των απαιτήσεων που αυτή θα πρέπει να ικανοποιεί. Οι απαιτήσεις, που διακρίνονται σε λειτουργικές και μη λειτουργικές, αποτελούν σημείο αναφοράς, τόσο για τον σχεδιαστή, όσο και για τον μηχανικό λογισμικού, οι οποίοι πρέπει να ανατρέχουν σε αυτές σε κάθε φάση του κύκλου ζωής του προϊόντος τους, ώστε να βεβαιώνονται ότι αυτό καλύπτει τις προσδοκίες του πελάτη.

Η ανάλυση του προβλήματος που καλείται να επιλύσει η εφαρμογή **Horario** και, κατ' επέκταση, η ανάλυση των απαιτήσεών της, χωρίζεται, σύμφωνα με τις αρχές του ανθρωποκεντρικού σχεδιασμού, σε δύο στάδια:

- την **ανάλυση χρηστών (user analysis)**, κατά την οποία μελετώνται όχι μόνο όσοι πρόκειται να χρησιμοποιούν την εφαρμογή σε τακτική βάση (τυπικοί χρήστες), αλλά και γενικότερα όσοι πρόκειται να επωφεληθούν άμεσα ή έμμεσα από αυτή (ενδιαφερόμενοι-stakeholders) και
- την **ανάλυση εργασιών (task analysis)**, κατά την οποία μελετώνται οι εργασίες που εκτελούν οι χρήστες κατά τη διάρκεια χρήσης της εφαρμογής, ώστε να επιτύχουν έναν δεδομένο στόχο.

3.1.1 Ανάλυση χρηστών

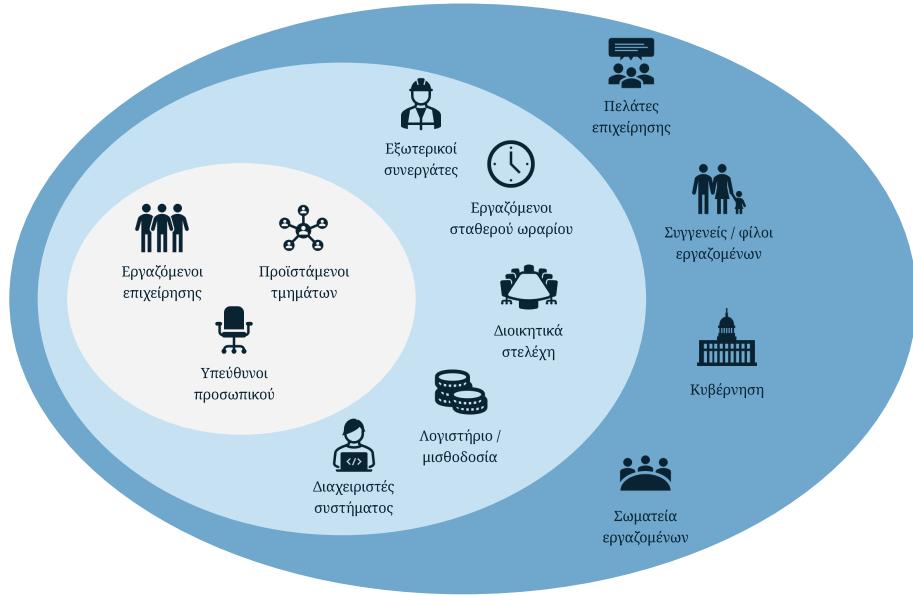
Σύμφωνα με τους (Αβιούρης κ.ά. 2018) και (McGrath 2021), οι χρήστες μίας εφαρμογής διακρίνονται σε πρωτεύοντες, δευτερεύοντες και τριτεύοντες.

- **Πρωτεύοντες** ονομάζονται οι χρήστες, οι οποίοι αλληλεπιδρούν συχνά με την εφαρμογή και έχουν τη μεγαλύτερη επιρροή σε αυτή. Η συμμετοχή των πρωτευόντων χρηστών είναι συνήθως υποχρεωτική για την ολοκλήρωση του σκοπού της εφαρμογής.
 - *Παραδείγματα πρωτευόντων χρηστών: εργαζόμενοι επιχείρησης, προϊστάμενοι τμημάτων, υπεύθυνοι προσωπικού.*

3.1 Ανάλυση προβλήματος και καθορισμός απαιτήσεων

- Δευτερεύοντες ονομάζονται οι χρήστες, οι οποίοι αλληλεπιδρούν με την εφαρμογή σπανιότερα ή μέσω ενός ενδιάμεσου προσώπου. Η αποδοχή και η συμφωνία των δευτερεύοντων χρηστών είναι συνήθως υποχρεωτική για την ολοκλήρωση του σκοπού της εφαρμογής.
 - Παραδείγματα δευτερεύοντων χρηστών: εργαζόμενοι σταθερού ωραρίου, διοικητικά στελέχη, υπεύθυνοι λογιστηρίου και μισθοδοσίας, εξωτερικοί συνεργάτες, διαχειριστές συστημάτων.
- Τριτεύοντες ονομάζονται οι χρήστες, οι οποίοι δε χρησιμοποιούν οι ίδιοι την εφαρμογή, αλλά επηρεάζονται έμμεσα από τη χρήση και τα αποτελέσματά της.
 - Παραδείγματα τριτεύοντων χρηστών: πελάτες επιχείρησης, συγγενείς και φίλοι εργαζομένων, σωματεία εργαζομένων, κυβέρνηση.

Στην εικόνα 3.1 αναπαρίστανται σχηματικά όλοι οι ενδιαφερόμενοι της εφαρμογής Horario, καταταγμένοι στις τρεις προαναφερθείσες κατηγορίες χρηστών. Οι πρωτεύοντες χρήστες βρίσκονται στο εσωτερικό επίπεδο και οι τριτεύοντες στο εξωτερικό.



Εικόνα 3.1. Σχηματική αναπαράσταση των ενδιαφερομένων της εφαρμογής Horario

3.1.2 Το πλαίσιο PACT

Το πλαίσιο PACT (PACT framework) είναι ένα μοντέλο σκέψης, στο οποίο βασίζεται σε μεγάλο βαθμό ο ανθρωποκεντρικός σχεδιασμός. Οι τέσσερις πυλώνες του είναι (Batagoda 2018):

- People (άνθρωποι): κατανόηση των ανθρώπων που θα αλληλεπιδράσουν με το σύστημα. Η διαφορετικότητα των ανθρώπων θα πρέπει να λαμβάνεται υπόψη κατά τη διάρκεια της

σχεδίασης ενός συστήματος. Παράμετροι που θα πρέπει να ληφθούν υπόψη είναι οι φυσικές διαφορές (φυσικά χαρακτηριστικά και αισθήσεις, διαταραχές στην επικοινωνία κ.λπ.), οι φυσιολογικές διαφορές (ικανότητα γρήγορης εύρεσης πληροφοριών, κατανόηση σύνθετων συστημάτων) και τα νοητικά μοντέλα, δηλαδή η εντύπωση που τους δημιουργείται για τον τρόπο που λειτουργεί ένα σύστημα, αφού αλληλεπιδράσουν για λίγο με αυτό.

- **Activities (δραστηριότητες):** κατανόηση των χαρακτηριστικών των εργασιών που θα πρέπει να εκτελούνται από την εφαρμογή. Αυτά περιλαμβάνουν χρονικά χαρακτηριστικά (πόσο συχνή είναι η εργασία), συνεργατικότητα, ασφάλεια κ.λπ.
- **Contexts (πλαίσια χρήσης):** κατανόηση των συνθηκών κάτω από τις οποίες γίνεται η εκτέλεση μίας εργασίας. Το συγκείμενο αυτό διαδραματίζει σημαντικό ρόλο, στον τρόπο με τον οποίο η εργασία εν τέλει θα εκτελεστεί.
- **Technologies (τεχνολογίες):** γνώση των διαθέσιμων τεχνολογιών, οι οποίες θα χρησιμοποιηθούν για την εκτέλεση της εργασίας. Οι τεχνολογίες αυτές καθορίζουν τον τρόπο εκτέλεσης και μπορεί να διευκολύνουν ή δυσκολέψουν την ολοκλήρωση της εργασίας.

Με βάση αυτά, έχουμε το ακόλουθο πλαίσιο PACT για την εφαρμογή Horario:

- **People:**

- Εργαζόμενοι επιχείρησης
 - Εξωτερικοί συνεργάτες
 - Διοικητικά στελέχη
 - Πελάτες και άλλοι ενδιαφερόμενοι που επηρεάζονται έμμεσα

Περισσότερες λεπτομέρειες σχετικά με τους χρήστες της εφαρμογής βρίσκονται στην Ενότητα [3.1.1](#).

- **Activities:**

- Προβολή επερχόμενων βαρδιών και δημιουργία προγράμματος εργασίας για τις διαφορετικές ομάδες της επιχείρησης
 - Διαχείριση αιτημάτων των χρηστών για απουσία από την εργασία
 - Παροχή χρήσιμων πληροφοριών σχετικά με την εργασία

- **Contexts:**

- Συσκευές χρήσης (υπολογιστής, κινητό τηλέφωνο, ταμπλέτα)
 - Μέγεθος επιχείρησης (πλήθος ομάδων, εργαζομένων)
 - Πολυπλοκότητα προτιμήσεων εργαζομένων

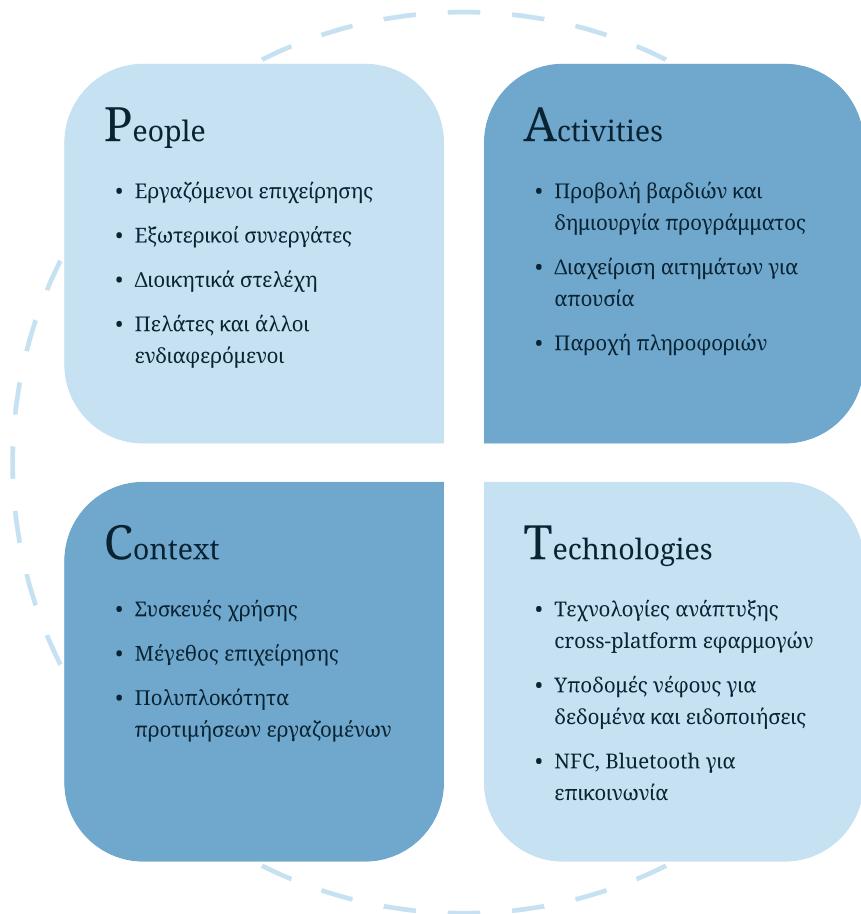
- **Technologies:**

- Τεχνολογίες / frameworks ανάπτυξης cross-platform εφαρμογών (π.χ. Flutter)

3.1 Ανάλυση προβλήματος και καθορισμός απαιτήσεων

- Υποδομές νέφους για την υποστήριξη της λειτουργικότητας της εφαρμογής και της παροχής ειδοποιήσεων σε πραγματικό χρόνο (real-time notifications)
- Τεχνολογία NFC και Bluetooth, για την επικοινωνία μεταξύ κοντινών συσκευών

Στην Εικόνα 3.2 αναπαρίστανται σχηματικά οι επιμέρους πυλώνες του πλαισίου PACT για την εφαρμογή Horario.



Εικόνα 3.2. Σχηματική αναπαράσταση του πλαισίου PACT για την εφαρμογή Horario

3.1.3 Περσόνες και σενάρια χρήσης

Η έννοια της περσόνας (*persona*) στη σχεδίαση διαδραστικών συστημάτων εισήχθη το 1999 από τον Alan Cooper. Αφορά την ανάπτυξη εικονικών χαρακτήρων, που αναπαριστούν τους χρήστες-στόχους της εφαρμογής, δηλαδή τους ανθρώπους για τους οποίους σχεδιάζεται το σύστημα.

Οι περσόνες είναι ένα πολύτιμο εργαλείο στον ανθρωποκεντρικό σχεδιασμό, αφού μπορούν να χρησιμοποιηθούν ως μία απλή και κατανοητή γλώσσα επικοινωνίας μεταξύ ενδιαφερομένων,

3.1 Ανάλυση προβλήματος και καθορισμός απαιτήσεων

σχεδιαστών και προγραμματιστών. Επιπλέον, επιτρέπουν στους σχεδιαστές να επικεντρωθούν σε ένα συγκεκριμένο σύνολο πραγματικών χρηστών, παίρνοντας συνήθως έτσι πιο σωστές σχεδιαστικές αποφάσεις. Πέρα από αυτά, όταν η εφαρμογή πρόκειται για ένα εμπορικό προϊόν, οι περσόνες συνήθως προσφέρουν και χρήσμες πληροφορίες που αφορούν το μάρκετινγκ και τις μελλοντικές πωλήσεις (Qiany 2020).

Για την ανάλυση απαιτήσεων της εφαρμογής Horario, δημιουργούμε έναν μικρόκοσμο, στον οποίο μελετάμε την ιστορία ενός εικονικού ξενοδοχείου σε ένα προάστιο της Αθήνας, το Starlight Hotel. Το Starlight Hotel αποτελείται από τέσσερις ομάδες, την υποδοχή (Front desk), το εστιατόριο (Restaurant), την υπηρεσία καθαρισμού (Housekeepers) και το τμήμα πωλήσεων (Sales). Η κάθε ομάδα στελεχώνεται από τον κατάλληλο αριθμό εργαζομένων, ωστόσο υπάρχουν εργαζόμενοι που ανήκουν σε περισσότερες από μία ομάδες (για παράδειγμα, στην υποδοχή και στο τμήμα πωλήσεων).



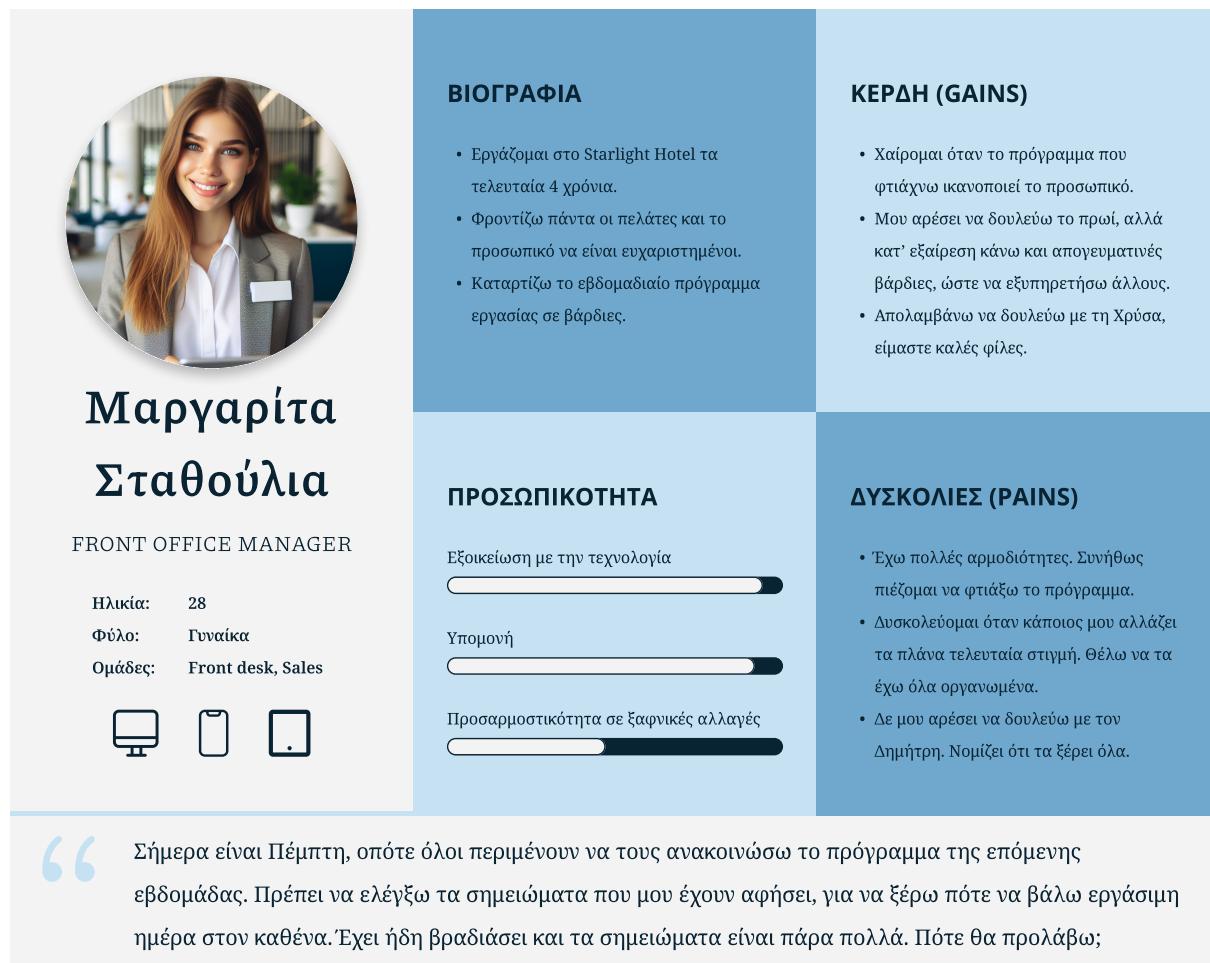
Εικόνα 3.3. Το εικονικό ξενοδοχείο Starlight Hotel

Επιλέγουμε κάποιους από τους εργαζομένους του Starlight Hotel και με βάση αυτούς, δημιουργούμε τέσσερις περσόνες. Η κάθε περσόνα αντιπροσωπεύει έναν τυπικό χρήστη της εφαρμογής, έτσι προσπαθούμε κάθε περσόνα να αντιπροσωπεύει έναν διαφορετικό συνδυασμό παραμέτρων του πλαισίου PACT (πρόσωπο, δραστηριότητα, πλαίσιο χρήσης, τεχνολογία). Για κάθε μία περσόνα, σκιαγραφούμε τις δυσκολίες (pains) και τους στόχους (gains), με βάση τα οποία αυτή δρα και, επιπλέον, καταστρώνουμε ένα σενάριο χρήσης της εφαρμογής.

3.1 Ανάλυση προβλήματος και καθορισμός απαιτήσεων

Περσόνα 1: Μαργαρίτα Σταθούλια

Η Μαργαρίτα είναι μία 28χρονη γυναίκα, που εργάζεται στην υποδοχή του Starlight Hotel τα τελευταία 4 χρόνια και σήμερα κατέχει τη θέση της Front Office Manager. Οι αρμοδιότητές της περιλαμβάνουν μεταξύ άλλων την ενεργή συμμετοχή στην υποδοχή του ξενοδοχείου (καλωσόρισμα και εξυπηρέτηση πελατών), την οργάνωση της υποδοχής και τον χειρισμό έκτακτων περιστατικών. Σημαντικά καθήκοντά της είναι η διαχείριση του προσωπικού της υποδοχής, η κατάρτιση του εβδομαδιαίου προγράμματος εργασίας σε βάρδιες και η επίβλεψη της ομαλής λειτουργίας των υπόλοιπων ομάδων του ξενοδοχείου, ερχόμενη σε επαφή με τους αντίστοιχους προϊστάμενους (managers). Πολλές φορές νιώθει ότι όλοι τα περιμένουν όλα από εκείνη, αλλά αυτό είναι που της αρέσει στη δουλειά της, ακόμα κι αν κουράζεται πολύ (Εικόνα 3.4).



Εικόνα 3.4. Περσόνα 1: Μαργαρίτα Σταθούλια

Περσόνα 2: Ιωσηφίνα Νικοπολίδη

Η Ιωσηφίνα είναι μία 30χρονη γυναίκα, που εργάζεται επίσης στην υποδοχή του Starlight Hotel τον τελευταίο χρόνο. Είναι νέα μητέρα, γι' αυτό και υποβάλλει συχνά αιτήματα για ρεπό, ώστε να φροντίσει την κόρη της (Εικόνα 3.5).



Εικόνα 3.5. Περσόνα 2: Ιωσηφίνα Νικοπολίδη

Περσόνα 3: Αγγελική Ηλιοπούλου

Η Αγγελική είναι μία 53χρονη γυναίκα, που εργάζεται στο Starlight Hotel από τότε που ιδρύθηκε και σήμερα είναι προϊσταμένη της ομάδας καθαρισμού. Ο ρόλος της είναι τόσο η συμμετοχή, όσο και η επίβλεψη της ομάδας. Ο ρόλος της είναι αρκετά πολύπλοκος, γιατί η ομάδα της απαρτίζεται από πολλά άτομα, λίγα από τα οποία έχουν σταθερά ωράρια. Συχνά, μάλιστα, δημιουργούνται αψιμαχίες στην ομάδα, τις οποίες η ίδια καλείται να επιλύσει (Εικόνα 3.6).

3.1 Ανάλυση προβλήματος και καθορισμός απαιτήσεων



**Αγγελική
Ηλιοπούλου**

HOUSEKEEPING MANAGER

Ηλικία: 53
Φύλο: Γυναίκα
Ομάδες: Housekeepers



ΒΙΟΓΡΑΦΙΑ

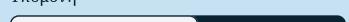
- Εργάζομαι στο Starlight Hotel από τη στιγμή που ιδρύθηκε.
- Φροντίζω το ξενοδοχείο να είναι πάντα καθαρό και οι πελάτες να έχουν ό,τι ζητήσουν.
- Συχνά χρειάζεται να επεμβαίνω στους υφισταμένους μουν.

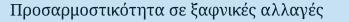
ΚΕΡΔΗ (GAINS)

- Οι πελάτες δίνουν συχνά φιλοδωρήματα, ιδίως όταν είναι ικανοποιημένοι από εμάς.
- Μου αρέσει να βοηθώ τη Μαργαρίτα στην κατάρτιση του προγράμματος. Συνήθως για τη δική μου ομάδα, το φτιάχνω εγώ και της το δίνω έτοιμο.

ΠΡΟΣΩΠΙΚΟΤΗΤΑ

Εξουκείωση με την τεχνολογία 

Υπομονή 

Προσαρμοστικότητα σε ξαφνικές αλλαγές 

ΔΥΣΚΟΛΙΕΣ (PAINS)

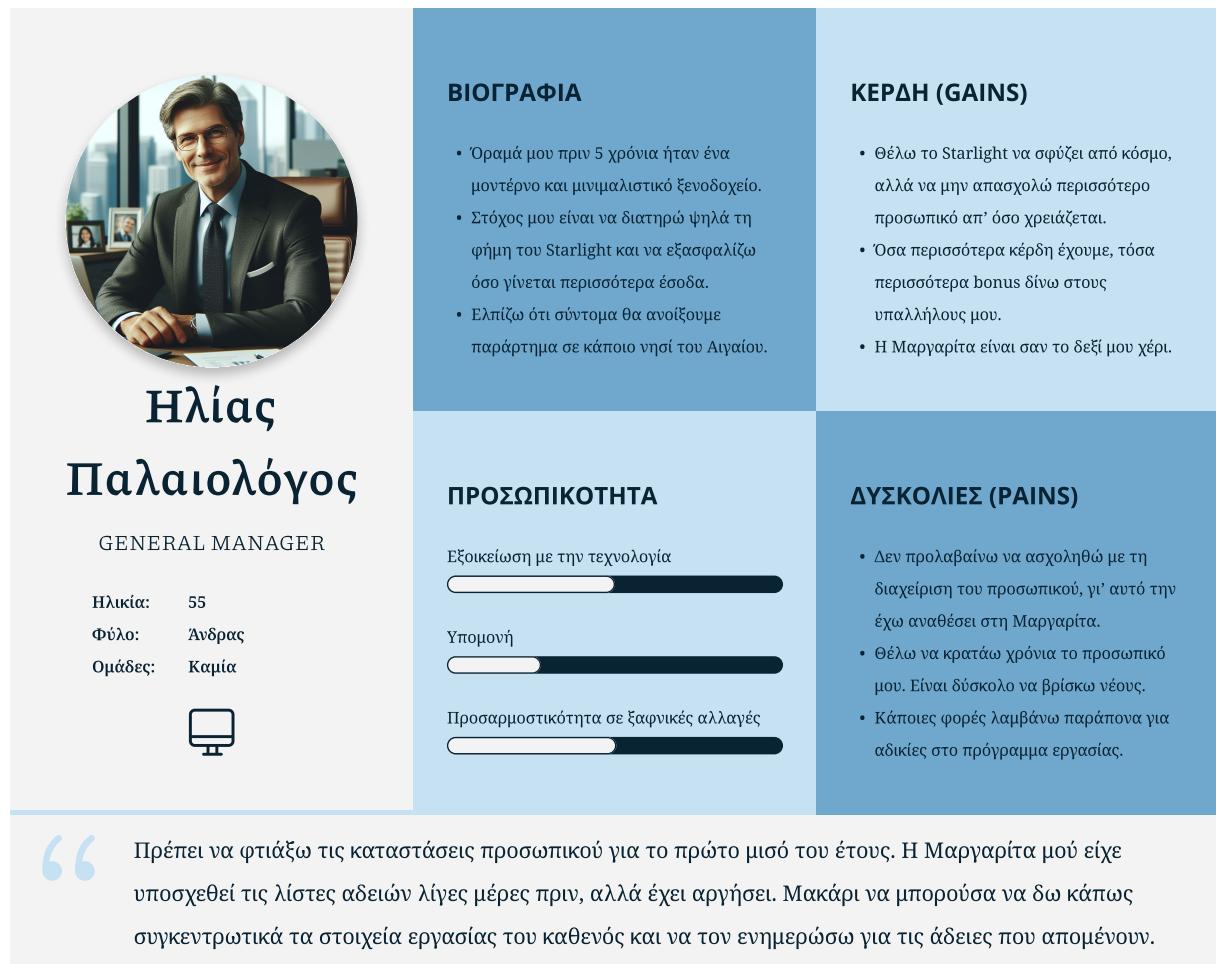
- Δυσκολεύομαι να θυμάμαι ποιος δουλεύει και πότε. Φοβάμαι ότι κάποιοι δουλεύουν λιγότερες ώρες απ' όσο πρέπει.
- Συχνά οι υφιστάμενοι μου ανταλλάσσουν μεταξύ τους βάρδιες, αλλά εγώ αργώ να το μάθω.

“ Μόλις με ενημέρωσε η Μαργαρίτα, ότι μεθαύριο, το Σάββατο, θα έχουμε μία δεξιώση γάμου. Πρέπει αμέσως να αυξήσω τις βάρδιες, γιατί το ξενοδοχείο θα είναι γεμάτο από κόσμο. Επί τη ευκαιρία, ας ελέγξω ποιοι δουλεψφαν χθες το πρωί, γιατί δεχτήκαμε παράπονα από έναν πελάτη.

Εικόνα 3.6. Περσόνα 3: Αγγελική Ηλιοπούλου

Περσόνα 4: Ηλίας Παλαιολόγος

Ο Ηλίας είναι ο 55χρονος ιδιοκτήτης και Γενικός Διευθυντής του Starlight Hotel. Ο ρόλος του είναι περισσότερο διοικητικός. Δεν έρχεται συχνά σε επαφή με το υπόλοιπο προσωπικό, παρά μόνο με τη Μαργαρίτα, την οποία θεωρεί το δεξί του χέρι και συμβουλεύεται συχνά για θέματα του προσωπικού. Τα καθήκοντά του περιλαμβάνουν μεταξύ άλλων τη γραφειοκρατική διαχείριση του ανθρωπίνου δυναμικού και την επικοινωνία με το λογιστήριο, για τις πληρωμές και τα οφέλη του προσωπικού (Εικόνα 3.7).



Εικόνα 3.7. Περσόνα 4: Ηλίας Παλαιολόγος

3.1.4 Ιστορίες χρηστών

Οι **ιστορίες χρηστών** (user stories) είναι μία ανεπίσημη, γενικευμένη έκφραση μίας λειτουργίας ενός λογισμικού, όπως εκλαμβάνεται από την οπτική του τελικού χρήστη ή πελάτη. Ο σκοπός της ιστορίας

3.1 Ανάλυση προβλήματος και καθορισμός απαιτήσεων

χρήστη είναι να περιγράψει πώς ένα τμήμα του λογισμικού θα προσφέρει μία συγκεκριμένη αξία στον χρήστη (Rehkopf, χ.χ.). Συχνά, οι ιστορίες χρηστών έχουν τη μορφή “Εγώ, ο/η …, ως …, θέλω να …, ώστε …”.

Παρακάτω αναλύονται συνοπτικά οι ιστορίες χρηστών, με την ελάχιστη δυνατή λεπτομέρεια, όπως εξάγονται από τις περσόνες της Ενότητας {#sec:design:analysis:personas}.

1. Η Ιωσηφίνα, ως υπάλληλος, θέλει να βλέπει εύκολα τι ώρα δουλεύει στην υποδοχή κάθε ημέρα.
2. Η Ιωσηφίνα, ως υπάλληλος, θέλει να λαμβάνει ειδοποίηση κάθε φορά που γίνονται αλλαγές στο πρόγραμμα.
3. Η Ιωσηφίνα, ως υπάλληλος, θέλει να ζητήσει ανταλλαγή βάρδιας με μία συνάδελφό της, για προσωπικούς λόγους.
4. Η Αγγελική, ως προϊσταμένη, θέλει να δημιουργεί και να επεξεργάζεται προγράμματα εργασίας με drag-and-drop, ώστε να ανταποκρίνεται γρήγορα σε απρόπτα συμβάντα.
5. Η Αγγελική, ως προϊσταμένη, θέλει να προσθέσει μία επιπλέον βάρδια μία συγκεκριμένη ημέρα, τροποποιώντας υπάρχον πρόγραμμα.
6. Ο Ηλίας, ως διευθυντής, θέλει να φτιάξει μία κατάσταση με τις ώρες που έχει εργαστεί ο καθένας.
7. Ο Ηλίας, ως διευθυντής, θέλει να στείλει αυτόματα ένα email σε όλους τους εργαζόμενους, ώστε να τους ενημερώσει σχετικά με τις υπολειπόμενες άδειες.
8. Ο Ηλίας, ως διευθυντής, θέλει να προσθέτει και να αφαιρεί χρήστες από την εφαρμογή, αλλά και να επεξεργάζεται τα δικαιώματά τους, ώστε να ανταποκρίνεται σε προσλήψεις, αποχωρήσεις και προαγωγές.
9. Η Μαργαρίτα, ως υπάλληλος, θέλει να ζητήσει άδεια σε ένα μελλοντικό χρονικό διάστημα.
10. Η Μαργαρίτα, ως υπάλληλος, θέλει να χτυπάει κάρτα με το κινητό της γρήγορα, χωρίς να χρειάζεται να το ξεκλειδώνει.
11. Η Μαργαρίτα, ως προϊσταμένη, θέλει να δει πόσοι υφιστάμενοί της έχουν ζητήσει ρεπό ή άδεια, ώστε να τα αποδεχτεί ή να τα απορρίψει.

3.1.5 Πίνακες απαιτήσεων

Με βάση τα σενάρια χρήσης για τις περσόνες που καταστρώθηκαν παραπάνω, αλλά και τις ιστορίες χρηστών, όπως επεξεργάστηκαν, κατασκευάζουμε πίνακες απαιτήσεων για την εφαρμογή, λειτουργικών και μη λειτουργικών. Οι λειτουργικές απαιτήσεις (functional requirements) περιγράφουν τις απαιτούμενες εργασίες, ενέργειες και δραστηριότητες που η εφαρμογή πρέπει να εκτελεί και περιλαμβάνουν τεχνικές λεπτομέρειες, υπολογισμούς και μεθόδους επεξεργασίας δεδομένων (Leonard 1999). Αντίθετα, οι μη λειτουργικές απαιτήσεις (non-functional requirements) (ή, εναλλακτικά, ποιοτικές απαιτήσεις) εισάγουν περιορισμούς στη σχεδίαση ή στην υλοποίηση, όπως απαιτήσεις απόδοσης, ασφάλειας και αξιοπιστίας.

3.1 Ανάλυση προβλήματος και καθορισμός απαιτήσεων

Στον Πίνακα 3.1, καταγράφονται οι βασικές λειτουργικές απαιτήσεις της εφαρμογής Horario, με σειρά προτεραιότητας και την εκτιμώμενη διάρκεια υλοποίησης.

Πίνακας 3.1: Πίνακας λειτουργικών απαιτήσεων

α/α	Περιγραφή λειτουργικής απαίτησης	Προτεραιότητα	Εκτιμώμενη διάρκεια
1	Προβολή ατομικού προγράμματος εργασίας, σε μορφή ατζέντας, όπου απεικονίζονται η θέση, η ημερομηνία, η ώρα και οι συνάδελφοι	Υψηλή	1-2 ημέρες
2	Προβολή ομαδικού προγράμματος εργασίας, σε μορφή ημερολογίου, όπου απεικονίζονται συνοπτικά οι εργαζόμενοι σε κάθε βάρδια	Υψηλή	2-3 ημέρες
3	Προβολή προσωπικών αιτημάτων για άδεια, ρεπό ή ανταλλαγή βάρδιας και της κατάστασης ελέγχου τους	Υψηλή	1-2 ημέρες
4	Προβολή αιτημάτων ομάδας που αναμένουν έλεγχο και δυνατότητα αποδοχής ή απόρριψή τους	Υψηλή	1-2 ημέρες
5	Δημιουργία νέου προγράμματος εργασίας για συγκεκριμένη ομάδα, σε καθορισμένο, μελλοντικό χρονικό διάστημα, με χειροκίνητη ανάθεση προσωπικού σε βάρδιες	Υψηλή	2-3 ημέρες
6	Επεξεργασία υπάρχοντος προγράμματος εργασίας, για προσθήκη ή αφαίρεση συγκεκριμένων βαρδιών, ή τροποποίηση προσωπικού ή ώρας βάρδιας	Υψηλή	1-2 ημέρες
7	Δημιουργία νέου προσωπικού αιτήματος για άδεια ή ρεπό ή ανταλλαγή βάρδιας	Υψηλή	1-2 ημέρες
8	Επεξεργασία αποθηκευμένων εργαζομένων επιχείρησης (προσθήκη, αφαίρεση, τροποποίηση)	Υψηλή	2-3 ημέρες

3.1 Ανάλυση προβλήματος και καθορισμός απαιτήσεων

<i>α/α</i>	Περιγραφή λειτουργικής απαίτησης	Προτεραιότητα	Εκτιμώμενη διάρκεια
9	Δημιουργία νέου προγράμματος εργασίας για συγκεκριμένη ομάδα, σε καθορισμένο, μελλοντικό χρονικό διάστημα, με αυτόματη ανάθεση προσωπικού σε βάρδιες	Μέτρια	10-15 ημέρες
10	Δημιουργία νέου αιτήματος για άδεια ή ρεπό ή ανταλλαγή βάρδιας, για λογαριασμό άλλου εργαζόμενου, ως προϊστάμενος ή διευθυντής	Μέτρια	1-2 ημέρες
11	Επεξεργασία αποθηκευμένων δεδομένων επιχείρησης (προσθήκη, αφαίρεση, τροποποίηση ομάδων, τοποθεσιών κ.λπ.)	Μέτρια	2-3 ημέρες
12	Καθορισμός περιορισμών προσωπικού σε βάρδιες, για χρήση από τον αλγόριθμο αυτόματης ανάθεσης προσωπικού σε βάρδιες	Μέτρια	1-2 ημέρες
13	Καθορισμός προτιμήσεων προσωπικού σε βάρδιες και συναδέλφους, για χρήση από τον αλγόριθμο αυτόματης ανάθεσης προσωπικού σε βάρδιες	Μέτρια	1-2 ημέρες
14	Εξαγωγή δεδομένων και στατιστικών από παρελθοντικά προγράμματα εργασίας	Χαμηλή	3-4 ημέρες
15	Παρακολούθηση υπολειπόμενων αδειών και αυτόματη ενημέρωση του προσωπικού	Χαμηλή	2-3 ημέρες
16	Χρονική παρακολούθηση εργασίας προσωπικού (ψηφιακή κάρτα, καταγραφή διαλειμμάτων και ωρών εργασίας)	Χαμηλή	3-4 ημέρες

Στον Πίνακα 3.2, καταγράφονται οι βασικές μη λειτουργικές απαιτήσεις της εφαρμογής Horario, με σειρά προτεραιότητας.

Πίνακας 3.2: Πίνακας μη λειτουργικών απαιτήσεων

α/α	Περιγραφή μη λειτουργικής απαίτησης	Προτεραιότητα
1	Φιλική προς τον χρήστη παρουσίαση προγράμματος εργασίας	Υψηλή
2	Εύκολη και διαισθητική διαδικασία δημιουργίας προγράμματος εργασίας, σε μορφή οδηγού	Υψηλή
3	Εύκολη και διαισθητική διαδικασία δημιουργίας νέου αιτήματος, σε μορφή οδηγού	Υψηλή
4	Λήψη ειδοποιήσεων για σημαντικά συμβάντα	Υψηλή
5	Απρόσκοπη λειτουργία εφαρμογής, με προβλέψιμα αποτελέσματα της κάθε ενέργειας, με σκοπό την καλή εμπειρία χρήστη	Υψηλή

3.1.6 Ανάλυση εργασιών: Εθνογραφία και Ιεραρχική ανάλυση εργασιών

Το τελευταίο βήμα της ανάλυσης που ακολουθήθηκε είναι η ανάλυση εργασιών (task analysis). Η πρώτη μέθοδος που ακολουθήθηκε είναι η **εθνογραφία** (ethnography), με διερεύνηση εντός **πλαισίου** (contextual inquiry). Η μέθοδος αυτή συνίσταται στην παρατήρηση των χρηστών στον φυσικό τους χώρο, χωρίς παρέμβαση του ερευνητή. Με αυτόν τον τρόπο, μπορούν να γίνουν αντιληπτές οι ανάγκες των χρηστών, οι στόχοι τους και οι μέθοδοι που ακολουθούν ώστε να τους επιτύχουν. Συγκεκριμένα, η διερεύνηση εντός πλαισίου (Beyer και Holtzblatt 1997) είναι μία εθνογραφική μέθοδος σχεδιασμού, στην οποία ο ερευνητής κατ' αρχήν παρατηρεί τους χρήστες στο φυσικό τους περιβάλλον. Σε μία σχέση δασκάλου - μαθητή, οι χρήστες εξηγούν στον ερευνητή τα καθήκοντά τους και, τελικά, ερμηνεύουν από κοινού τις παρατηρήσεις που έγιναν. Η διερεύνηση εντός πλαισίου περιλαμβάνει επίσης ανοιχτού τύπου συνεντεύξεις με τους χρήστες, στις οποίες ο ερευνητής πρέπει να εστιάζει σε συγκεκριμένους στόχους ή ερωτήσεις, αλλά να είναι ευέλικτος σε οποιαδήποτε άλλη πληροφορία επιθυμούν να προσθέσουν οι χρήστες (Αβούρης κ.ά. 2018).

Στα πλαίσια ανάλυσης της εφαρμογής **Horario**, παρατηρήσαμε τυπικές ημέρες των εργαζομένων ενός ξενοδοχείου σε διάρκεια μίας εβδομάδας και καταγράψαμε τις ενέργειές τους, που αφορούν το ζήτημα του χρονοπρογραμματισμού της εργασίας. Συγκεκριμένα, παρατηρήσαμε τις ακόλουθες ενέργειες:

- Ο υπεύθυνος του ξενοδοχείου καταρτίζει το πρόγραμμα εργασίας σε βάθος μίας εβδομάδας, το αργότερο μέχρι την προηγούμενη Πέμπτη.
- Οι εργαζόμενοι του ξενοδοχείου πραγματοποιούν αιτήματα στον υπεύθυνο, τα οποία κυρίως

3.1 Ανάλυση προβλήματος και καθορισμός απαιτήσεων

αφορούν τη μη ανάθεσή τους σε συγκεκριμένες βάρδιες (π.χ. Δε θέλω να δουλεύω την επόμενη Τρίτη το απόγευμα, Θα ήθελα ρεπό το επόμενο Σαββατοκύριακο).

- Ο υπεύθυνος του ξενοδοχείου ενδέχεται να αναθέσει τον χρονοπρογραμματισμό επιμέρους ομάδων σε υφισταμένους του (π.χ. το πρόγραμμα των σερβιτόρων του εστιατορίου στον υπεύθυνο εστιατορίου).
- Υπάρχουν περιορισμοί που καθορίζουν την ανάθεση συγκεκριμένων εργαζομένων σε βάρδιες, καθώς και συμβάσεις που ορίζουν αυστηρά τις αναθέσεις (π.χ. ένας εργαζόμενος ενδέχεται να έχει προσληφθεί αποκλειστικά ώστε να εκτελεί νυχτερινές βάρδιες, ή ένας εργαζόμενος να έχει σταθερό ωράριο 9:00 π.μ. - 5:00 μ.μ.).
- Υπάρχουν προτιμήσεις των εργαζομένων για συγκεκριμένες βάρδιες της εβδομάδας (π.χ. Προτιμώ να δουλεύω Κυριακή πρωί, Δε μου αρέσει να δουλεύω απογεύματα καθημερινών).
- Υπάρχουν προτιμήσεις των εργαζομένων για συναδέλφους (π.χ. Προτιμώ να δουλεύω με έναν συγκεκριμένο συνάδελφο, Δε συμπαθώ τον τάδε).
- Με βάση όλα τα παραπάνω, αλλά και τις ανάγκες στελέχωσης του ξενοδοχείου, ο υπεύθυνος προσπαθεί, με βάση πρωτίστως την εμπειρία του, αλλά και κυρίως με τη μέθοδο δοκιμής και σφάλματος (trial-and-error) να δημιουργήσει το βέλτιστο πρόγραμμα εργασίας, που καλύπτει όλα τα αιτήματα του προσωπικού και ικανοποιεί κατά το δυνατόν τις προτιμήσεις του.

Στη συνέχεια, εφαρμόσαμε την τεχνική της **ιεραρχικής ανάλυσης εργασιών** (*hierarchical task analysis*). Σύμφωνα με την τεχνική αυτή, επιλέγουμε τον **στόχο** (*goal*) ενός χρήστη της εφαρμογής μας και αναλύουμε τις διαφορετικές **εργασίες** (*tasks*) που πρέπει να εκτελεστούν, προκειμένου να ολοκληρωθεί ο στόχος. Την ίδια διαδικασία επαναλαμβάνουμε διαδοχικά για κάθε μία εργασία, έως ότου φτάσουμε στο επίπεδο στοιχειωδών, μη περαιτέρω διασπάσιμων ενεργειών. Έτσι, καταλήγουμε σε μία δενδροειδή δομή βημάτων. Ο τρόπος διαπέρασης του δένδρου αυτού, ως διαδικασία ολοκλήρωσης του στόχου, μπορεί να καθοδηγείται από περιορισμούς και συνθήκες, οι οποίες ονομάζονται **πλάνα** (*plans*) (Αβούρης κ.ά. 2018).

Στα πλαίσια ανάλυσης της εφαρμογής **Horario**, εφαρμόζουμε ιεραρχική ανάλυση εργασιών για τρεις βασικούς στόχους της εφαρμογής: τη δημιουργία ενός νέου προγράμματος εργασίας, την προβολή των επερχόμενων βαρδιών και την δημιουργία ενός νέου αιτήματος από έναν εργαζόμενο.

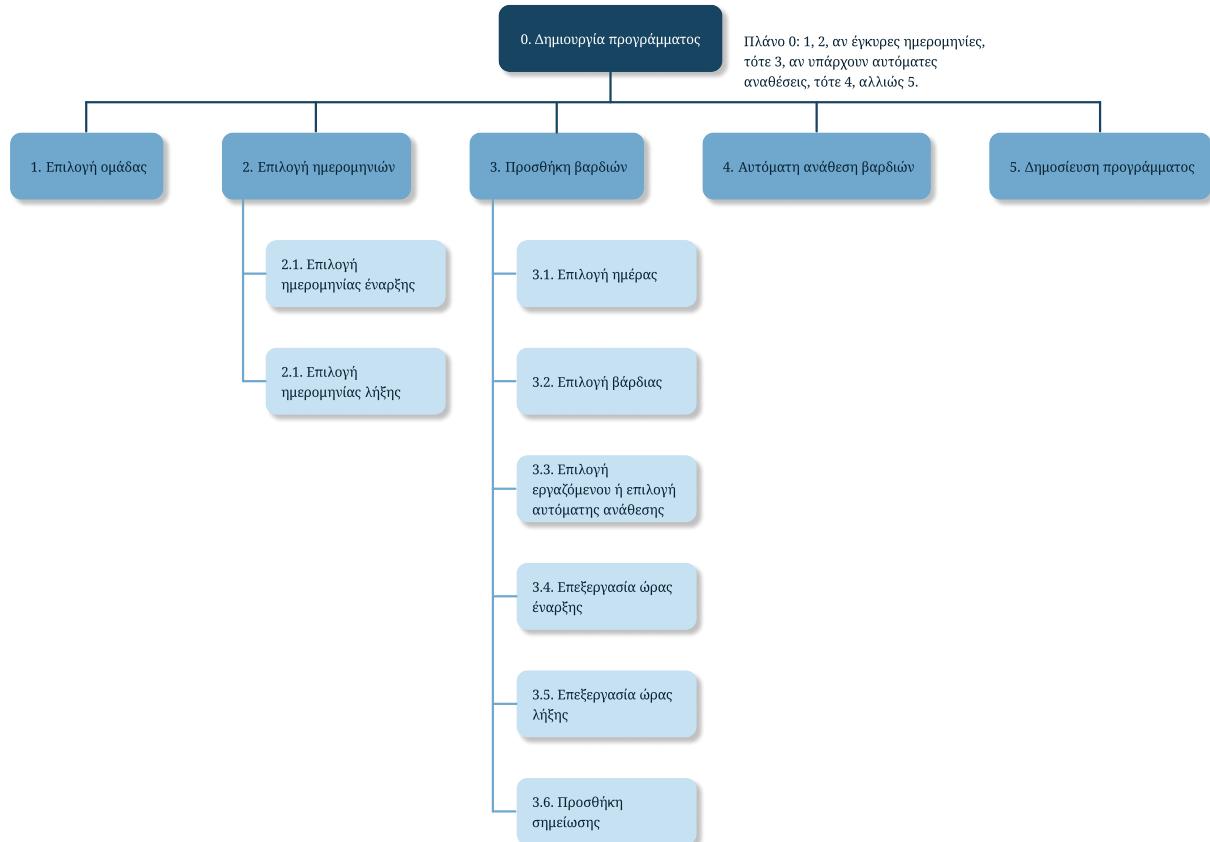
Στόχος 1: Δημιουργία προγράμματος εργασίας

Δημιουργούμε το δένδρο ιεραρχικής ανάλυσης εργασιών που φαίνεται στην Εικόνα 3.8. Στην κορυφή του δένδρου βρίσκεται ο στόχος, που είναι η αγορά εισιτηρίων. Ο στόχος διασπάται σε 5 επιμέρους εργασίες υψηλού επιπέδου: την επιλογή ομάδας, την επιλογή ημερομηνίων, την προσθήκη βαρδιών, την αυτόματη ανάθεση βαρδιών και τη δημοσίευση του προγράμματος. Η εκτέλεση αυτών των εργασιών καθορίζεται από το πλάνο 0, το οποίο ορίζει την εκτέλεση των 2 πρώτων βημάτων, την ενεργοποίηση του τρίτου, αν οι ημερομηνίες είναι έγκυρες, έπειτα την εκτέλεση του τέταρτου, μόνο

3.1 Ανάλυση προβλήματος και καθορισμός απαιτήσεων

εάν έχει ζητηθεί τουλάχιστον μία αυτόματη ανάθεση βάρδιας σε εργαζόμενο και, τέλος, την εκτέλεση του πέμπτου.

Με όμοιο τρόπο γίνεται η διάσπαση των υψηλού επιπέδου εργασιών σε εργασίες χαμηλότερου επιπέδου και παρόμοια πλάνα καθορίζουν την εκτέλεσή τους, με σκοπό την ολοκλήρωση του στόχου.



Εικόνα 3.8. Ιεραρχική ανάλυση εργασιών για τη δημιουργία προγράμματος εργασίας

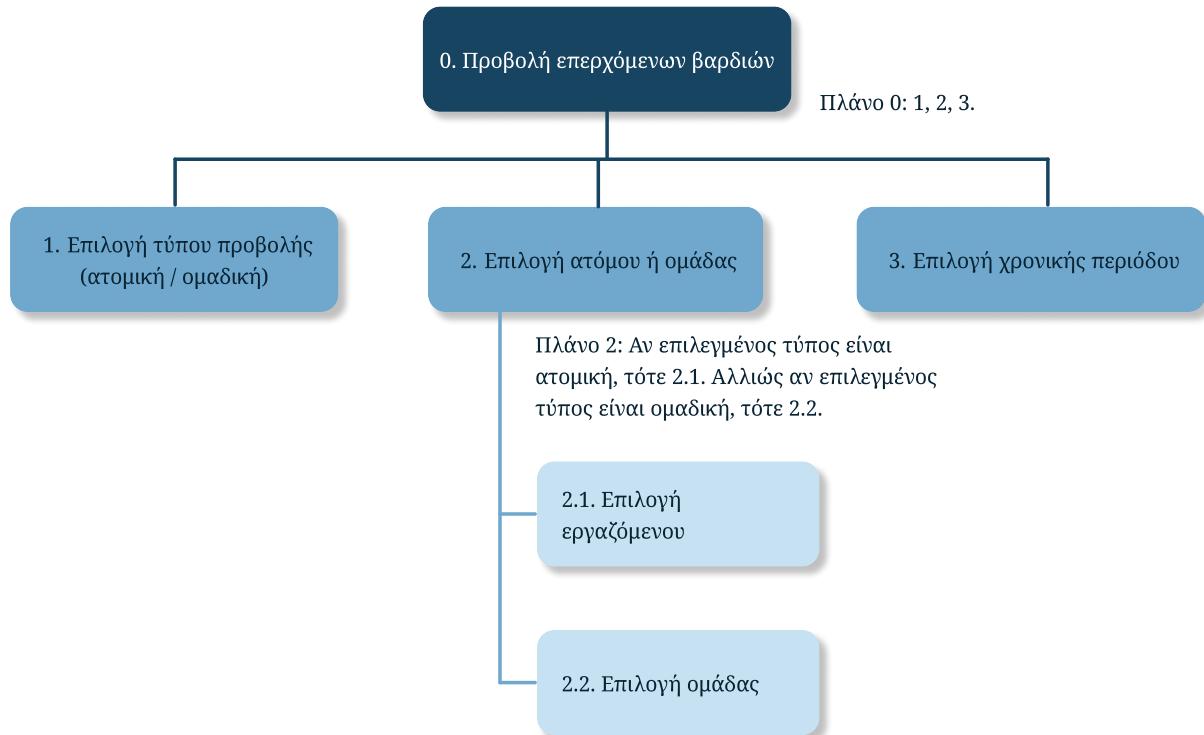
Στόχος 2: Προβολή επερχόμενων βαρδιών

Στην Εικόνα 3.9 φαίνεται το αντίστοιχο δένδρο ιεραρχικής ανάλυσης εργασιών για τον στόχο της προβολής επερχόμενων βαρδιών.

Στόχος 3: Δημιουργία αιτήματος από εργαζόμενο

Στην Εικόνα 3.10 φαίνεται το αντίστοιχο δένδρο ιεραρχικής ανάλυσης εργασιών για τον στόχο της δημιουργίας αιτήματος από εργαζόμενο.

3.2 Μελέτη ανταγωνιστικών λύσεων



Εικόνα 3.9. Ιεραρχική ανάλυση εργασιών για την προβολή επερχόμενων βαρδιών

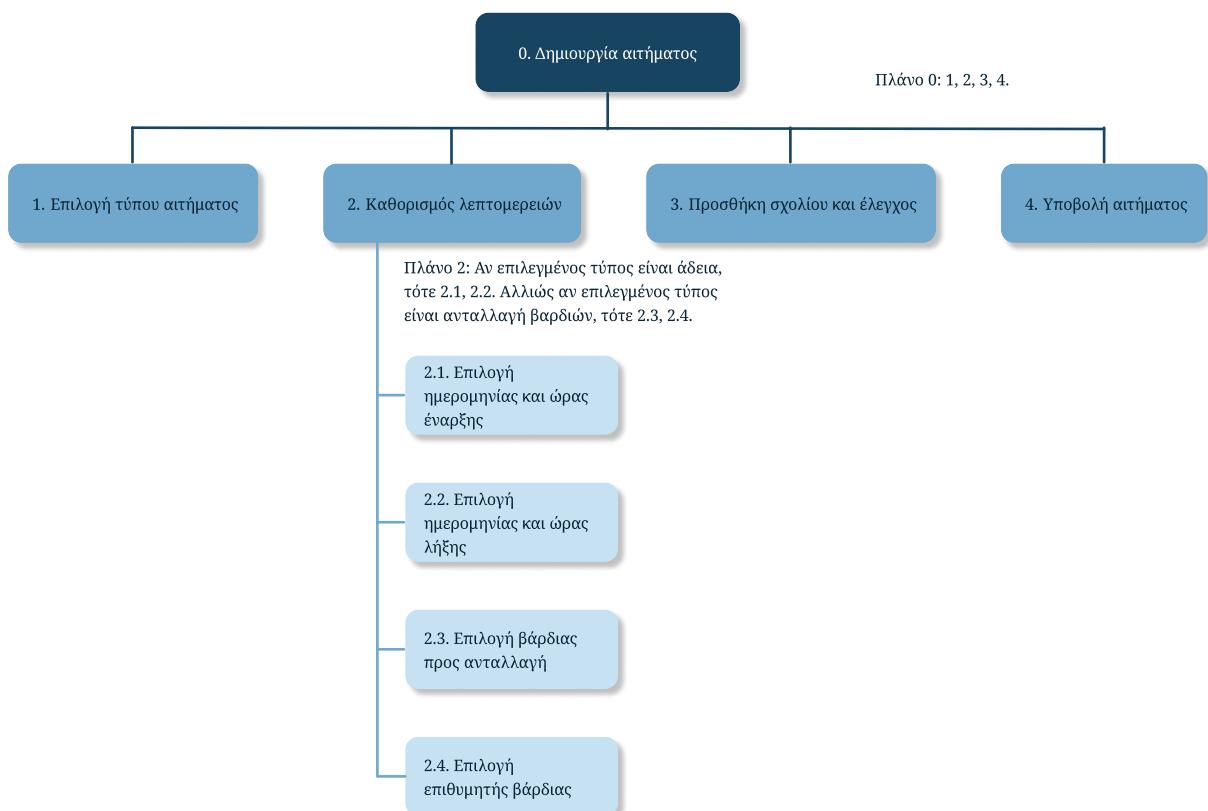
3.2 Μελέτη ανταγωνιστικών λύσεων

Έχοντας ολοκληρώσει την ανάλυση των απαιτήσεων της εφαρμογής μας και προτού ξεκινήσουμε τη σχεδίαση της διεπαφής, μελετήσαμε υπάρχουσες λύσεις για τη διαχείριση του προγράμματος εργασίας του προσωπικού σε μία επιχείρηση. Για κάθε μία από τις εφαρμογές αυτές, εξετάσαμε ποιες λειτουργίες προσφέρουν στους χρήστες, καθώς και πώς οργανώνονται τα επιμέρους βήματα των λειτουργιών αυτών. Σταθήκαμε ιδιαίτερα στην οπτική παρουσίαση του περιεχομένου και στις ενέργειες που καλούνται να εκτελέσουν οι χρήστες. Τελικά, συνδυάσαμε στοιχεία από τις υπάρχουσες εφαρμογές, επιλέγοντας αυτά που κρίναμε πως βρίσκονται πιο κοντά στο νοητικό μοντέλο των χρηστών. Με τον τρόπο αυτό, προσπαθούμε η εκμάθηση του συστήματος από τους χρήστες να είναι όσο το δυνατόν πιο ομαλή.

3.2.1 When I Work

To [When I Work](#) είναι μία εφαρμογή διαχείρισης προσωπικού, για επιχειρήσεις στις οποίες η εργασία οργανώνεται κατά βάρδιες. Οι λειτουργίες που υποστηρίζει περιλαμβάνουν:

- τον χρονοπρογραμματισμό εργασίας για τους υπαλλήλους,



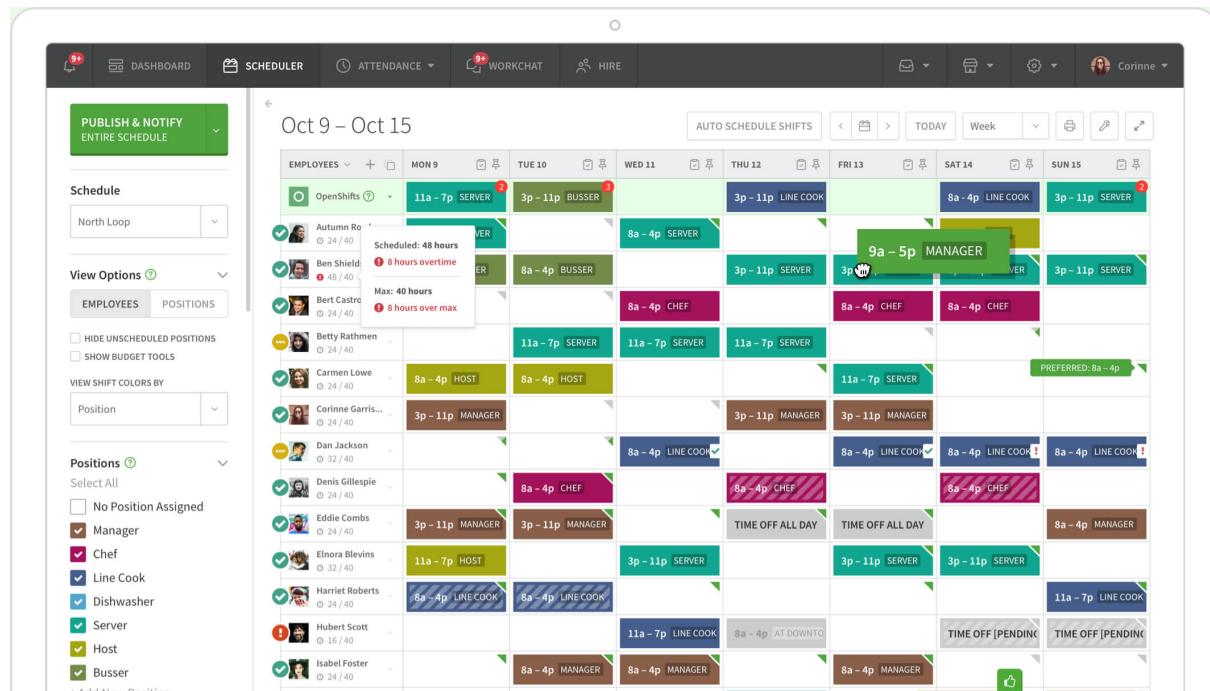
Εικόνα 3.10. Ιεραρχική ανάλυση εργασιών για τη δημιουργία αιτήματος από εργαζόμενο

3.2 Μελέτη ανταγωνιστικών λύσεων

- την παρακολούθηση της εργασίας (ψηφιακή κάρτα), καταγράφοντας ώρες εργασίας, διαλείμματα και αργίες,
- την επικοινωνία μεταξύ εργαζομένων σε μία πλατφόρμα μηνυμάτων και
- την υποστήριξη πληρωμών για τους εργαζόμενους.

Οι σχεδιαστές του When I Work έχουν δώσει μεγάλη έμφαση στην εύκολη ρύθμιση της εφαρμογής, ώστε να τεθεί σε λειτουργία από την πρώτη μέρα υιοθέτησης της από μία επιχείρηση. Ταυτόχρονα, εστιάζουν στην αυτοματοποίηση των λειτουργιών, χωρίς να χρειάζεται η παρέμβαση ανθρώπου, μειώνοντας τον χρόνο που αφιερώνουν στον χρονοπρογραμματισμό οι προϊστάμενοι.

Η εφαρμογή διαθέτει online έκδοση, προσβάσιμη από έναν περιηγητή (browser), αλλά και εκδόσεις που εγκαθίστανται σε συσκευές Android και iOS.



Εικόνα 3.11. Η online έκδοση της εφαρμογής When I Work (πηγή: wheniwork.com)

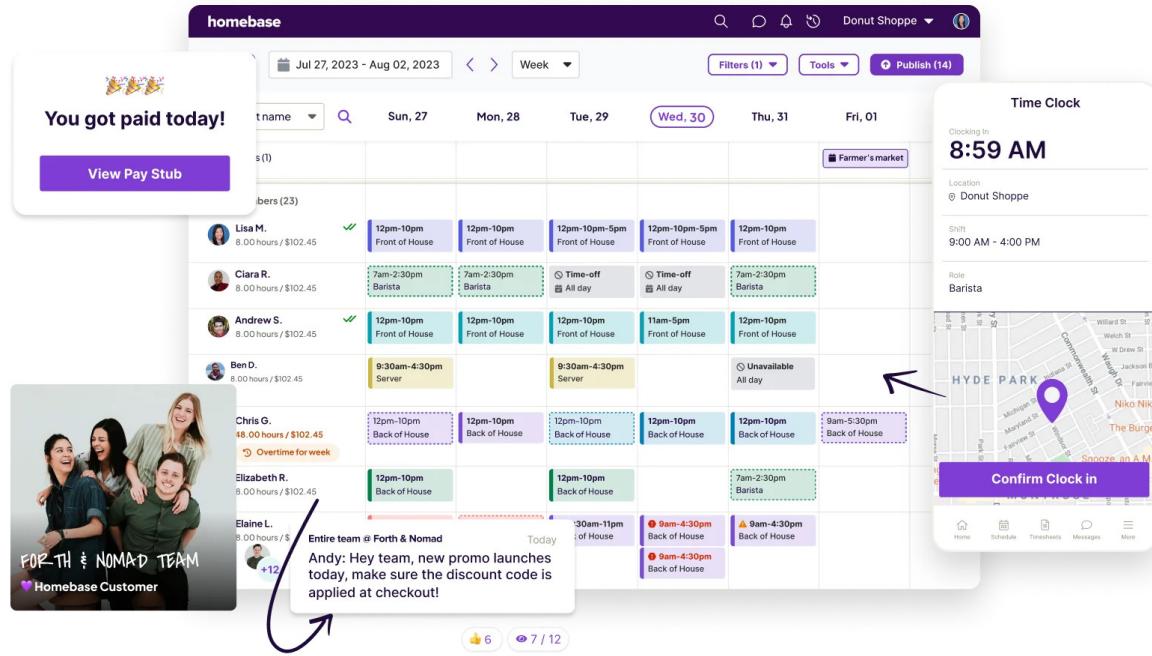
3.2.2 Homebase

Το **Homebase** είναι μία ολιστική εφαρμογή διαχείρισης προσωπικού μίας επιχείρησης, της οποίας οι λειτουργίες εκτείνονται πέρα από τον χρονοπρογραμματισμό βαρδιών. Επιπλέον, περιλαμβάνουν:

- Επισκόπηση της κατάστασης εργασίας του προσωπικού και εξαγωγή στατιστικών
- Πληρωμές
- Διαχείριση προσλήψεων και εκπαίδευσης

3.2 Μελέτη ανταγωνιστικών λύσεων

- Επικοινωνία
- Λειτουργίες ανθρωπίνου δυναμικού και ευχαρίστησης εργαζομένων



Εικόνα 3.12. Η εφαρμογή Homebase (πηγή: joinhomebase.com)

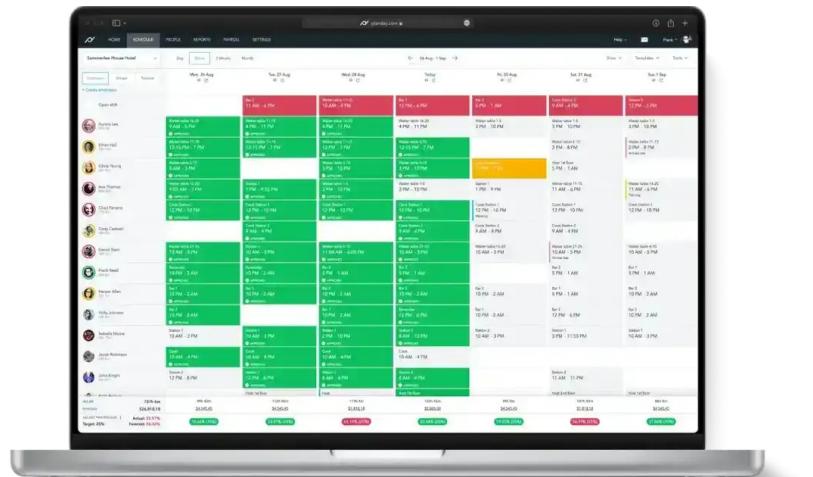
3.2.3 Planday

Το Planday είναι ακόμα μία εφαρμογή που περιλαμβάνει όλες τις παραπάνω λειτουργίες, αλλά επικεντρώνεται κυρίως στην επεκτασιμότητα σε πολλούς κλάδους επιχειρήσεων, όπως λιανική, τουρισμό, υγεία κ.λπ., αλλά και σε επιχειρήσεις διαφορετικού μεγέθους. Υποστηρίζει επίσης τη διαλειτουργικότητα με άλλες εφαρμογές, προσφέροντας στους προγραμματιστές ένα δημόσια διαθέσιμο API.

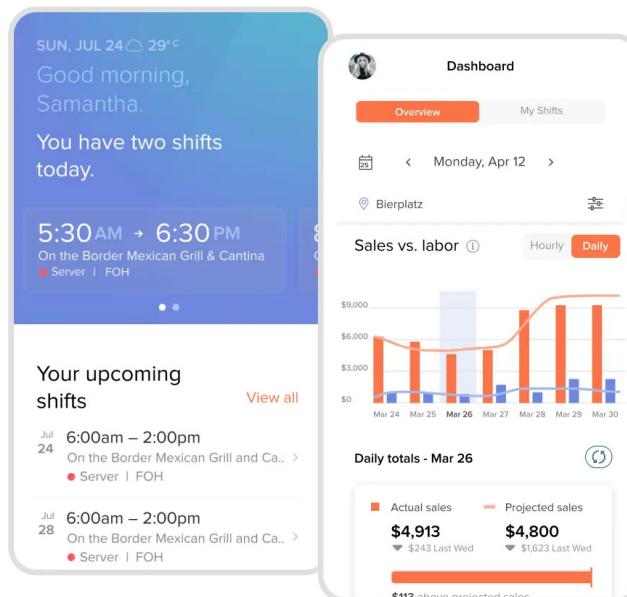
3.2.4 7shifts

Το 7shifts είναι μία πλατφόρμα διαχείρισης προσωπικού για εστιατόρια. Σκοπός του είναι η απλοποίηση του χρονοπρογραμματισμού εργασίας, υποστηρίζοντας παρόμοιες λειτουργίες με τις προαναφερθείσες λύσεις.

3.2 Μελέτη ανταγωνιστικών λύσεων



Εικόνα 3.13. Η εφαρμογή Planday (πηγή: planday.com)

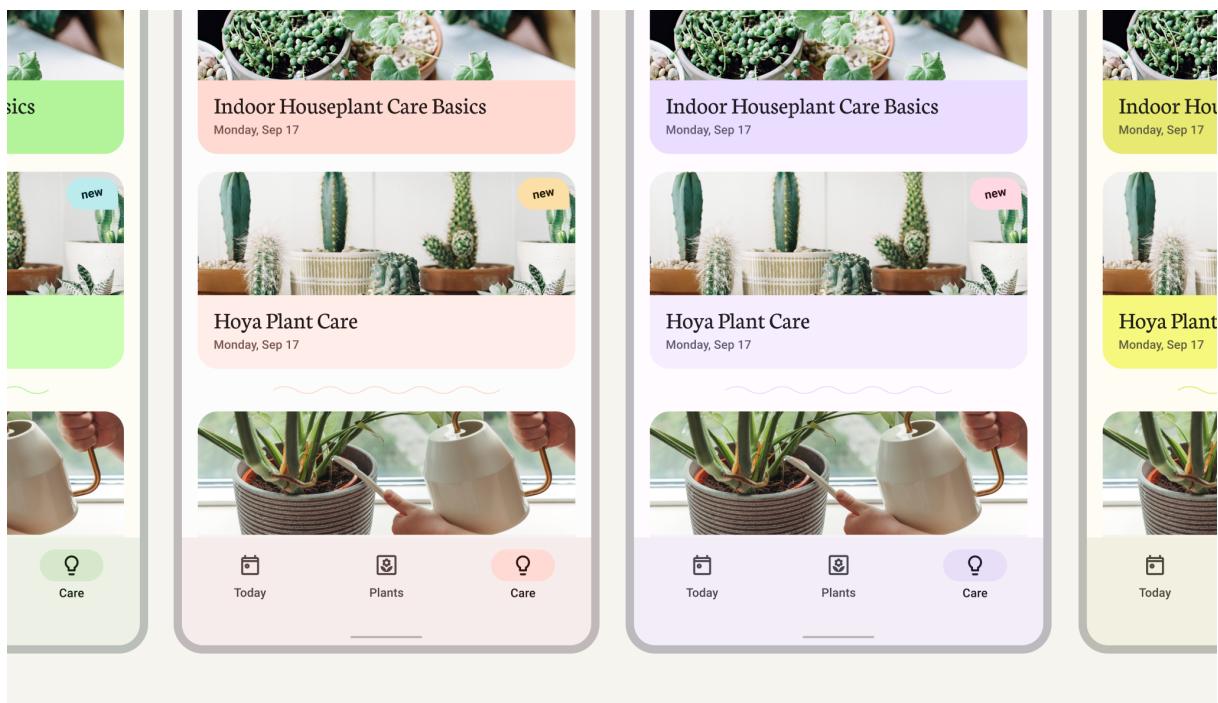


Εικόνα 3.14. Η εφαρμογή 7shifts (πηγή: 7shifts.com)

3.3 Σχεδιαστικές επιλογές

Έχοντας πλέον μελετήσει και τις ανταγωνιστικές λύσεις και έχοντας καταλήξει στο σύνολο των λειτουργιών που θα υποστηρίζει η εφαρμογή μας, προχωρήσαμε σε κάποιες επιλογές, στις οποίες θα βασιστεί η σχεδίαση της γραφικής διεπαφής.

Η βασική σχεδιαστική επιλογή που πραγματοποιήσαμε είναι η υιοθέτηση του **Material Design**. Το **Material Design**, με την πιο πρόσφατη έκδοσή του, Material 3, είναι ένα σύστημα κατευθυντήριων γραμμών (guidelines), γραφικών στοιχείων (components) και εργαλείων (tools), τα οποία υποστηρίζουν τις βέλτιστες πρακτικές σχεδίασης γραφικών διεπαφών. Αναπτυγμένο από την Google, αλλά υποστηριζόμενο από την κοινότητα ως έργο ανοιχτού κώδικα, το Material Design διευκολύνει τη συνεργασία μεταξύ σχεδιαστών και προγραμματιστών (Google, χ.χ.) (Εικόνα 3.15).



Εικόνα 3.15. Μία ενδεικτική εφαρμογή που ακολουθεί τις προδιαγραφές του Material 3 (πηγή: m3.material.io)

3.3.1 Επωνυμία και λογότυπο

Από τις πιο σημαντικές επιλογές, με τις οποίες έρχεται αντιμέτωπος ο σχεδιαστής οποιουδήποτε προϊόντος είναι η επιλογή της κατάλληλης επωνυμίας. Ιδιαίτερα για μία εφαρμογή, όπου ο ανταγωνισμός είναι μεγάλος, η επωνυμία του προϊόντος είναι αυτή που το καθιστά μοναδικό και αναγνωρίσιμο. Θεωρείται ότι η επιλογή της επωνυμίας είναι ίσως “η πιο σημαντική απόφαση

3.3 Σχεδιαστικές επιλογές

μάρκετινγκ που μπορεί κανείς να πάρει” (Ries και Trout 2001). Υπάρχουν πολλοί κανόνες του μάρκετινγκ που προδιαγράφουν πώς ορίζεται μία “καλή” επωνυμία, όπως να είναι απλή, ξεχωριστή και με νόημα (Robertson 1989).

Για την εφαρμογή μας, επιλέξαμε το όνομα **Horario**. Το horario είναι η ισπανική μετάφραση της λέξης “πρόγραμμα” ή “χρονοδιάγραμμα”. Η επωνυμία αυτή ταιριάζει απόλυτα με την περιγραφή και τον σκοπό μίας εφαρμογής που δημιουργεί αυτόματα το πρόγραμμα βαρδιών για μία μεγάλη επιχείρηση. Το όνομα είναι απλό και εύηχο (διαβάζεται /o'rɑ:jo/, κατά το Διεθνές Φωνητικό Αλφάβητο - IPA). Ταυτόχρονα, παραπέμπει στην ελληνική λέξη “ωράριο”, γεγονός που διευκολύνει τους Έλληνες χρήστες της εφαρμογής, αφού αυτή απευθύνεται πρωτίστως στην ελληνική αγορά.

Για τη σχεδίαση του λογοτύπου της εφαρμογής, καθώς και την επιλογή του κυρίαρχου χρώματος της επωνυμίας, χρησιμοποιήσαμε την εφαρμογή **Brandmark** και στη συνέχεια, επεξεργαστήκαμε τις προτάσεις στο **Figma**. Στην Εικόνα 3.16 φαίνεται το κύριο λογότυπο της εφαρμογής, με το σύνθημα (σλόγκαν) της επωνυμίας, ενώ στην Εικόνα 3.17, φαίνεται το μονόγραμμα της επωνυμίας, το οποίο χρησιμοποιείται σε σημεία όπου απαιτείται μικρών διαστάσεων λογότυπο, όπως σε εικονίδια (favicon) και την οθόνη καλωσορίσματος (splash screen).



Εικόνα 3.16. Το λογότυπο της επωνυμίας *Horario*



Εικόνα 3.17. Το μονόγραμμα της επωνυμίας *Horario*

3.3.2 Χρωματική παλέτα

Οι προδιαγραφές του Material 3 καθορίζουν ένα ενιαίο χρωματικό σύστημα, το οποίο αναδεικνύει τον ρόλο του καθενός στοιχείου της διεπαφής, αλλά και την ιεραρχία του στον χώρο. Έτσι, κάθε στοιχείο είναι αναγνωρίσιμο και η λειτουργία του αναμενόμενη, βελτιώνοντας την εμπειρία χρήστη.

Συγκεκριμένα, το χρωματικό σύστημα του Material 3 ορίζει μία πληθώρα χρωματικών ρόλων (Google, χ.χ.), όπως:

- **Πρωτεύων (primary), δευτερεύων (secondary), τριτεύων (tertiary)**: δίνουν έμφαση διαφορετικού επιπέδου σε στοιχεία του προσκηνίου (foreground).
- **Περιέκτης (container)**: χρησιμοποιείται ως γέμισμα για στοιχεία του προσκηνίου.
- **Επάνω σε (on)**: χρησιμοποιείται για στοιχεία που βρίσκονται επάνω σε κάποιο αντίστοιχο χρώμα. Για παράδειγμα, ο ρόλος *on primary* χρησιμοποιείται για κείμενο και εικονίδια που βρίσκονται επάνω σε χρώμα *primary*.

Η χρωματική παλέτα του Material 3 μπορεί να είναι είτε στατική, είτε δυναμική (Google, χ.χ.):

- Η **στατική** χρωματική παλέτα καθορίζεται από τον σχεδιαστή και παραμένει σταθερή ανεξαρτήτως των επιλογών του χρήστη. Τα στατικά χρώματα προτιμώνται συνήθως σε εφαρμογές, στις οποίες πρέπει να αναδειχθεί το κυριαρχο χρώμα της επωνυμίας.
- Η **δυναμική** χρωματική παλέτα δεν καθορίζεται από τον σχεδιαστή και δεν είναι γνωστή σε αυτόν. Προκύπτει κατά την εκτέλεση (run-time) της εφαρμογής, από επιλογές περιβάλλοντος του χρήστη, όπως τα χρώματα της ταπετσαρίας (wallpaper) της συσκευής του. Τα δυναμικά χρώματα προτιμώνται συνήθως σε εφαρμογές, στις οποίες δίνεται έμφαση στην εξατομίκευση και στις προσωπικές προτιμήσεις, και όχι στην ίδια την επωνυμία.

Για την εφαρμογή **Horario**, επιλέξαμε μία στατική χρωματική παλέτα.

Τα χρώματα της στατικής παλέτας του Material 3 μπορούν είτε να επιλεγούν ξεχωριστά από τον σχεδιαστή, είτε να δημιουργηθούν αυτόματα από εργαλείο του Material Design, το [Material Theme Builder](#). Για την αυτόματη δημιουργία της παλέτας, απαιτείται ένα μόνο βασικό χρώμα (source color), το οποίο και συνήθως επιλέγεται να είναι το χρώμα της επωνυμίας της εφαρμογής.

Για την εφαρμογή **Horario**, επιλέξαμε ως βασικό χρώμα μία απόχρωση του μπλε, διότι από τη θεωρία χρωμάτων γνωρίζουμε ότι το μπλε μεταδίδει στους χρήστες συναισθήματα, όπως σταθερότητα, εμπιστοσύνη και αξιοπιστία. Αυτά τα συναισθήματα είναι σημαντικά για κάθε εφαρμογή, για αυτό άλλωστε το μπλε κυριαρχεί στις περισσότερες σύγχρονες εφαρμογές. Στην Εικόνα 3.18 αναπαρίσταται η χρωματική παλέτα της εφαρμογής, όπως δημιουργήθηκε από το [Material Theme Builder](#) για ανοιχτόχρωμο θέμα (light theme), ενώ στην Εικόνα 3.19 αναπαρίσταται η αντίστοιχη παλέτα για σκούροχρωμο θέμα (dark theme).

3.3.3 Εικονίδια και γραφικά

Ένα σημαντικό τμήμα κάθε γραφικής διεπαφής είναι τα εικονίδια (icons). Τα εικονίδια είναι χρήσιμη για να επιταχυνθεί η επεξεργασία της πληροφορίας από τον χρήστη. Στο πνεύμα αυτό, προσπαθήσαμε τα κείμενα, δηλαδή οι φράσεις και οι τίτλοι οθονών να έχουν όσο το δυνατόν πιο συνοπτική διατύπωση. Στηριχθήκαμε σε μεγάλο βαθμό στην αρχή του μινιμαλισμού, της απλότητας και της αποφυγής περιττών στοιχείων (8^{ος} κανόνας Nielsen).

3.3 Σχεδιαστικές επιλογές

Primary		Secondary		Tertiary		Error	
	P-40		S-40		T-40		E-40
On Primary	P-100	On Secondary	S-100	On Tertiary	T-100	On Error	E-100
Primary Container	P-90	Secondary Container	S-90	Tertiary Container	T-90	Error Container	E-90
On Primary Container	P-10	On Secondary Container	S-10	On Tertiary Container	T-10	On Error Container	E-10
Surface Dim		Surface		Surface Bright		Inverse Surface	
N-87		N-98		N-98		N-20	
Surf. Container Lowest	Surf. Container Low	Surf. Container	Surf. Container High	Surf. Container Highest		Inverse On Surface	N-95
N-100	N-96	N-94	N-92	N-90		Inverse Primary	P-80
On Surface	N-10	On Surface Var.	NV-30	Outline	NV-50	Outline Variant	NV-80
						Scrim	N-0
						Shadow	N-0

Εικόνα 3.18. Η χρωματική παλέτα της εφαρμογής για ανοιχτόχρωμο θέμα (light theme)

Dark Scheme					
Primary		Secondary		Tertiary	
	P-80		S-80		T-80
On Primary	P-20	On Secondary	S-20	On Tertiary	T-20
Primary Container		Secondary Container		Tertiary Container	
	P-30		S-30		T-30
On Primary Container	P-90	On Secondary Container	S-90	On Tertiary Container	T-90
Surface Dim		Surface		Surface Bright	
	N-6		N-6		N-24
Surf. Container Lowest	Surf. Container Low	Surf. Container	Surf. Container High	Surf. Container Highest	
N-4	N-10	N-12	N-17	N-24	
On Surface	N-90	On Surface Var.	NV-90	Outline	NV-60
				Outline Variant	NV-30
				Scrim	N-0
				Shadow	N-0

Εικόνα 3.19. Η χρωματική παλέτα της εφαρμογής για σκουρόχρωμο θέμα (dark theme)

3.4 Σχεδίαση σκαριφημάτων και πρωτοτύπων

Η Google προσφέρει μία πολύ μεγάλη [βιβλιοθήκη εικονιδίων](#), τα οποία εναρμονίζονται με τις υπόλοιπες σχεδιαστικές οδηγίες του Material Design και τα οποία χρησιμοποιήσαμε εκτενώς στην εφαρμογή μας.

Ακόμη, χρησιμοποιήσαμε την ιστοσελίδα [unDraw](#), για να βρούμε μοντέρνα γραφικά (illustrations), τα οποία αντικαθιστούν εικόνες συγκεκριμένων δεδομένων, όταν δεν έχουν επιλεγεί αντίστοιχες από τον χρήστη (για παράδειγμα, εικόνα μίας ομάδας της επιχείρησης).

3.3.4 Γραμματοσειρές

Ο τρόπος παρουσίασης του κειμένου είναι επίσης καθοριστικής σημασίας σε μία μοντέρνα εφαρμογή. Η Google προσφέρει επίσης μία πολύ μεγάλη [βιβλιοθήκη γραμματοσειρών](#), που ταιριάζουν με τις προδιαγραφές του Material Design.

Για την εφαρμογή **Horario** χρησιμοποιήθηκε η προεπιλεγμένη γραμματοσειρά του Material 3, η Roboto. Πρόκειται για μία sans-serif γραμματοσειρά, η οποία προσφέρει σύγχρονη και καθαρή εμφάνιση. Ταυτόχρονα, υποστηρίζει πολλές γλώσσες, μία από τις οποίες είναι και τα Ελληνικά.

Ακόμη, χρησιμοποιήθηκε η καλλιγραφική γραμματοσειρά Dancing Script, σε ένα μόνο σημείο και συγκεκριμένα στον χαιρετισμό του χρήστη στην Αρχική Οθόνη. Η χρήση καλλιγραφικής γραμματοσειράς σε αυτό το σημείο έχει σκοπό να προσθέσει μια πινελιά προσωπικότητας και διασκέδασης στην εφαρμογή.

3.4 Σχεδίαση σκαριφημάτων και πρωτοτύπων

Ξεκινήσαμε με μία πρώιμη απεικόνιση των οθονών της εφαρμογής μας, σχεδιάζοντας σκίτσα στο χαρτί. Τα **σκαριφήματα** (wireframes) είναι μία αναπαράσταση του σχεδιασμού που ορίζει μόνο τη στατική δομή και τη διάταξη της διεπαφής. Γνωρίζουμε ότι η τεχνική αυτή προτιμάται από τους σχεδιαστές, αφού το αποτέλεσμα είναι αναλώσιμο και μπορεί πολύ εύκολα να τροποποιηθεί, χωρίς μεγάλο κόστος.

Ωστόσο, τα σκαριφήματα δεν περιλαμβάνουν διαδραστικά στοιχεία, γεγονός το οποίο καθιστά δύσκολη την αξιολόγηση της εμπειρίας χρήστη. Για τον σκοπό αυτό, προχωρήσαμε στην ανάπτυξη **πρωτοτύπων** (prototypes). Τα πρωτότυπα αποτελούν οπτικές αναπαραστάσεις του τελικού προϊόντος, τα οποία έχουν και κάποια λειτουργικά χαρακτηριστικά, όπως, για παράδειγμα, η πλοήγηση και το πάτημα κουμπιών. Επομένως, μέσω αυτών, μπορούμε να μετρήσουμε την εμπειρία χρήστη.

Για τη σχεδίαση των πρωτοτύπων, αποφύγαμε τη χρήση κάποιου σχεδιαστικού εργαλείου, όπως το [Figma](#), κυρίως για λόγους εξοικονόμησης χρόνου. Αντ' αυτού, ξεκινήσαμε απευθείας με τη δόμηση της διάταξης (layout) της εφαρμογής, γράφοντας κώδικα Flutter, δεδομένου ότι το Flutter αποτελεί

ένα framework, του οποίου το τελικό αποτέλεσμα είναι το ίδιο με αυτό που φαίνεται τη στιγμή της ανάπτυξης (*what-you-see-is-what-you-get*, WYSIWYG). Ωστόσο, στο στάδιο αυτό, το αποτέλεσμα δεν περιλαμβάνει λειτουργικότητα, πέρα από τις βασικές που προβλέπονται από τον ορισμό του πρωτότυπου, όπως αποθήκευση δεδομένων. Η πλήρης ανάπτυξη της εφαρμογής θα αναλυθεί στο Κεφάλαιο 4.

Στις επόμενες Ενότητες, παρατίθενται τα σκαριφήματα και τα πρωτότυπα που σχεδιάστηκαν για τις διάφορες οθόνες της εφαρμογής και γίνεται σχολιασμός τους.

3.4.1 Αρχική οθόνη (Home page)

Η αρχική οθόνη είναι η πρώτη με την οποία έρχεται σε επαφή ο χρήστης, όταν ανοίξει την εφαρμογή. Ο κύριος σκοπός της είναι:

- να καλωσορίσει τον χρήστη,
- να τον ενημερώσει για τη τρέχουσα κατάσταση του συστήματος,
- να του δώσει μία συνοπτική εικόνα για τα επερχόμενα συμβάντα και
- να τον καθοδηγήσει προς τις υπόλοιπες οθόνες της εφαρμογής.

Στην Εικόνα 3.20 βλέπουμε το σκαρίφημα της Αρχικής οθόνης και τα πρωτότυπα σε ανοιχτόχρωμο και σκουρόχρωμο θέμα, καθώς και στην Ελληνική έκδοση.

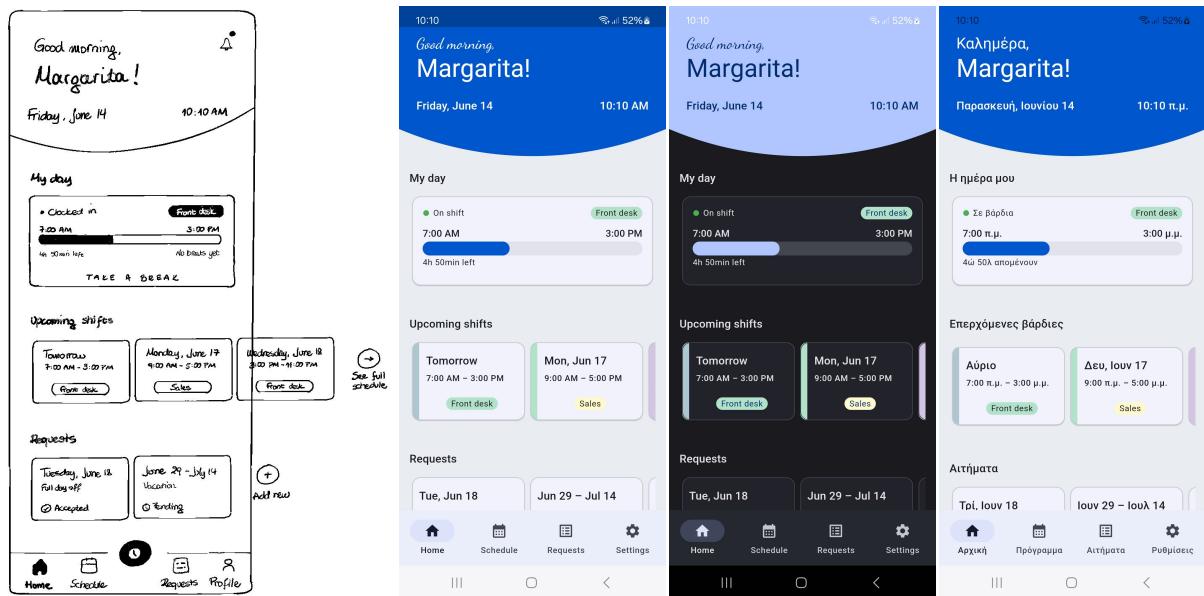
Αξίζει να αναφερθεί πως κατά τη δημιουργία των πρωτοτύπων, κάποια στοιχεία αφαιρέθηκαν από το σχέδιο, που σχετίζονται κυρίως με λειτουργίες, η υλοποίηση των οποίων δεν ήταν αρκετά υψηλής προτεραιότητας. Μία από αυτές είναι η χρονική παρακολούθηση της εργασίας (*ψηφιακή κάρτα*). Τα σχετικά στοιχεία που απαντώνται στο σκαρίφημα, αλλά αφαιρέθηκαν στο πρωτότυπο είναι το κεντρικό κουμπί ψηφιακής κάρτας (το κουμπί με εικονίδιο ρολογιού στη γραμμή πλοήγησης) και το κουμπί έναρξης διαλείμματος στην κάρτα τρέχουσας κατάστασης του συστήματος (*My day*).

Πατώντας πάνω σε οποιαδήποτε κάρτα της Αρχικής οθόνης (*My day*, *Upcoming shifts* και *Requests*), ο χρήστης πλοηγείται γρήγορα στα αντίστοιχα στοιχεία, χωρίς να χρειάζεται πρώτα να μεταφερθεί στις επιμέρους οθόνες. Επιπλέον, όπως φαίνεται από το σκαρίφημα, με κύλιση προς τα δεξιά, ο χρήστης έχει τη δυνατότητα να δει το πλήρες πρόγραμμα εργασίας, μέσω της συντόμευσης *See full schedule*, αλλά και να δημιουργήσει ένα νέο αίτημα (*Add new*).

3.4.2 Οθόνη προγράμματος: Επιλογή προβολής (Schedule page: View selection)

Στην Εικόνα 3.21 βλέπουμε το σκαρίφημα και τα πρωτότυπα της οθόνης επιλογής προβολής για την οθόνη Προγράμματος. Σε αυτή την οθόνη ο χρήστης έχει τη δυνατότητα να επιλέξει μία από τις διαθέσιμες προβολές του προγράμματος εργασίας, την Ατομική προβολή (Ενότητα 3.4.3) και την Ομαδική προβολή (Ενότητα 3.4.4).

3.4 Σχεδίαση σκαριφημάτων και πρωτοτύπων



Εικόνα 3.20. Η Αρχική οθόνη: σκαρίφημα και πρωτότυπα

Για την Ατομική προβολή, οι διαθέσιμες επιλογές είναι πρωτίστως, ο τρέχων χρήστης, αλλά και οι συνάδελφοί του στις ομάδες που ανήκει. Η πρόσβαση στις προβολές αυτές προσφέρει διαφάνεια του συστήματος προς τον χρήστη, ώστε να μπορεί να επιβεβαιώσει πως δε συμβαίνουν αδικίες κατά την αυτόματη ανάθεση βαρδιών.

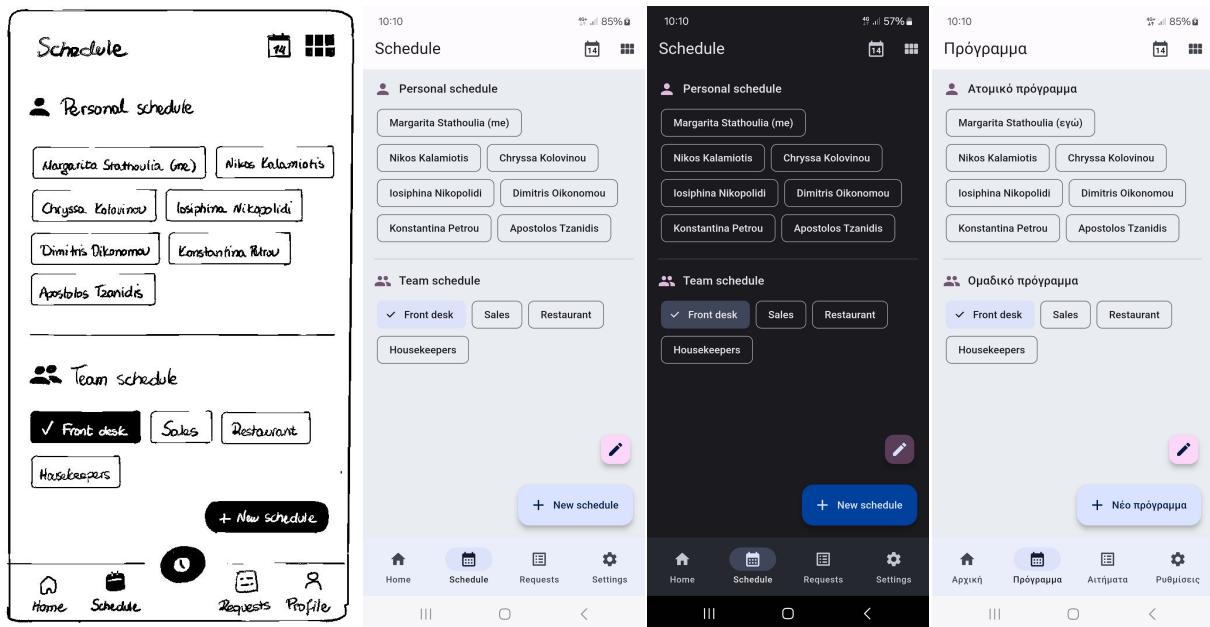
Για την Ομαδική προβολή, οι διαθέσιμες επιλογές είναι οι ομάδες στις οποίες ανήκει ο χρήστης. Γενικότερα, εφαρμόζεται η Αρχή του ελαχίστου δικαιώματος (*Least privilege principle*), σύμφωνα με την οποία, τα δεδομένα στα οποία έχει πρόσβαση ο χρήστης και οι ενέργειες που αυτός δικαιούται να εκτελέσει, είναι οι ελάχιστες που απαιτούνται ώστε να εκτελεστεί μία εργασία, δεδομένων των δικαιωμάτων του χρήστη.

Στην κάτω δεξιά γωνία, και σε οποιαδήποτε προβολή της οθόνης Προγράμματος, βρίσκονται τα κουμπιά επεξεργασίας και δημιουργίας προγράμματος εργασίας, τα οποία είναι ορατά μόνο στους χρήστες με δικαιώματα Διευθυντή (Manager) ή Διαχειριστή (Administrator) και μόνο για τις ομάδες που αυτοί διευθύνουν (Ενότητα 3.4.6). Επίσης, στην επάνω δεξιά γωνία, και σε οποιαδήποτε προβολή της οθόνης Προγράμματος, υπάρχει το κουμπί γρήγορης μετάβασης στην τρέχουσα ημέρα, αλλά και το κουμπί εμφάνισης και απόκρυψης της οθόνης επιλογής προβολής.

3.4.3 Οθόνη προγράμματος: Ατομική προβολή (Schedule page: Personal view)

Στην Εικόνα 3.22 βλέπουμε το σκαρίφημα και τα πρωτότυπα της Ατομικής προβολής της οθόνης Προγράμματος. Σε αυτή την οθόνη ο χρήστης έχει τη δυνατότητα να δει το πρόγραμμα εργασίας

3.4 Σχεδίαση σκαριφημάτων και πρωτότυπων



Εικόνα 3.21. Η οθόνη επιλογής προβολής για την οθόνη Προγράμματος: σκαρίφημα και πρωτότυπα

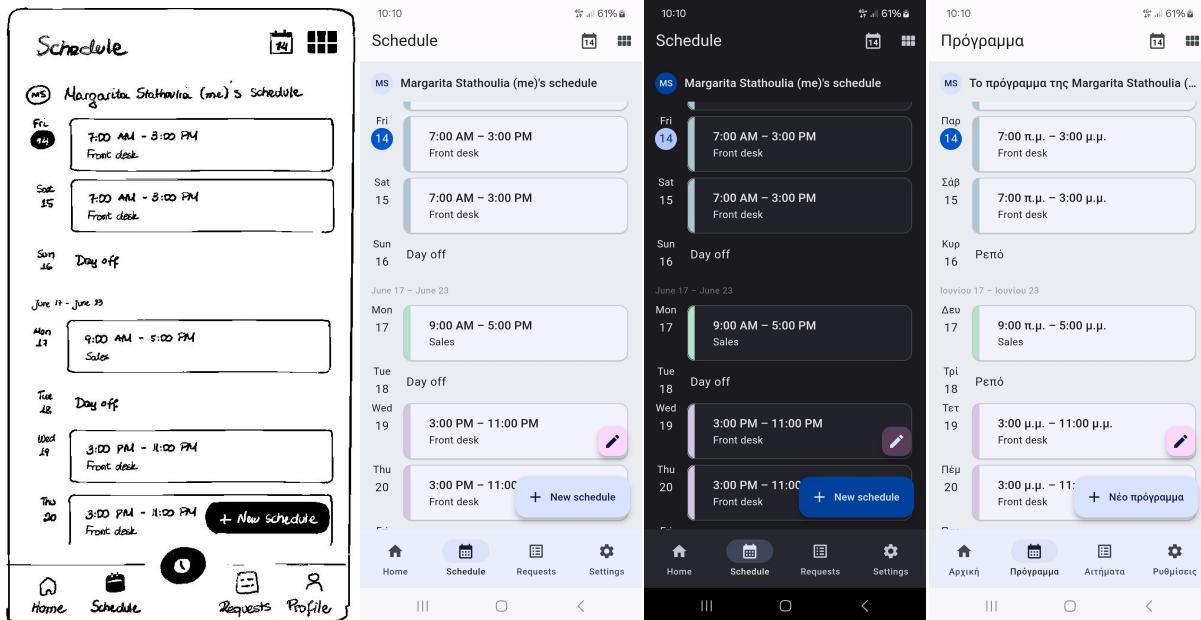
εστιασμένο σε ένα μόνο άτομο, σε μορφή ατζέντας. Ο χρόνος αναπτύσσεται στον κατακόρυφο άξονα, με ένα στοιχείο για κάθε ημέρα και ομαδοποιημένα κατά εβδομάδες.

Η κάθε βάρδια αναπαρίσταται με μία κάρτα, στην οποία αναφέρεται η ομάδα στην οποία θα πρέπει να εργαστεί ο χρήστης για τη συγκεκριμένη βάρδια, καθώς και οι ώρες έναρξης και λήξης αυτής. Πατώντας στην κάρτα ανοίγει το Φύλλο βάρδιας, το οποίο παρέχει στον χρήστη περισσότερες λεπτομέρειες σχετικά με την επιλεγμένη βάρδια (Ενότητα 3.4.5). Αξίζει να αναφέρουμε πως έχουμε χρησιμοποιήσει χρωματικούς κώδικες για τις ομάδες και τις βάρδιες, ώστε να διευκολύνουμε τον χρήστη από την ανάγνωση μονότονου κειμένου (2^os κανόνας Nielsen).

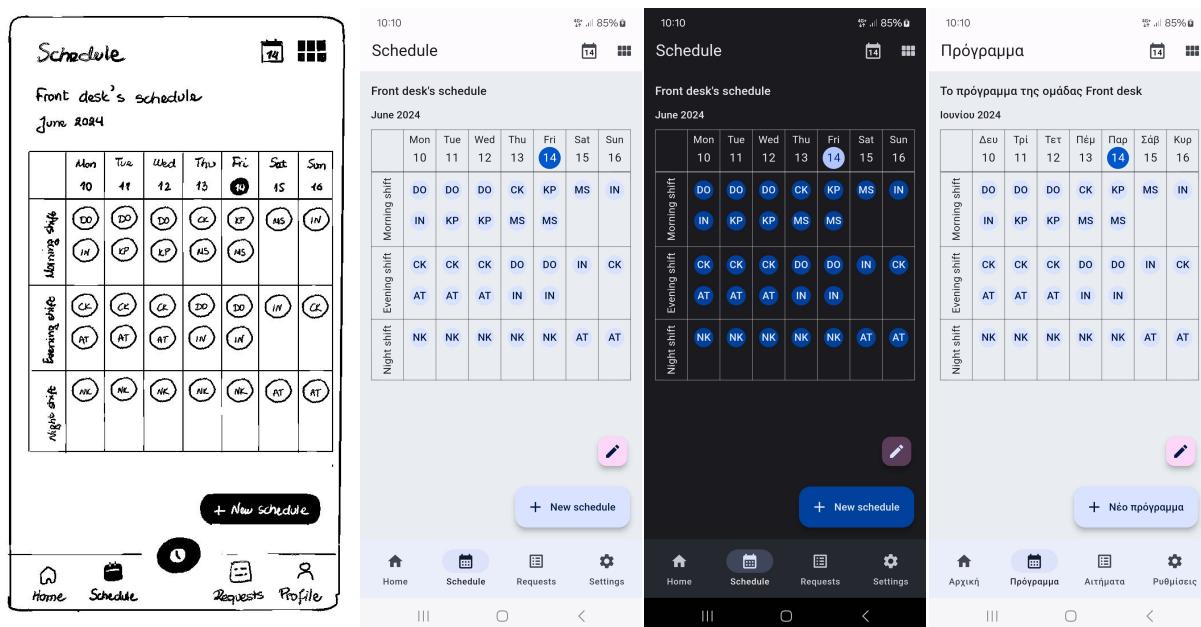
3.4.4 Οθόνη προγράμματος: Ομαδική προβολή (Schedule page: Team view)

Στην Εικόνα 3.23 βλέπουμε το σκαρίφημα και τα πρωτότυπα της Ομαδικής προβολής της οθόνης Προγράμματος. Σε αυτή την οθόνη ο χρήστης έχει τη δυνατότητα να δει το πρόγραμμα εργασίας εστιασμένο σε μία ομάδα, σε μορφή εβδομαδιαίου ημερολογίου σε πίνακα. Ο χρόνος αναπτύσσεται τόσο στον οριζόντιο άξονα, έχοντας μία στήλη του πίνακα για κάθε ημέρα, αλλά και στον κατακόρυφο άξονα, έχοντας μία γραμμή του πίνακα για κάθε βάρδια εντός της ημέρας. Η κάθε βάρδια περιέχεται σε ένα κελί του πίνακα, στο οποίο εμφανίζονται οι εικόνες των εργαζομένων που στελεχώνουν τη συγκεκριμένη βάρδια. Πατώντας στο κελί ανοίγει το Φύλλο βάρδιας, το οποίο παρέχει στον χρήστη περισσότερες λεπτομέρειες σχετικά με την επιλεγμένη βάρδια (Ενότητα 3.4.5).

3.4 Σχεδίαση σκαριφημάτων και πρωτοτύπων



Εικόνα 3.22. Η Ατομική προβολή της οθόνης Προγράμματος: σκαρίφημα και πρωτότυπα



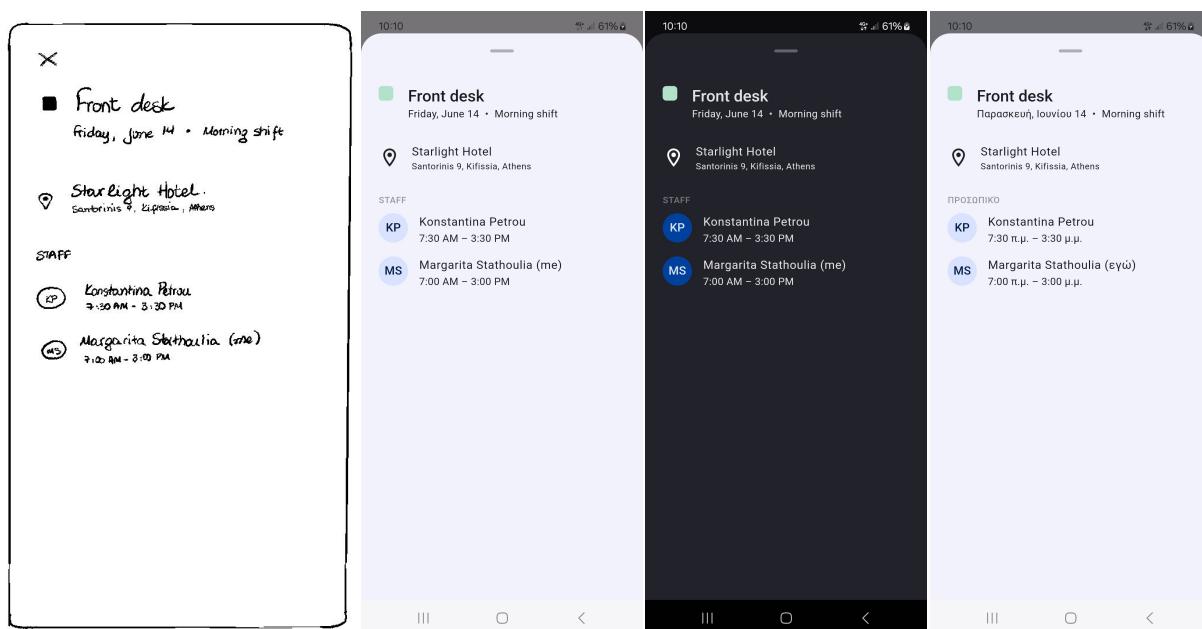
Εικόνα 3.23. Η Ομαδική προβολή της οθόνης Προγράμματος: σκαρίφημα και πρωτότυπα

3.4.5 Φύλλο βάρδιας (Shift sheet)

Στην Εικόνα 3.24 βλέπουμε το σκαριφημα και τα πρωτότυπα του Φύλλου βάρδιας. Το φύλλο αυτό περιέχει αναλυτικές πληροφορίες για τη βάρδια που επέλεξε να δει ο χρήστης, όπως το όνομα της βάρδιας, την ημερομηνία, την ομάδα στην οποία ανήκει η βάρδια και την τοποθεσία αυτής, καθώς και τα ονόματα των εργαζομένων που στελεχώνουν τη βάρδια και τις ώρες έναρξης και λήξης της βάρδιας του καθενός.

Αξίζει να σημειώσουμε πως, παρά τη σταθερή ώρα έναρξης και λήξης της κάθε βάρδιας (για παράδειγμα, η Πρωινή βάρδια (Morning shift) αρχίζει στις 7:00 π.μ. και τελειώνει στις 3:00 μ.μ.), δίνουμε τη δυνατότητα στον δημιουργό του προγράμματος να προσαρμόσει τις ώρες έναρξης και λήξης για κάθε εργαζόμενο ξεχωριστά, σε ένα εύρος τεσσάρων ωρών εκατέρωθεν της προεπιλεγμένης (για παράδειγμα, η Κωνσταντίνα Πέτρου θα προσέλθει στις 7:30 π.μ. και θα αποχωρήσει στις 3:30 μ.μ.).

Το Φύλλο βάρδιας χρησιμοποιείται, προσαρμοσμένο για επεξεργασία, στους οδηγούς δημιουργίας και επεξεργασίας προγράμματος (Ενότητα 3.4.6).



Εικόνα 3.24. Το Φύλλο βάρδιας: σκαριφημα και πρωτότυπα

3.4.6 Οδηγός δημιουργίας / επεξεργασίας προγράμματος (Create / Edit schedule wizard)

Η διαδικασία δημιουργίας ή επεξεργασίας ενός προγράμματος εργασίας παρέχεται στον χρήστη ως μία ακολουθία αυστηρά καθορισμένων βημάτων, σε μορφή οδηγού (wizard). Ένας οδηγός κατευθύνει τον χρήστη στην εκτέλεση όλων των απαραίτητων ενεργειών για την ολοκλήρωση ενός στόχου. Όπως φαίνεται και στις παρακάτω εικόνες, στο πάνω μέρος του οδηγού είναι πάντα ορατή η γραμμή ένδειξης ολοκλήρωσης, καθώς και ένα κείμενο με τις οδηγίες του τρέχοντος βήματος.

Επιπλέον, στο κάτω μέρος του οδηγού βρίσκονται τα κουμπιά πλοήγησης. Το κουμπί *Πίσω* (*Back*) επιτρέπει στον χρήστη να επιστρέψει σε ένα προηγούμενο βήμα, προκειμένου να κάνει μία διόρθωση ή μία διαφορετική επιλογή, ενώ το κουμπί *Συνέχεια* (*Continue*) μεταφέρει τον χρήστη στο επόμενο βήμα. Αξίζει να σημειώσουμε πως η εμφάνιση του κουμπιού *Συνέχεια* εξαρτάται από το περιεχόμενο και την κατάσταση του τρέχοντος βήματος του οδηγού, όπως εξηγείται στις ακόλουθες ενότητες.

Ο οδηγός προγράμματος είναι ο ίδιος, τόσο για τη δημιουργία, όσο και για την επεξεργασία ενός προγράμματος, με μόνη διαφορά ένα επιμέρους βήμα, το οποίο εξηγείται παρακάτω.

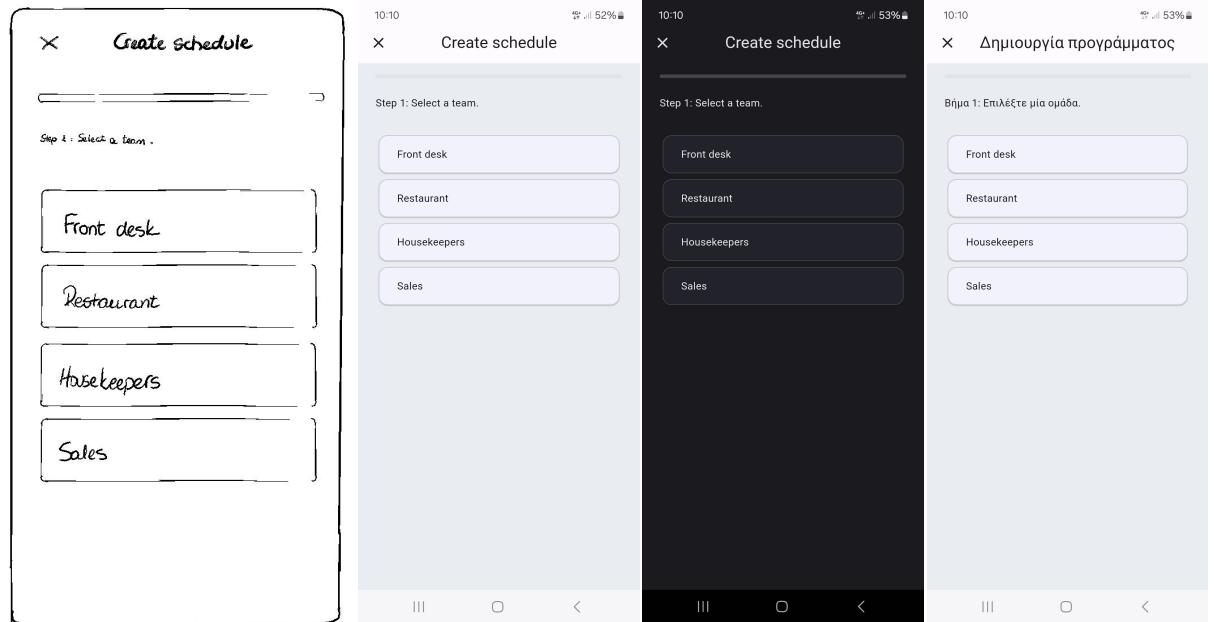
Βήμα 1: Επιλογή ομάδας

Το πρώτο βήμα του οδηγού, τόσο για τη δημιουργία όσο και για την επεξεργασία ενός προγράμματος εργασίας, είναι η επιλογή ομάδας. Κάθε πρόγραμμα αφορά μία μόνο ομάδα, η επιλογή της οποίας είναι και το εναρκτήριο σημείο της διαδικασίας. Ο χρήστης καλείται να επιλέξει μία από τις ομάδες που διευθύνει (Εικόνα 3.25).

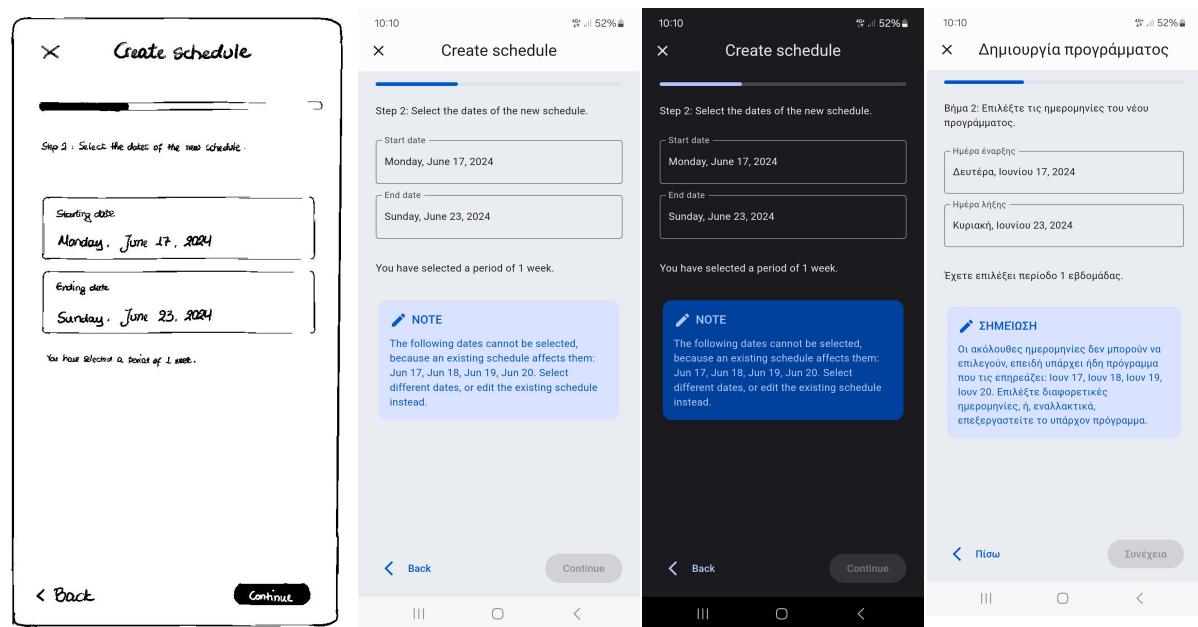
Βήμα 2: Επιλογή ημερομηνιών ή υπάρχοντος προγράμματος

Το δεύτερο βήμα του οδηγού διαφέρει εάν πρόκειται για διαδικασία δημιουργίας ή επεξεργασίας προγράμματος. Στην περίπτωση δημιουργίας νέου (Εικόνα 3.26), ο χρήστης καλείται να επιλέξει ένα εύρος ημερομηνιών, για τις οποίες θα ισχύει το πρόγραμμα. Το χρονικό μήκος αυτού του εύρους μπορεί να είναι μεταβλητό, ώστε να εξυπηρετεί τις ανάγκες της ομάδας. Ο αλγόριθμος αυτόματης ανάθεσης βαρδιών μπορεί να υποστηρίζει μεγάλα εύρη, ακόμα και μηνών, που σπάνια προτιμώνται στις επιχειρήσεις. Το προεπιλεγμένο εύρος είναι μία εβδομάδα και, συγκεκριμένα, η επόμενη από την τρέχουσα. Προϋπόθεση, ώστε οι επιλεγμένες ημερομηνίες να είναι έγκυρες, είναι να μην περιλαμβάνουν πρόγραμμα για τη συγκεκριμένη ομάδα. Σε τέτοια περίπτωση, αντί δημιουργίας νέου, ο χρήστης θα πρέπει να προχωρήσει σε επεξεργασία υπάρχοντος προγράμματος. Εάν οι επιλεγμένες ημερομηνίες δεν είναι έγκυρες, στον χρήστη θα εμφανιστεί ένα μήνυμα που εξηγεί το πρόβλημα, όπως περιγράφεται από τον 9^ο κανόνα του Nielsen.

3.4 Σχεδίαση σκαριφημάτων και πρωτότυπων



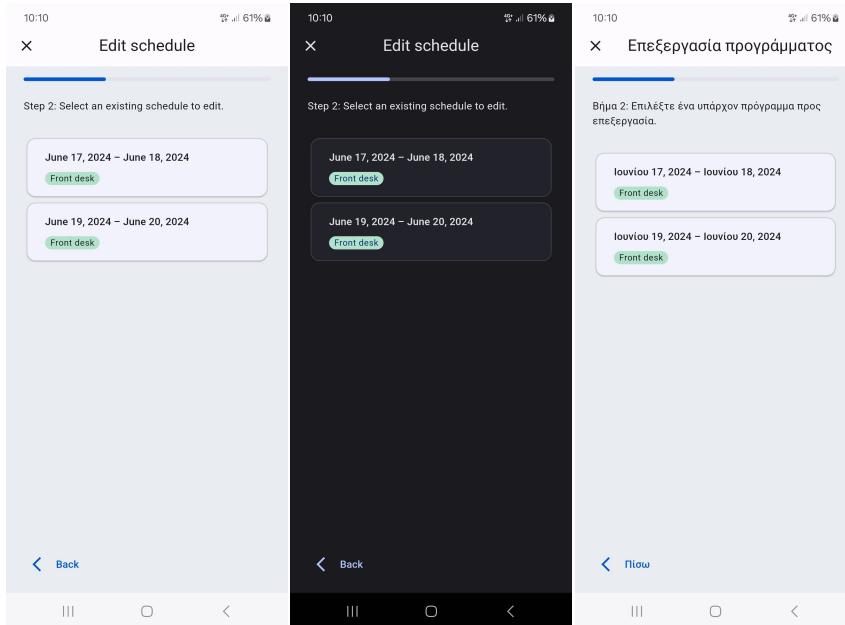
Εικόνα 3.25. Οδηγός προγράμματος: Βήμα 1: σκαρίφημα και πρωτότυπα



Εικόνα 3.26. Οδηγός προγράμματος: Βήμα 2 (δημιουργία νέου): σκαρίφημα και πρωτότυπα

3.4 Σχεδίαση σκαριφημάτων και πρωτοτύπων

Στην περίπτωση επεξεργασίας υπάρχοντος προγράμματος (Εικόνα 3.27), ο χρήστης καλείται να επιλέξει ένα πρόγραμμα, από μία λίστα όλων των μελλοντικών προγραμμάτων για την επιλεγμένη ομάδα.



Εικόνα 3.27. Οδηγός προγράμματος: Βήμα 2 (επεξεργασία υπάρχοντος): πρωτότυπα

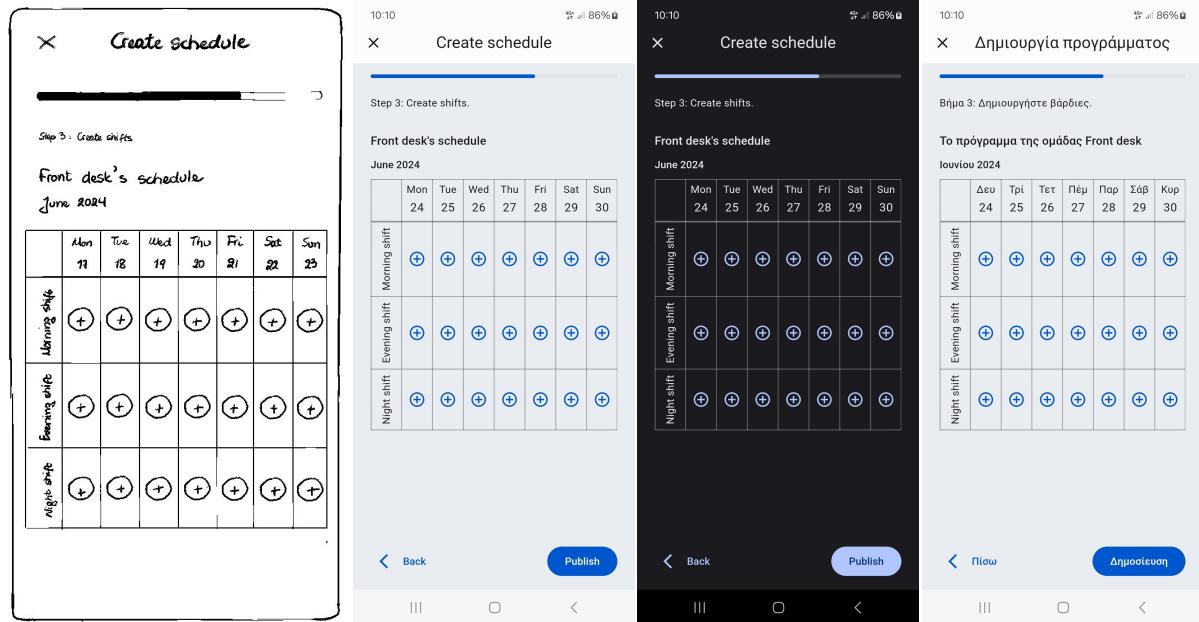
Βήμα 3: Προσθήκη βαρδιών

Το τρίτο βήμα του οδηγού είναι η προσθήκη βαρδιών. Στον χρήστη εμφανίζεται η Ομαδική προβολή της οθόνης Προγράμματος, τροποποιημένη, με εικονίδια προσθήκης σε κάθε κελί (Εικόνα 3.28). Πατώντας στο κελί, εμφανίζεται το Φύλλο βάρδιας, το οποίο επιτρέπει την προσθήκη, αφάίρεση και τροποποίηση αναθέσεων στη συγκεκριμένη βάρδια (Εικόνα 3.29). Στο σημείο αυτό, ο χρήστης μπορεί να επιλέξει μεταξύ χειροκίνητης επιλογής εργαζόμενου και αυτόματης ανάθεσης. Επιπλέον, μπορεί να προσαρμόσει τις ώρες έναρξης και λήξης της βάρδιας για τον εργαζόμενο, καθώς και να προσθέσει μία σημείωση.

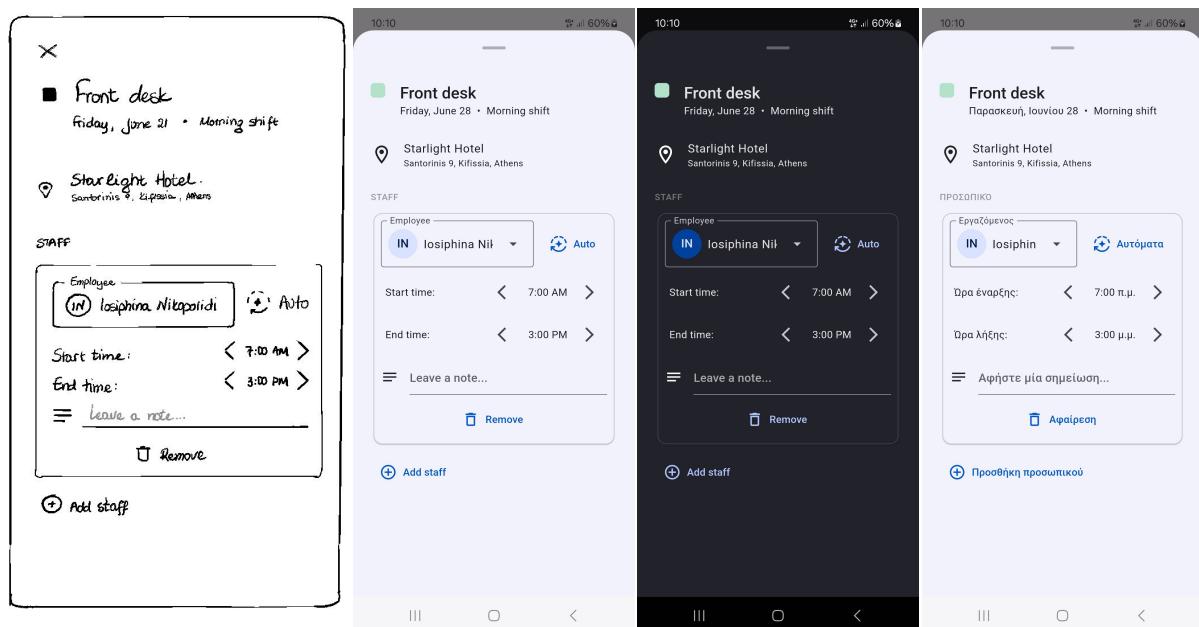
Όταν ολοκληρωθεί η προσθήκη βαρδιών, και στην περίπτωση που έχει επιλεγεί τουλάχιστον μία φορά η αυτόματη ανάθεση βαρδιών, το κουμπί *Συνέχεια (Continue)* αλλάζει σε *Ανάθεση βαρδιών (Assign shifts)*. Πατώντας το, εμφανίζεται μία ερώτηση για επιβεβαίωση (Εικόνα 3.30) και έπειτα εκτελείται ο αλγόριθμος αυτόματης ανάθεσης βαρδιών.

Όταν γίνει η ανάθεση, στον οδηγό εμφανίζονται οι βάρδιες όπως ανατέθηκαν, και το κουμπί *Συνέχεια (Continue)* αλλάζει σε *Δημοσίευση (Publish)*. Πατώντας το, εμφανίζεται μία ερώτηση για επιβεβαίωση

3.4 Σχεδίαση σκαριφημάτων και πρωτότυπων



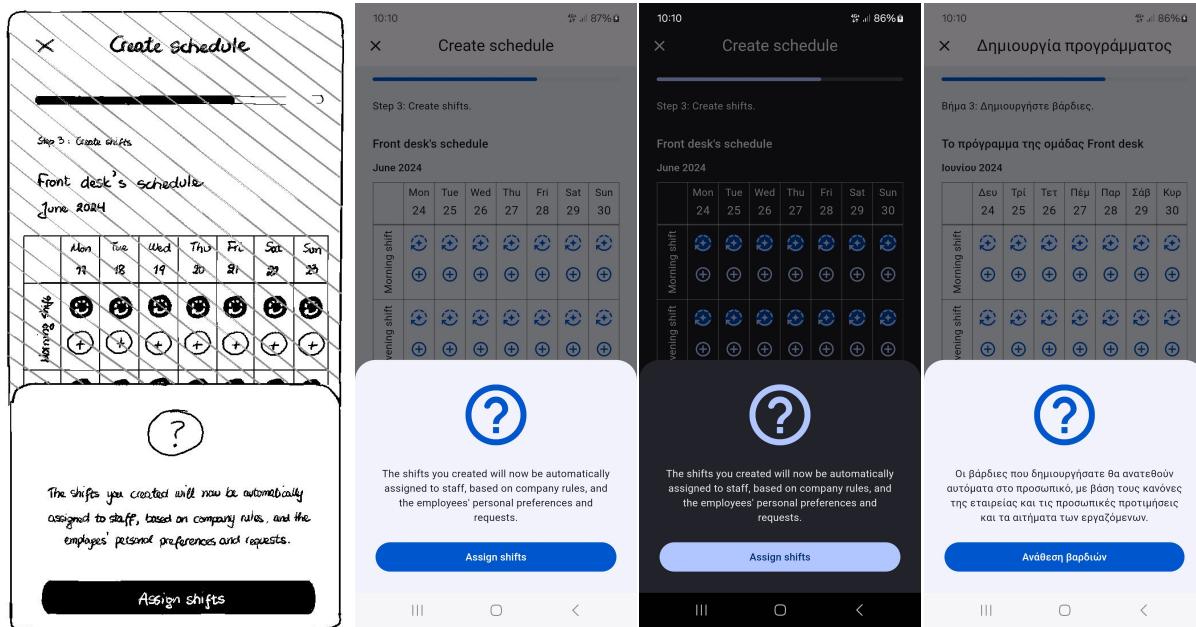
Εικόνα 3.28. Οδηγός προγράμματος: Βήμα 3: σκαρίφημα και πρωτότυπα



Εικόνα 3.29. Οδηγός προγράμματος: Βήμα 3 (Φύλλο βάρδιας): σκαρίφημα και πρωτότυπα

3.4 Σχεδίαση σκαριφημάτων και πρωτοτύπων

(Εικόνα 3.31). Με τη δημοσίευση του προγράμματος, εμφανίζεται μήνυμα επιτυχίας (Εικόνα 3.32), και ο χρήστης επιστρέφει στην Οθόνη προγράμματος.



Εικόνα 3.30. Οδηγός προγράμματος: Βήμα 3 (Επιβεβαίωση αυτόματης ανάθεσης): σκαρίφημα και πρωτότυπα

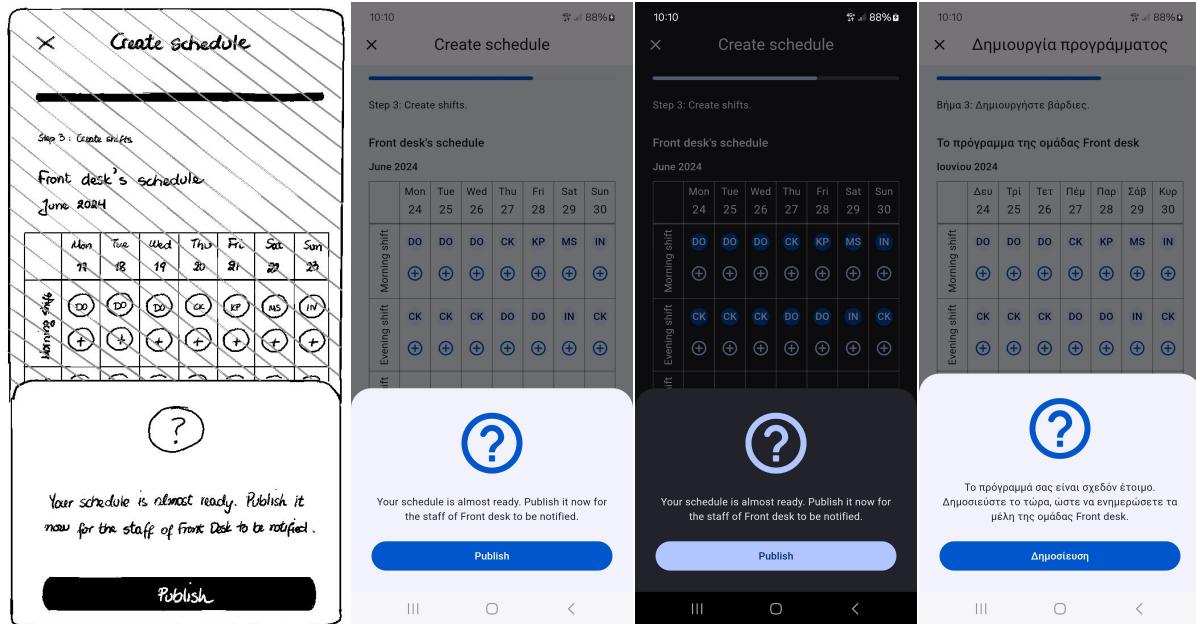
3.4.7 Οθόνη αιτημάτων και Φύλλο αιτήματος (Requests page και Request sheet)

Στην Εικόνα 3.33 βλέπουμε το σκαρίφημα και τα πρωτότυπα της Οθόνης αιτημάτων. Σε αυτή την οθόνη βλέπουμε μία λίστα με όλα τα αιτήματα του χρήστη, ομαδοποιημένα σε επερχόμενα (*Upcoming*) και παρελθοντικά (*Past*). Τα επερχόμενα αιτήματα είναι ταξινομημένα κατά αύξουσα χρονική σειρά (το πιο σύντομο πρώτο), ενώ τα παρελθοντικά αιτήματα είναι ταξινομημένα κατά φθίνουσα χρονική σειρά (το πιο πρόσφατο πρώτο). Το κάθε αίτημα αναπαρίσταται με μία κάρτα, στην οποία αναφέρεται ο τύπος του, η ημερομηνία εφαρμογής, ο κωδικός του και η κατάσταση έγκρισης.

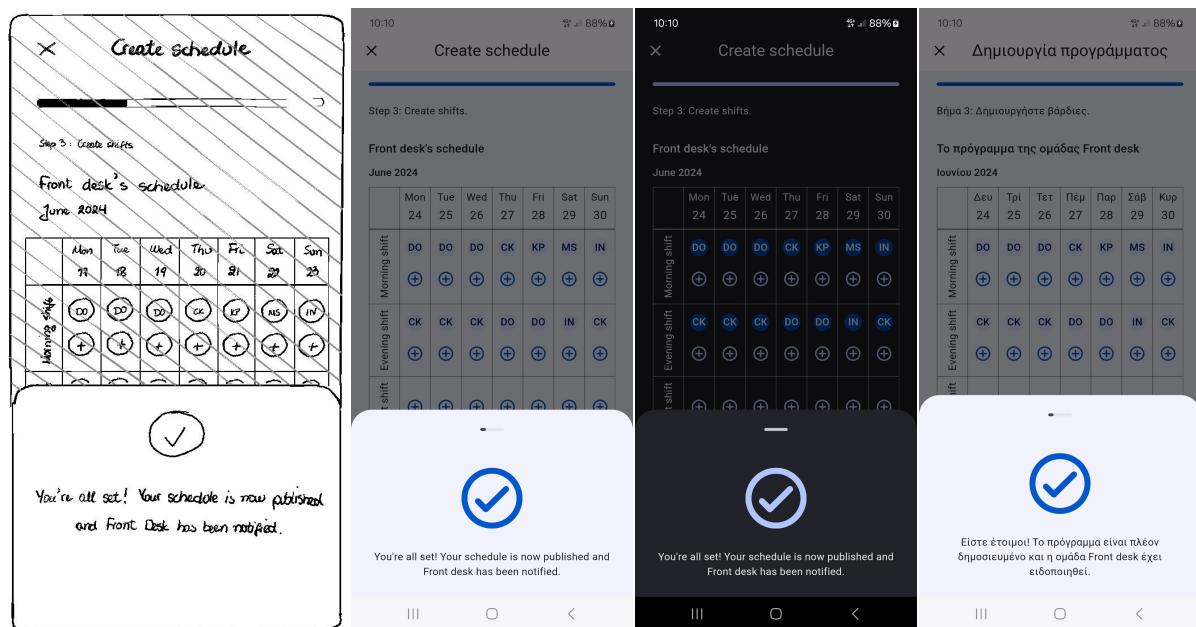
Στην περίπτωση που ο τρέχων χρήστης διαθέτει δικαιώματα διευθυντή (manager) ή διαχειριστή (administrator) και είναι προϊστάμενος τουλάχιστον μίας ομάδας, έχει τη δυνατότητα να δει τα αιτήματα όλων των υφισταμένων του, καθώς και να τα αποδεχτεί ή απορρίψει (Εικόνα 3.34).

Πατώντας στην κάρτα ανοίγει το Φύλλο αιτήματος, το οποίο περιέχει αναλυτικές πληροφορίες για το αίτημα που επέλεξε να δει ο χρήστης, όπως την ημερομηνία υποβολής, τον εργαζόμενο που υπέβαλε το αίτημα, τις ώρες ισχύος του αιτήματος και το σχόλιο που άφησε ο εργαζόμενος. Επιπλέον, παρουσιάζονται τα στοιχεία της έγκρισης του αιτήματος, όπως τον προϊστάμενο που αποδέχτηκε ή απέρριψε το αίτημα, την ημερομηνία έγκρισης και το σχόλιο που άφησε ο προϊστάμενος.

3.4 Σχεδίαση σκαριφημάτων και πρωτοτύπων

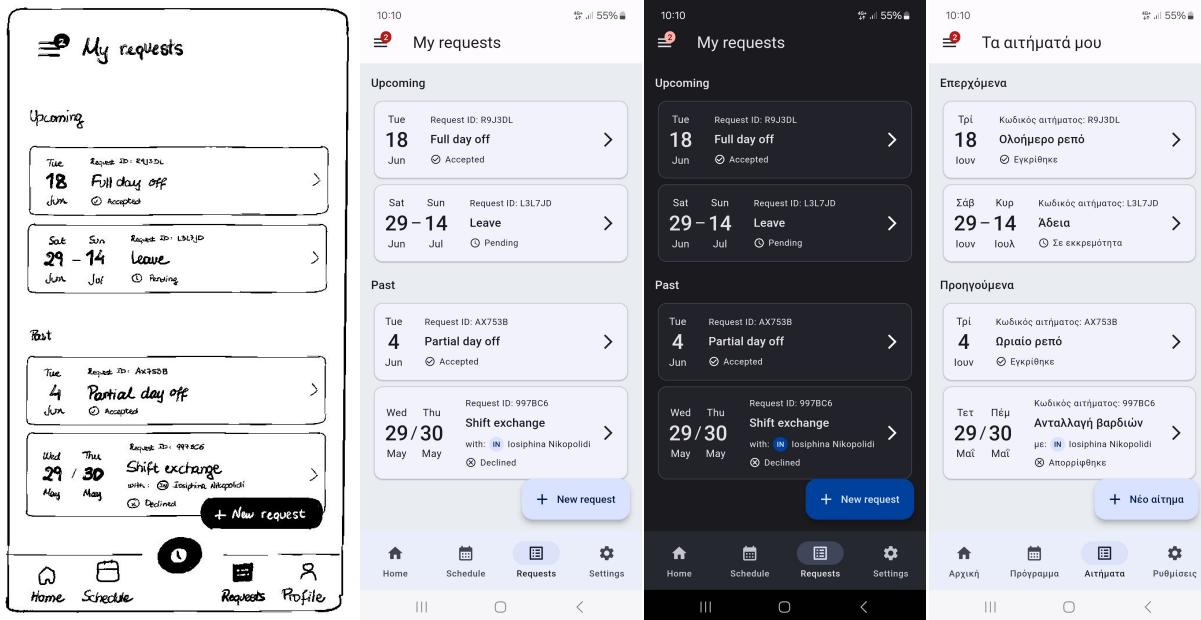


Εικόνα 3.31. Οδηγός προγράμματος: Βήμα 3 (Επιβεβαίωση δημοσίευσης): σκαριφηματικό πρωτότυπο

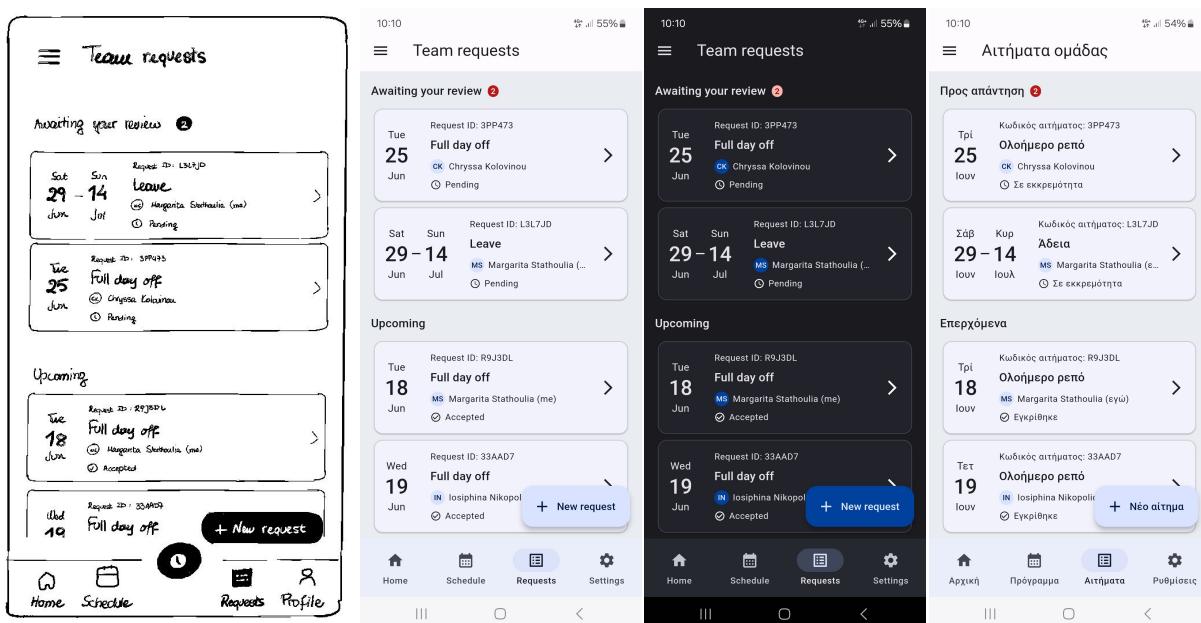


Εικόνα 3.32. Οδηγός προγράμματος: Ολοκλήρωση: σκαριφηματικό πρωτότυπο

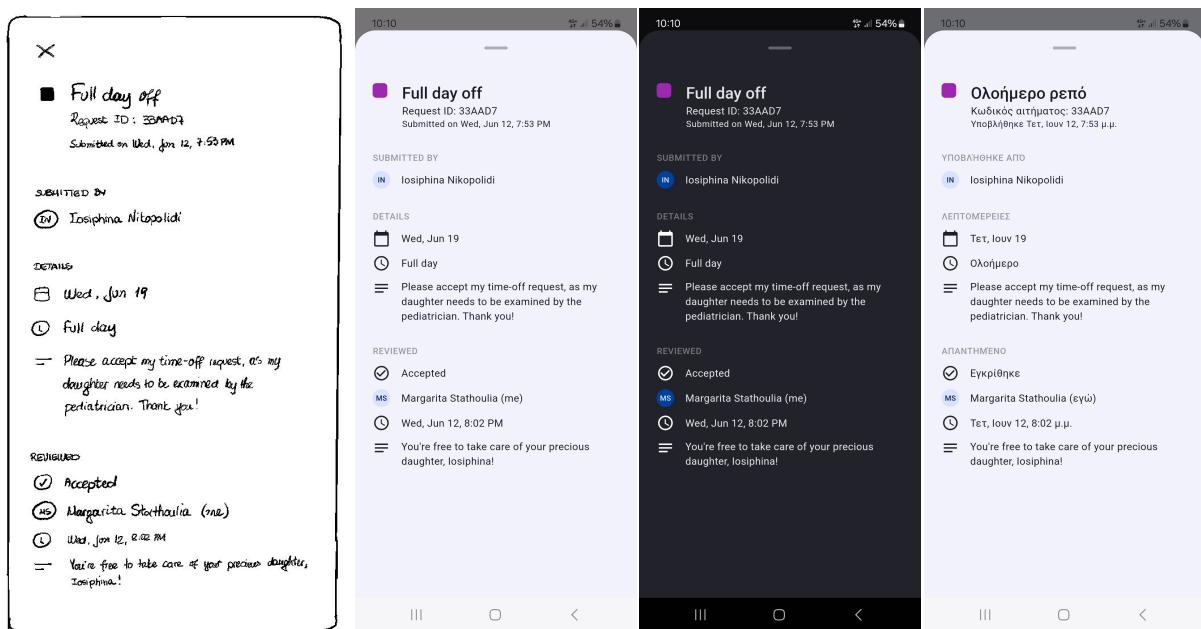
3.4 Σχεδίαση σκαριφημάτων και πρωτοτύπων



Εικόνα 3.33. Η Οθόνη αιτημάτων: σκαριφήμα και πρωτότυπα



Εικόνα 3.34. Η Οθόνη αιτημάτων για αιτήματα ομάδας: σκαριφήμα και πρωτότυπα



Εικόνα 3.35. Το Φύλλο αιτήματος: σκαρίφημα και πρωτότυπα

Στην περίπτωση που ο τρέχων χρήστης διαθέτει δικαιώματα διευθυντή (manager) ή διαχειριστή (administrator) και το αίτημα δεν είναι εγκεκριμένο, το Φύλλο αιτήματος δίνει τη δυνατότητα στον χρήστη να το αποδεχτεί ή να το απορρίψει, αφήνοντας ένα σχόλιο για τον εργαζόμενο που το υπέβαλε (Εικόνα 3.36)

3.4.8 Οδηγός νέου αιτήματος (New request wizard)

Η διαδικασία υποβολής νέου αιτήματος παρέχεται στον χρήστη επίσης στη μορφή οδηγού (wizard).

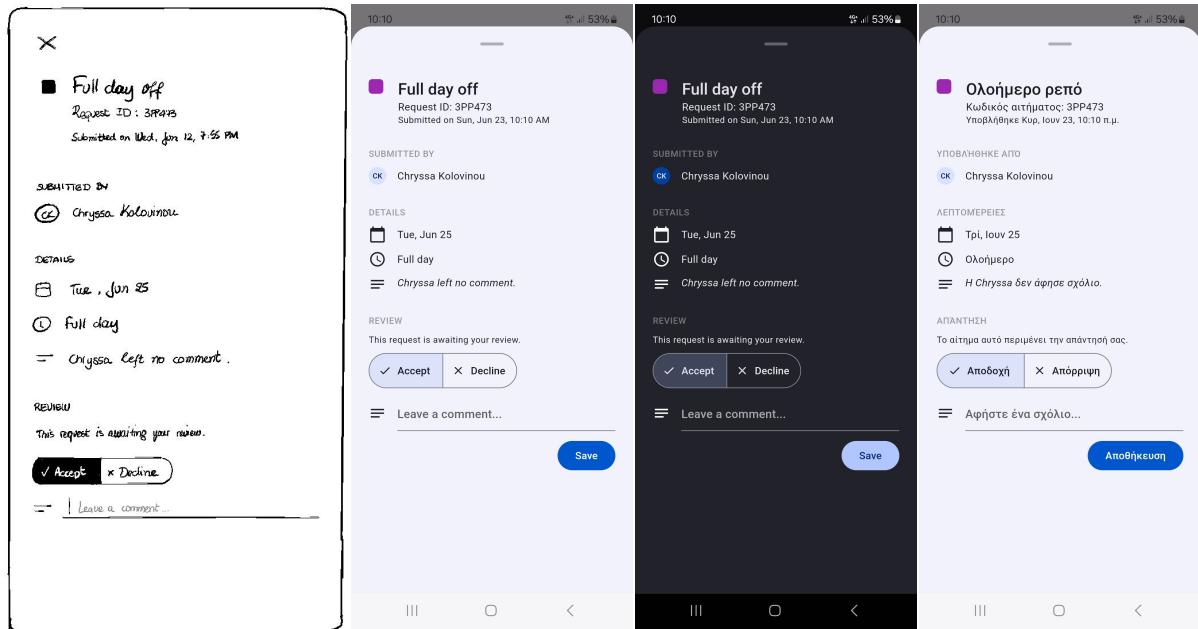
Βήμα 1: Επιλογή τύπου αιτήματος

Το πρώτο βήμα του οδηγού είναι η επιλογή τύπου αιτήματος. Ο χρήστης καλείται να επιλέξει μεταξύ αιτήματος άδειας ή αιτήματος ανταλλαγής βάρδιας (Εικόνα 3.37).

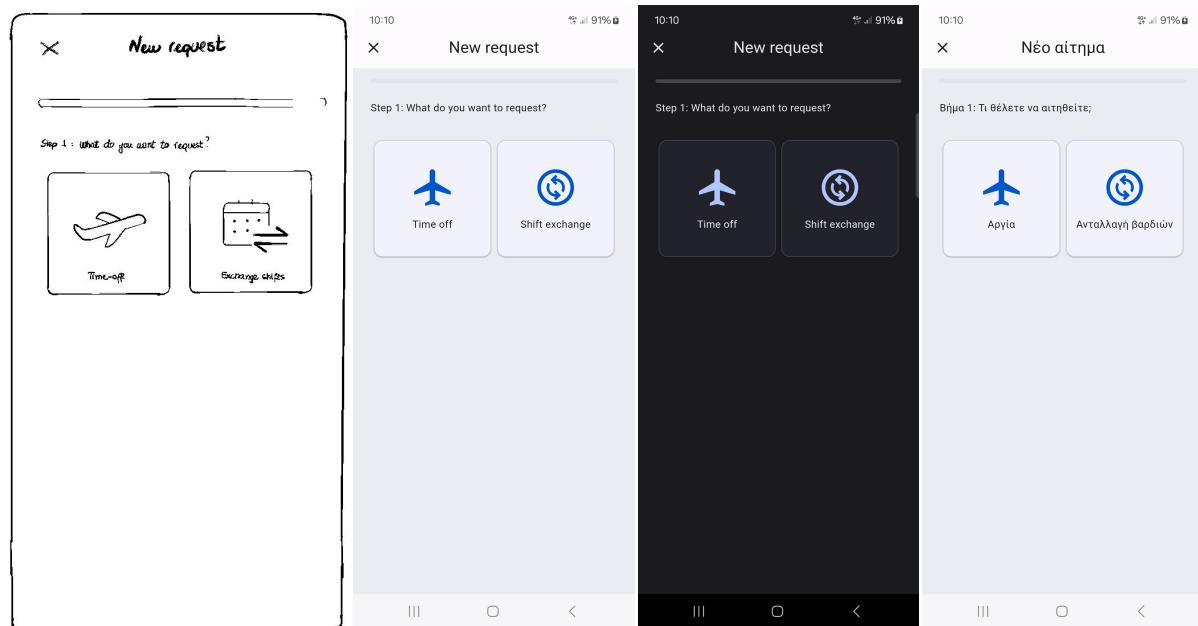
Βήμα 2 (Αίτημα άδειας): Επιλογή λεπτομερειών

Εάν ο χρήστης επιλέξει αίτημα άδειας, το επόμενο βήμα είναι η επιλογή ημερομηνιών και ωρών ισχύος. Εδώ, ο χρήστης μπορεί είτε να επιλέξει ένα εύρος πολλών ημερών, και άρα να αιτηθεί άδεια, είτε να επιλέξει μία μόνο ημέρα, και άρα να αιτηθεί ρεπό. Στην περίπτωση αυτή, μπορεί να αιτηθεί είτε για ολόκληρη την ημέρα, είτε για λίγες ώρες, καθορίζοντας ώρες έναρξης και λήξης (Εικόνα 3.38)

3.4 Σχεδίαση σκαριφημάτων και πρωτοτύπων

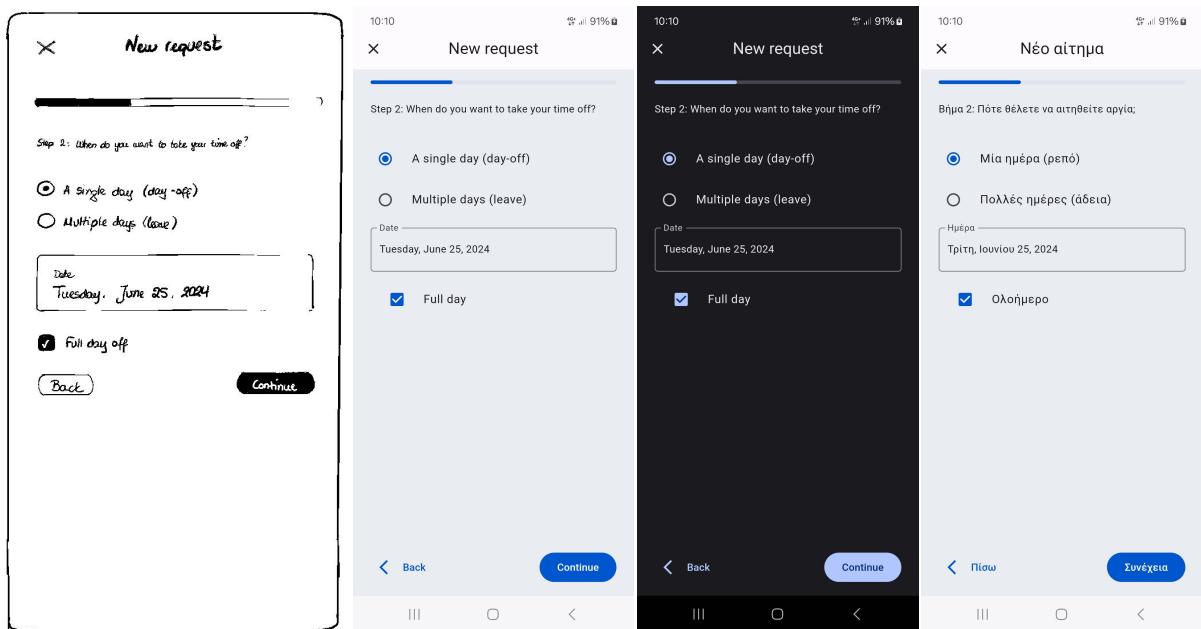


Εικόνα 3.36. Το Φύλλο αιτήματος για αίτημα προς έγκριση: σκαρίφημα και πρωτότυπα



Εικόνα 3.37. Οδηγός αιτήματος: Βήμα 1: σκαρίφημα και πρωτότυπα

3.4 Σχεδίαση σκαριφημάτων και πρωτοτύπων



Εικόνα 3.38. Οδηγός αιτήματος: Βήμα 2 (αίτημα άδειας): σκαρίφημα και πρωτότυπα

Βήματα 2-3 (Αίτημα ανταλλαγής βαρδιών): Επιλογή βαρδιών

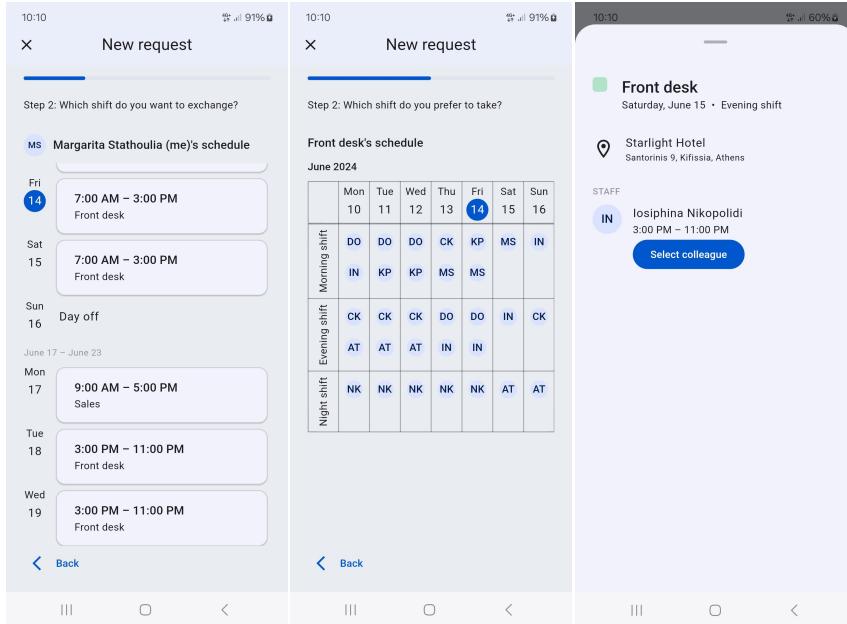
Εάν ο χρήστης επιλέξει αίτημα ανταλλαγής βαρδιών, τα επόμενα βήματα είναι η επιλογή των βαρδιών. Στο δεύτερο βήμα, στον οδηγό εμφανίζεται η Ατομική προβολή της Οθόνης προγράμματος, ώστε ο χρήστης να επιλέξει προς ανταλλαγή μία βάρδια που του έχει ανατεθεί (Εικόνα 3.39). Στο τρίτο βήμα, στον οδηγό εμφανίζεται η Ομαδική προβολή της Οθόνης προγράμματος, για την ομάδα στην οποία ανήκει η βάρδια που επέλεξε προηγουμένως ο χρήστης. Πατώντας πάνω σε κάποιο κελί, ο χρήστης μπορεί να επιλέξει τη βάρδια που επιθυμεί να αναλάβει, και επιπλέον τον συνάδελφο με τον οποίο θα πραγματοποιήσει την ανταλλαγή (Εικόνα 3.39).

Βήμα 3/4: Επισκόπηση και υποβολή

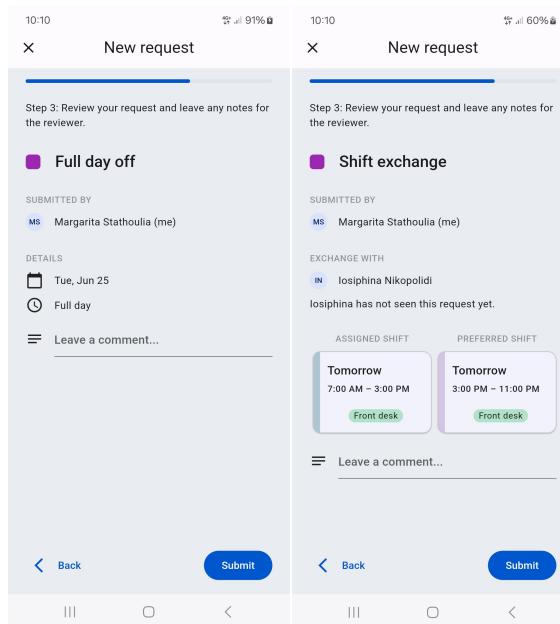
Στο τελευταίο βήμα, στον χρήστη εμφανίζεται μία επισκόπηση του αιτήματος που έχει δημιουργήσει και στο οποίο μπορεί να προσθέσει ένα σχόλιο. Ουσιαστικά, πρόκειται για το Φύλλο αιτήματος, όπως παρουσιάστηκε στην Ενότητα 3.4.7 (Εικόνα 3.40).

Πατώντας *Υποβολή (Submit)* το αίτημα του χρήστη καταχωρείται, οι προϊστάμενοί του ειδοποιούνται και στον χρήστη εμφανίζεται μήνυμα επιτυχίας (Εικόνα 3.41). Έπειτα, γίνεται επιστροφή στην Οθόνη αιτημάτων.

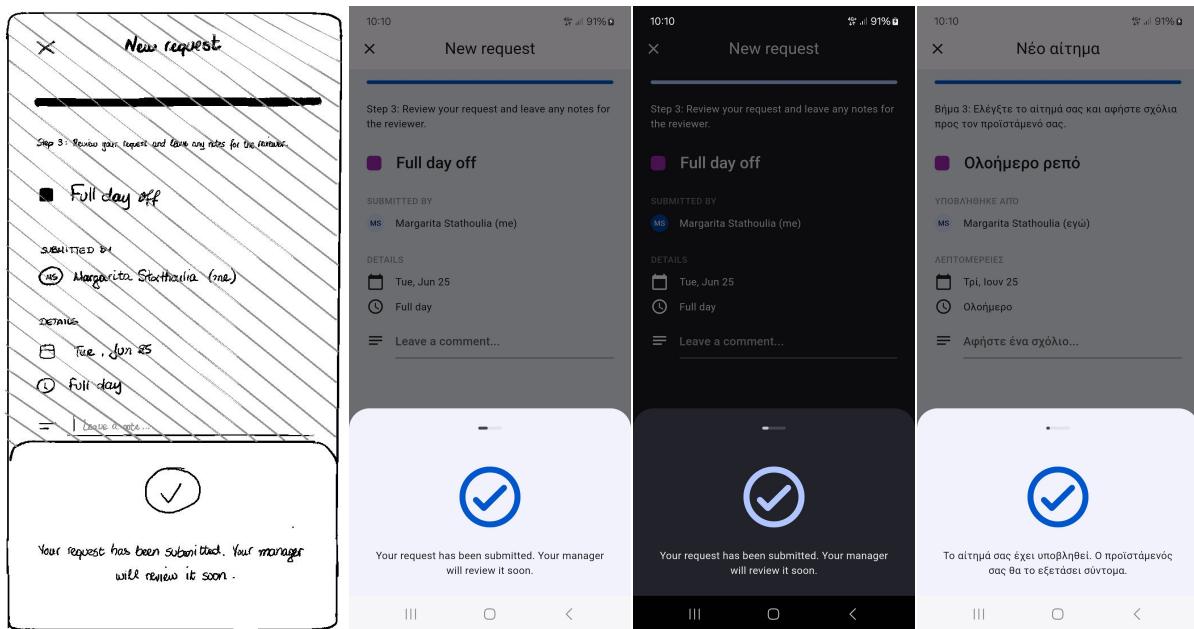
3.4 Σχεδίαση σκαριφημάτων και πρωτοτύπων



Εικόνα 3.39. Οδηγός αιτήματος: Βίματα 2-3 (αίτημα ανταλλαγής βαρδιών): πρωτότυπα



Εικόνα 3.40. Οδηγός αιτήματος: Βίμα 3/4: πρωτότυπα



Εικόνα 3.41. Οδηγός αιτήματος: Ολοκλήρωση: σκαριφηματικό πρωτότυπο

3.4.9 Οθόνη ρυθμίσεων (Settings page)

Η Οθόνη ρυθμίσεων (Εικόνα 3.42) εξυπηρετεί τρεις σκοπούς:

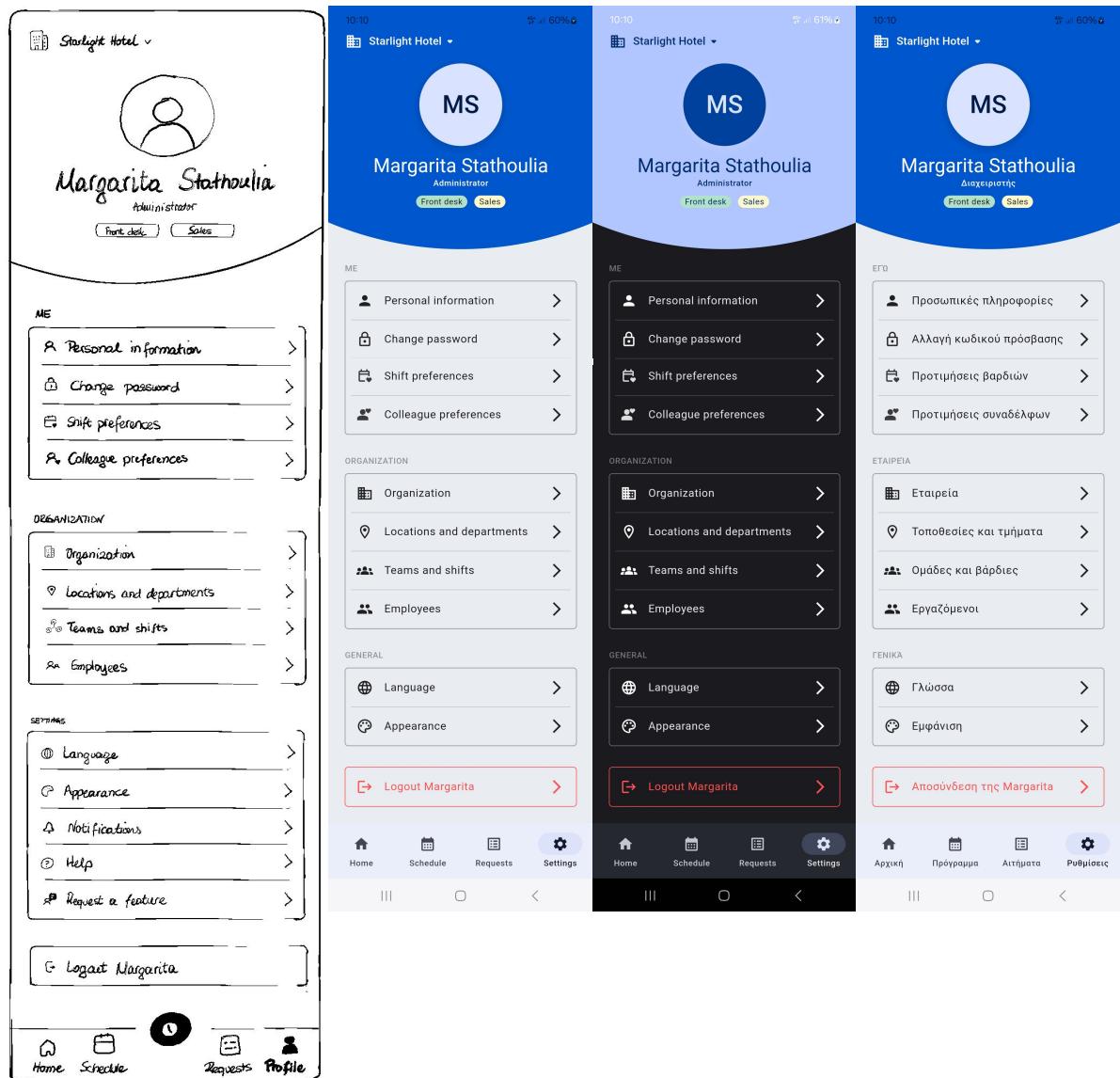
- την προβολή και διαχείριση του προφίλ του τρέχοντος χρήστη,
- την εναλλαγή του τρέχοντος οργανισμού, σε περίπτωση που ο χρήστης ανήκει σε περισσότερους από έναν οργανισμό,
- τη διαχείριση του οργανισμού, σε περίπτωση που ο χρήστης διαθέτει δικαιώματα διευθυντή (manager) ή διαχειριστή (administrator).

Η οθόνη αυτή παρέχει την πρόσβαση στις επιμέρους οθόνες τοποθεσιών, ομάδων, εργαζομένων, περιορισμών βαρδιών εργαζομένου, προτιμήσεων βαρδιών εργαζομένου και προτιμήσεων συναδέλφων εργαζομένου.

3.4.10 Οθόνη τοποθεσιών (Locations page)

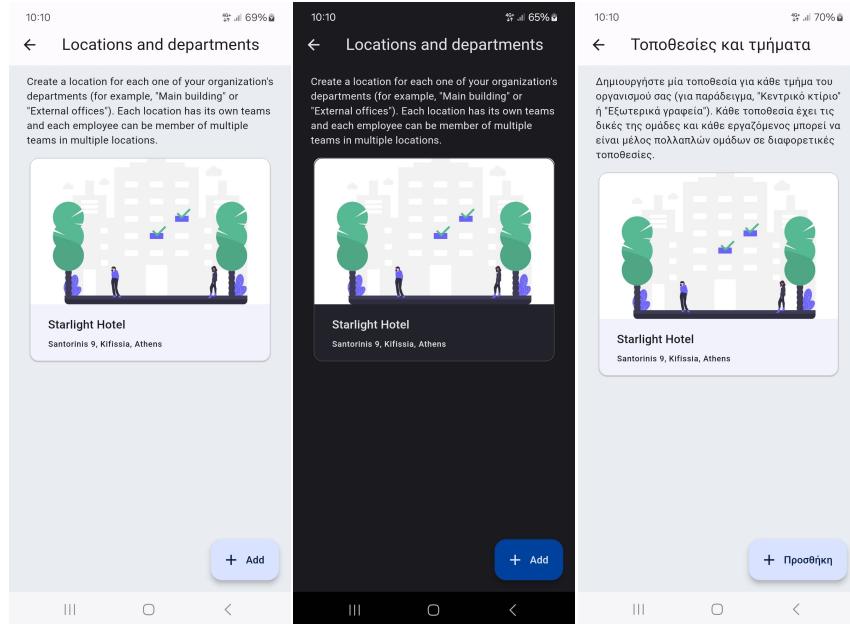
Στην Οθόνη τοποθεσιών (Εικόνες 3.43 και 3.44) έχουν πρόσβαση μόνο οι διαχειριστές (administrators) του οργανισμού. Σε αυτή την οθόνη, ο χρήστης μπορεί να επεξεργάζεται τοποθεσίες του οργανισμού. Ο κάθε οργανισμός αποτελείται από μία ή περισσότερες τοποθεσίες, οι οποίες περιλαμβάνουν ομάδες. Τα δεδομένα της τοποθεσίας περιλαμβάνουν το όνομα, την περιγραφή (προαιρετικά), τη διεύθυνσή της, καθώς και μία φωτογραφία και τις ομάδες που αυτή περιλαμβάνει.

3.4 Σχεδίαση σκαριφημάτων και πρωτοτύπων

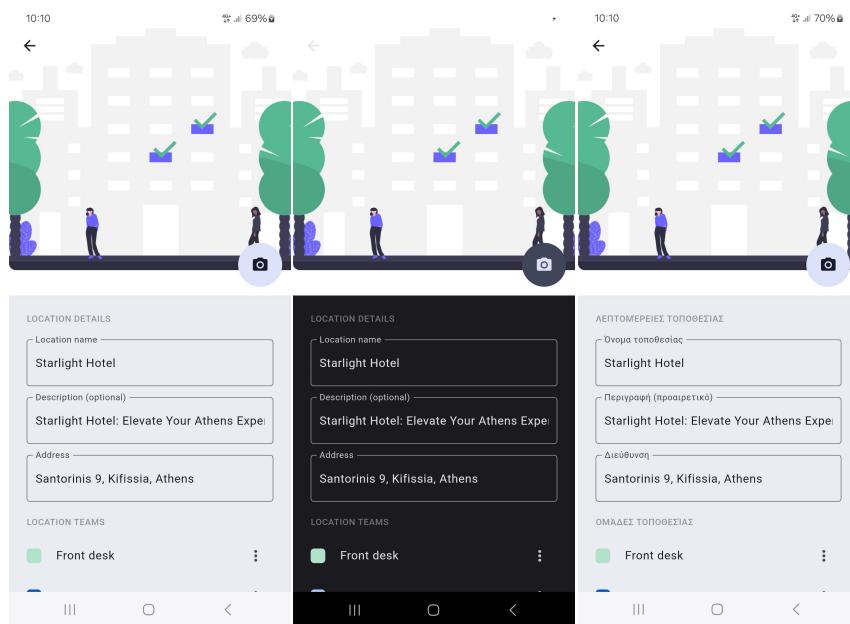


Εικόνα 3.42. Η Οθόνη ρυθμίσεων: σκαριφηματα και πρωτότυπα

3.4 Σχεδίαση σκαριφημάτων και πρωτότυπων



Εικόνα 3.43. Η Οθόνη τοποθεσιών: πρωτότυπα

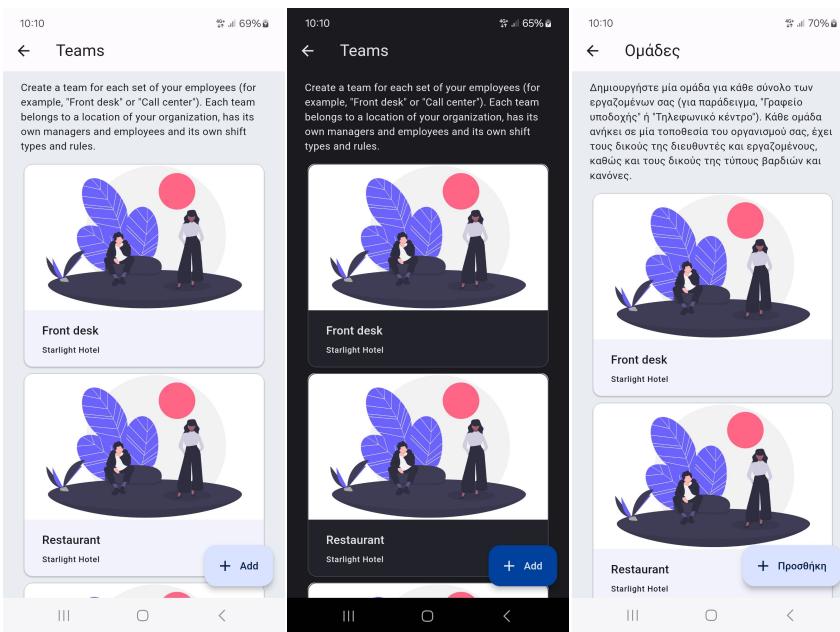


Εικόνα 3.44. Η Οθόνη επεξεργασίας τοποθεσίας: πρωτότυπα

3.4 Σχεδίαση σκαριφημάτων και πρωτοτύπων

3.4.11 Οθόνη ομάδων (Teams page)

Στην Οθόνη ομάδων (Εικόνες 3.45 και 3.46) έχουν πρόσβαση μόνο οι διαχειριστές (administrators) του οργανισμού. Σε αυτή την οθόνη, ο χρήστης μπορεί να επεξεργάζεται ομάδες του οργανισμού. Η κάθε ομάδα αποτελείται από το όνομά της, την τοποθεσία στην οποία ανήκει, μία φωτογραφία, ένα χρώμα που χρησιμοποιείται στον χρωματικό κώδικα της Οθόνης προγράμματος. Επιπλέον, στην οθόνη αυτή φαίνονται επίσης τα μέλη και οι διευθυντές της ομάδας, καθώς επίσης και οι τύποι βαρδιών που η ομάδα περιλαμβάνει. Από εδώ, υπάρχει επίσης η δυνατότητα επεξεργασίας των τύπων βαρδιών (Εικόνα 3.47).

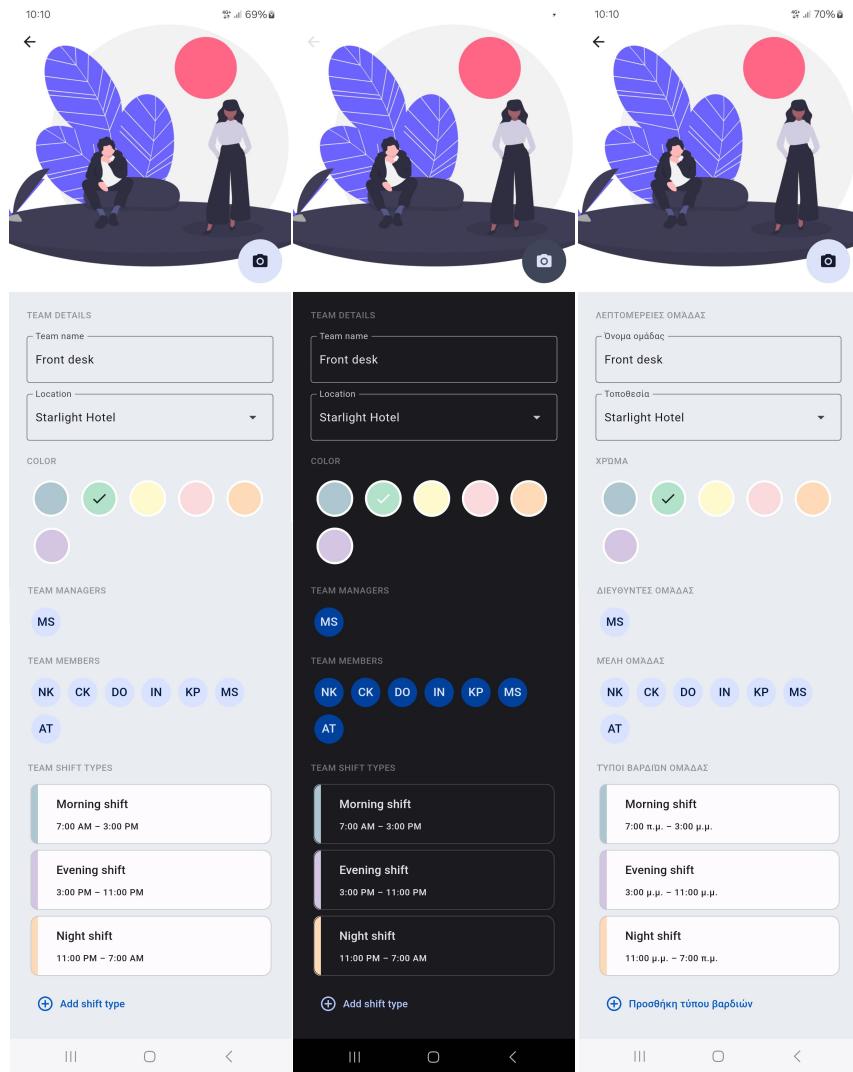


Εικόνα 3.45. Η Οθόνη ομάδων: πρωτότυπα

3.4.12 Οθόνη εργαζομένων (Employees page)

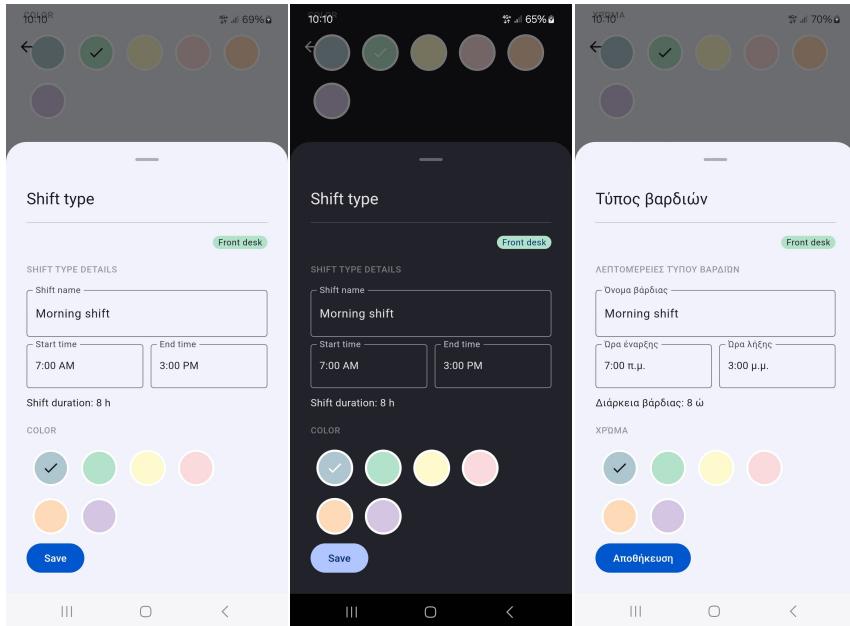
Στην Οθόνη εργαζομένων (Εικόνες 3.48 και 3.49) έχουν πρόσβαση μόνο οι διαχειριστές (administrators) του οργανισμού. Σε αυτή την οθόνη, ο χρήστης μπορεί να δει μία λίστα με όλους τους εργαζομένους του οργανισμού και να τους επεξεργαστεί. Οι πληροφορίες κάθε υπαλλήλου περιλαμβάνουν το όνομα και το επώνυμό του, το φύλο του (που χρησιμοποιείται στην Ελληνική μετάφραση), τις ομάδες στις οποίες ανήκει και τις οποίες διαχειρίζεται, τους περιορισμούς βαρδιών, τις προτιμήσεις βαρδιών και τις προτιμήσεις συναδέλφων.

3.4 Σχεδίαση σκαριφημάτων και πρωτότυπων

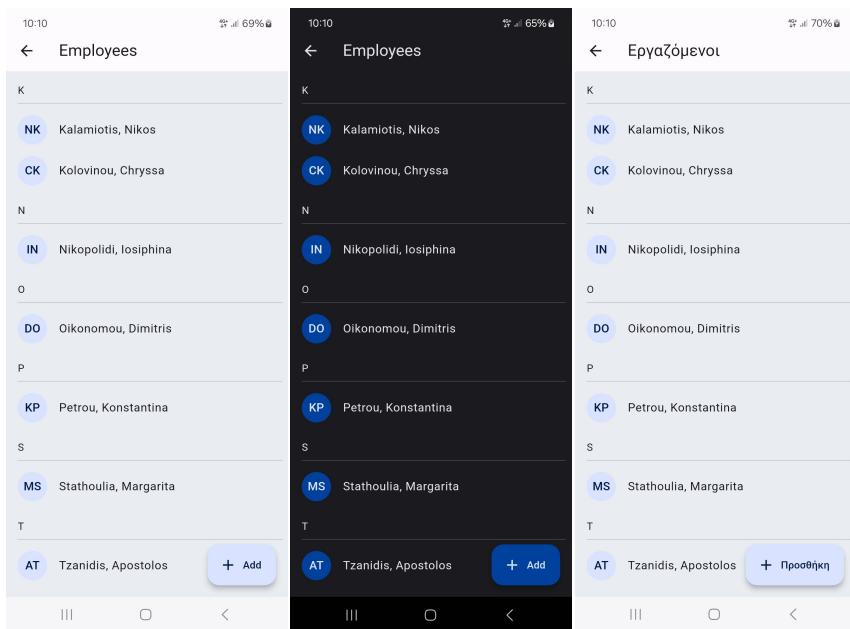


Εικόνα 3.46. Η Οθόνη επεξεργασίας ομάδας: πρωτότυπα

3.4 Σχεδίαση σκαριφημάτων και πρωτοτύπων

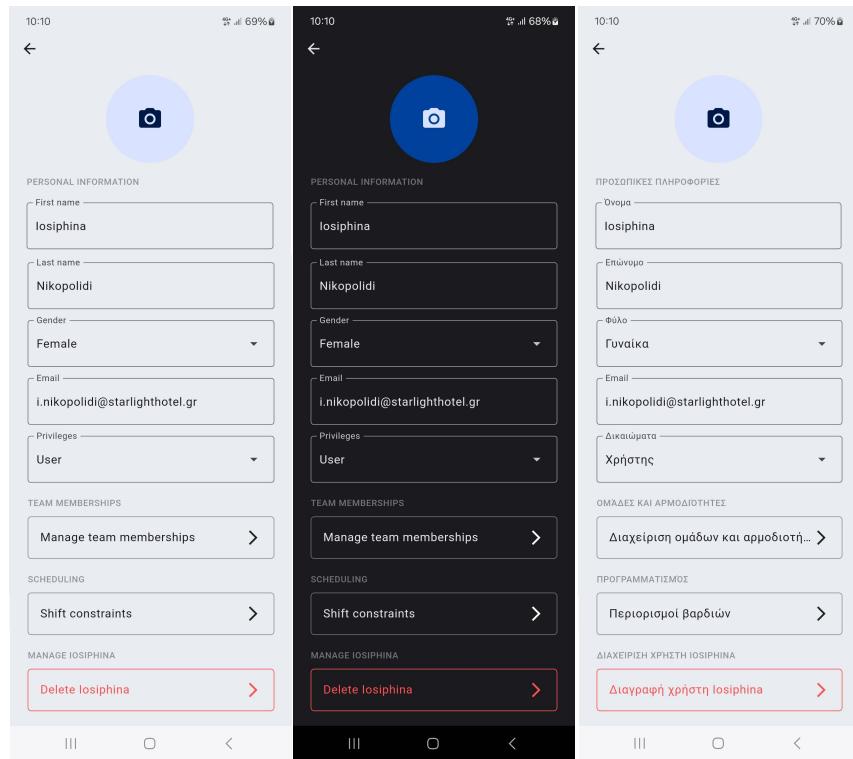


Εικόνα 3.47. Το Φύλλο τύπου βαρδιών: πρωτότυπα



Εικόνα 3.48. Η Οθόνη εργαζομένων: πρωτότυπα

3.4 Σχεδίαση σκαριφημάτων και πρωτότυπων

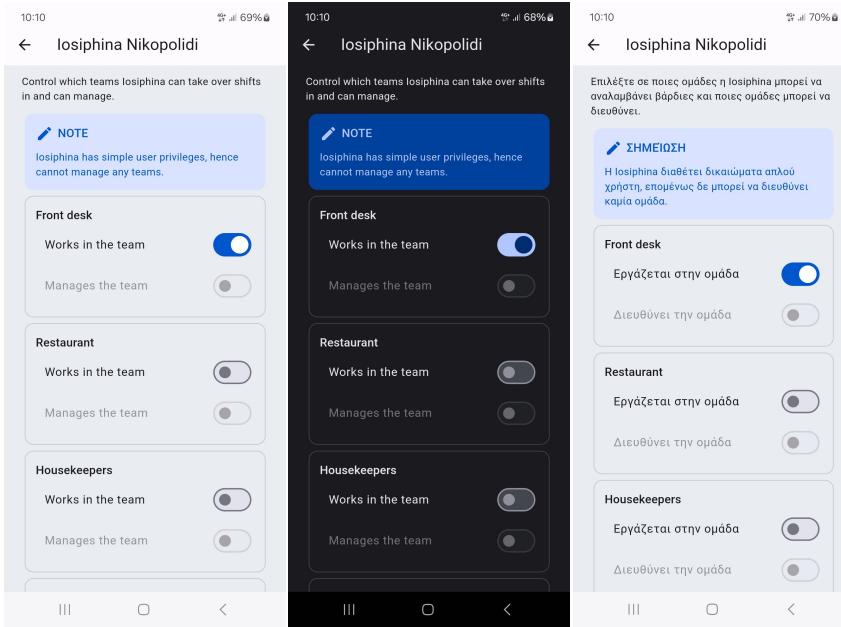


Εικόνα 3.49. Η Οθόνη επεξεργασίας εργαζομένου: πρωτότυπα

3.4 Σχεδίαση σκαριφημάτων και πρωτοτύπων

3.4.13 Οθόνη συμμετοχής εργαζομένου σε ομάδες (Employee team membership page)

Στην Οθόνη συμμετοχής εργαζομένου σε ομάδες (Εικόνα 3.50) έχουν πρόσβαση μόνο οι διαχειριστές (administrators) του οργανισμού. Σε αυτή την οθόνη, ο χρήστης μπορεί να επιλέξει σε ποιες ομάδες ανήκει ο επιλεγμένος εργαζόμενος και ποιες αυτός διαχειρίζεται.



Εικόνα 3.50. Η Οθόνη συμμετοχής εργαζομένου σε ομάδες: πρωτότυπα

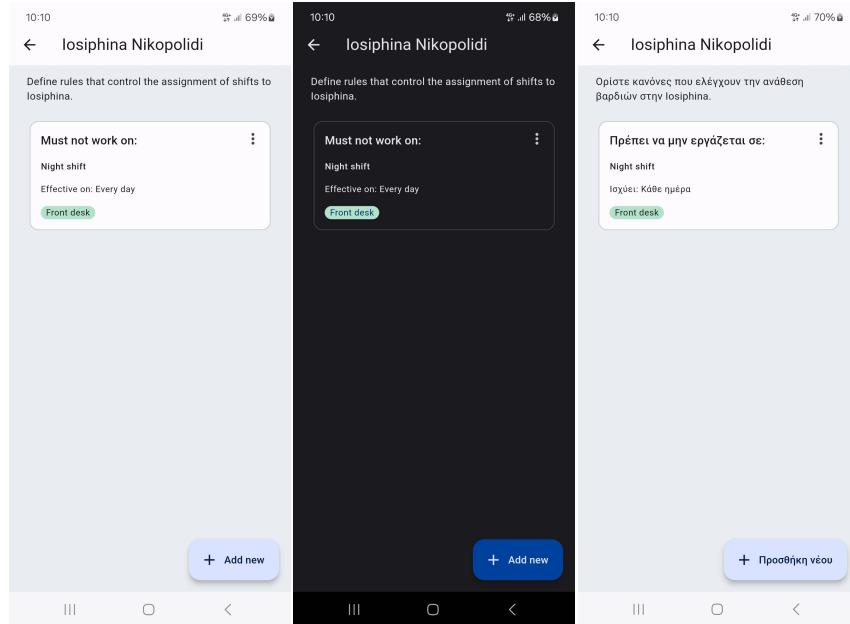
3.4.14 Οθόνη περιορισμών βαρδιών εργαζομένου (Shift constraints page)

Στην Οθόνη περιορισμών βαρδιών εργαζομένου (Εικόνες 3.51 και 3.52) έχουν πρόσβαση μόνο οι διευθυντές (managers) και οι διαχειριστές (administrators) του οργανισμού. Σε αυτή την οθόνη, ο χρήστης μπορεί να ορίσει τους κανόνες που διέπουν την αυτόματη ανάθεση του εργαζομένου σε βάρδιες. Ο κάθε κανόνας-περιορισμός εκφράζεται με τη μορφή: Ο …πρέπει να / να μην εργάζεται, τις ημέρες …, στην ομάδα …, στη βάρδια ….

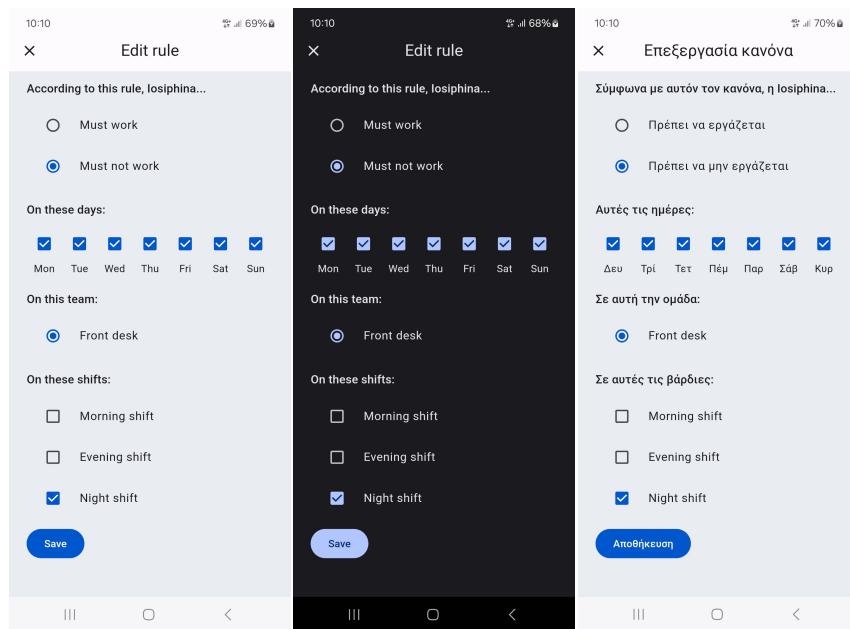
3.4.15 Οθόνη προτιμήσεων βαρδιών εργαζομένου (Shift preferences page)

Στην Οθόνη προτιμήσεων βαρδιών εργαζομένου (Εικόνα 3.53) έχει πρόσβαση μόνο ο τρέχων χρήστης για τον λογαριασμό του. Σε αυτή την οθόνη, ο χρήστης μπορεί να ορίσει τις προτιμήσεις σε βάρδιες, με βάση τις οποίες γίνεται η αυτόματη ανάθεσή του σε βάρδιες. Η κάθε προτίμηση αποτελείται από το όνομα της βάρδιας και μία τιμή του συντελεστή προτίμησης. Ο συντελεστής προτίμησης μπορεί να

3.4 Σχεδίαση σκαριφημάτων και πρωτότυπων



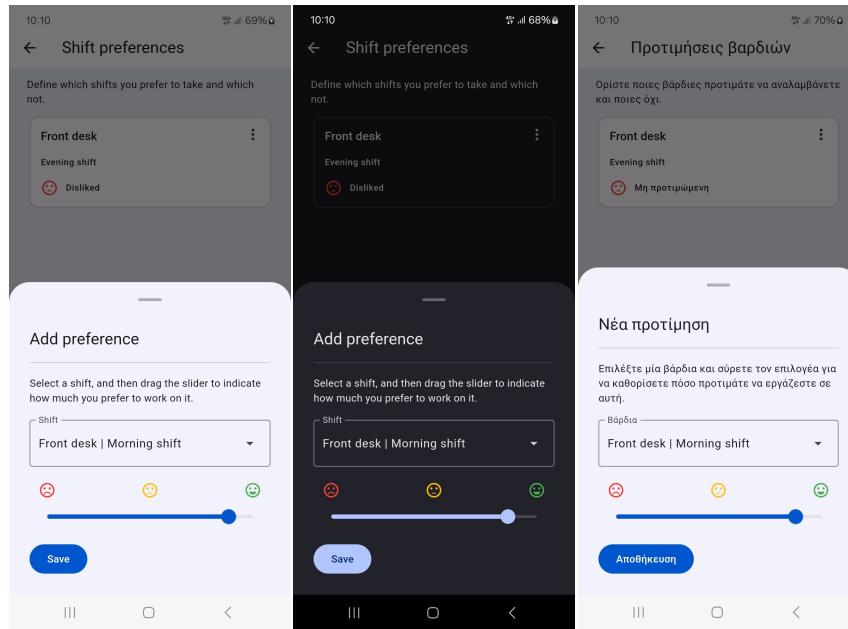
Εικόνα 3.51. Η Οθόνη περιορισμών βαρδιών εργαζομένου: πρωτότυπα



Εικόνα 3.52. Η Οθόνη επεξεργασίας περιορισμού βαρδιών εργαζομένου: πρωτότυπα

3.5 Σύνοψη

είναι είτε θετικός (οπότε ο χρήστης προτιμά να εργάζεται στη συγκεκριμένη βάρδια), είτε αρνητικός (οπότε ο χρήστης προτιμά να μην εργάζεται στη συγκεκριμένη βάρδια). Η επιλογή γίνεται από έναν δρομέα, με τρία εικονίδια προσώπου: στην αριστερή άκρη ένα κακοδιάθετο, κόκκινο πρόσωπο (που εκφράζει αρνητικά συναισθήματα), στη δεξιά άκρη ένα καλοδιάθετο, πράσινο πρόσωπο (που εκφράζει θετικά συναισθήματα) και στη μέση ένα ουδέτερο, κίτρινο πρόσωπο (που εκφράζει ουδέτερη στάση).



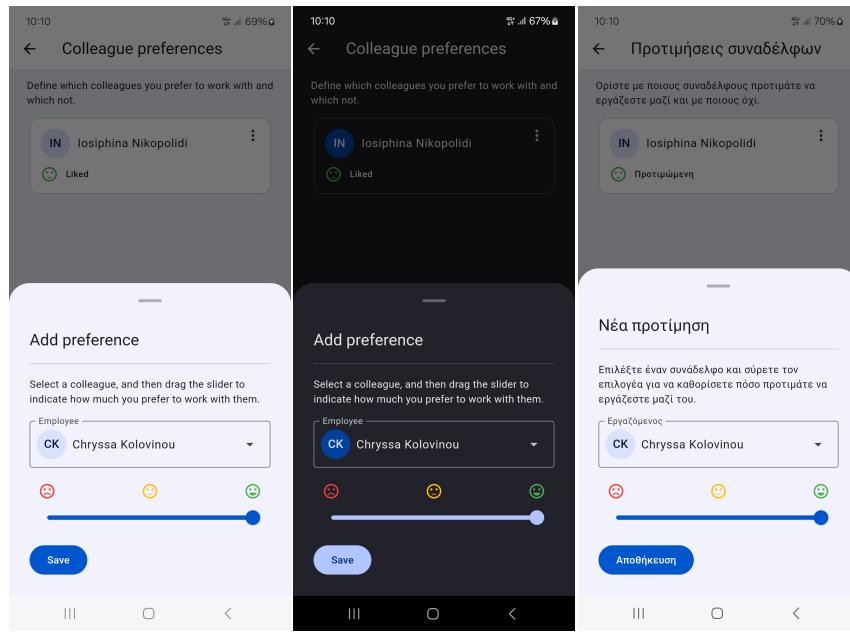
Εικόνα 3.53. Η Οθόνη προτιμήσεων βαρδιών εργαζομένου: σκαρίφημα και πρωτότυπα

3.4.16 Οθόνη προτιμήσεων συναδέλφων εργαζομένου (Colleague preferences page)

Στην Οθόνη προτιμήσεων συναδέλφων εργαζομένου (Εικόνα 3.54) έχει πρόσβαση μόνο ο τρέχων χρήστης για τον λογαριασμό του. Όπως και στην Οθόνη προτιμήσεων βαρδιών εργαζομένου, ο χρήστης μπορεί με την ίδια ακριβώς διαδικασία να καθορίσει με ποιους συναδέλφους προτιμά να εργάζεται και με ποιους όχι, επιλέγοντας από μία λίστα με τους εργαζόμενους, με τους οποίους βρίσκονται στις ίδιες ομάδες.

3.5 Σύνοψη

Κατά τη διαδικασία δημιουργίας ενός προϊόντος, σημαντικό ρόλο παίζει η σωστή προετοιμασία και σχεδίαση. Στο Κεφάλαιο αυτό, πραγματοποιήθηκε ανάλυση των απαιτήσεων που καλείται να επιλύσει η εφαρμογή Horario, με τις μεθόδους του ανθρωποκεντρικού σχεδιασμού. Συγκεκριμένα,



Εικόνα 3.54. Η Οθόνη προτιμήσεων συναδέλφων εργαζομένου: σκαρίφημα και πρωτότυπα

έγινε ανάλυση των χρηστών και των εργασιών της εφαρμογής, χρησιμοποιώντας το πλαίσιο PACT, τη μέθοδο των περσόνων (personas) και την ιεραρχική ανάλυση εργασιών. Στη συνέχεια, αφού μελετήθηκαν ανταγωνιστικές λύσεις και πάρθηκαν σχεδιαστικές αποφάσεις, σχεδιάστηκαν σκαριφήματα και πρωτότυπα για τις διάφορες οιθόνες της εφαρμογής, τα οποία και σχολιάστηκαν εκτενώς.

4 Υλοποίηση της εφαρμογής

Στο παρόν Κεφάλαιο, θα συζητηθούν θέματα που αφορούν την ανάπτυξη της εφαρμογής **Horario**. Έχοντας πλέον έτοιμα τα σχέδια της γραφικής διεπαφής σε κώδικα Flutter (βλ. Ενότητα 3.4), κληθήκαμε να πάρουμε αποφάσεις αρχιτεκτονικής λογισμικού και, με βάση αυτές, να αναπτύξουμε μία πλήρως λειτουργική, cross-platform και βασισμένη στο cloud, εφαρμογή.

4.1 Αρχιτεκτονική συστήματος

Στην Εικόνα 4.1 παριστάνεται η πλήρης, υψηλού επιπέδου αρχιτεκτονική του συστήματος της εφαρμογής **Horario**. Το full-stack σύστημα χωρίζεται σε δύο επιμέρους τμήματα, το front-end και το back-end.

Ο χρήστης αλληλεπιδρά με την εφαρμογή μέσω του front-end. Στην Ενότητα 4.2, εξηγείται η επιμέρους αρχιτεκτονική του front-end, η οποία βασίζεται στο μοτίβο Model-View-ViewModel (MVVM).

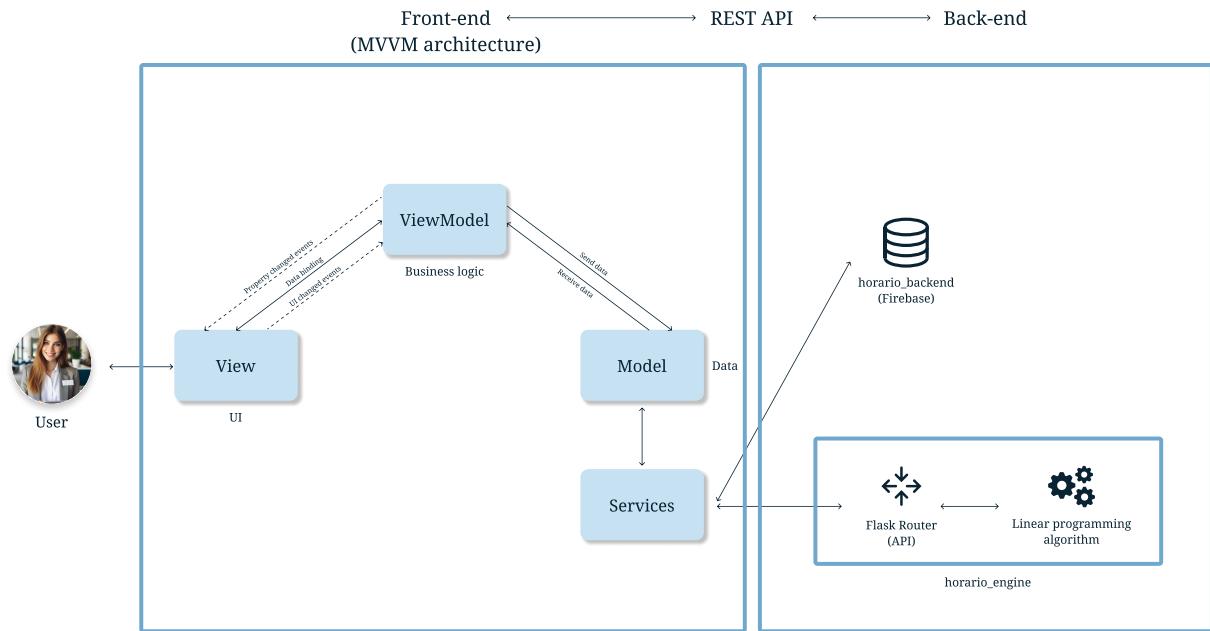
Τα δεδομένα που προσφέρονται στον χρήστη από την εφαρμογή μέσω του front-end, αλλά και αυτά που εισάγει ο χρήστης στην εφαρμογή αποθηκεύονται και διαχειρίζονται από το back-end, που αποτελεί ένα ανεξάρτητο τμήμα του συστήματος. Το back-end υποστηρίζεται από δύο υπηρεσίες. Στις Ενότητες 4.4 και 4.6, αναλύεται η μία από αυτές, το **Horario Backend**, η οποία προσφέρει την αποθήκευση και διαχείριση των δεδομένων, αλλά και άλλες λειτουργικότητες, μέσω της υπηρεσίας Firebase. Στην Ενότητα 4.7, αναλύεται η δεύτερη back-end υπηρεσία, το **Horario Engine**, η οποία αποτελείται από έναν εξυπηρετητή που φιλοξενεί τη μηχανή αυτόματης ανάθεσης βαρδιών.

Τα δύο υποσυστήματα, front-end και back-end, επικοινωνούν μέσω REST APIs, χρησιμοποιώντας αιτήματα και αποκρίσεις του πρωτοκόλλου HTTP.

4.2 Η αρχιτεκτονική Model-View-ViewModel (MVVM)

Ήδη στην Ενότητα 2.1.3, αναφέρθηκε το “widget” ως στοιχειώδης μονάδα τμήματος κώδικα μίας εφαρμογής, με το framework Flutter. Κάθε οθόνη της εφαρμογής είναι ένα widget, το οποίο αποτελείται από άλλα, επιμέρους widgets, όπως κουμπιά, κείμενα, εικόνες κ.λπ. Τα widgets αποτελούνται επιπλέον

4.2 Η αρχιτεκτονική Model-View-ViewModel (MVVM)



Εικόνα 4.1. Η πλήρης, υψηλού επιπέδου αρχιτεκτονική της εφαρμογής

από συναρτήσεις - μεθόδους, τα αποτελέσματα των οποίων μπορούν να χρησιμοποιηθούν, ώστε να προβληθούν στον χρήστη.

Σε αυτό το πνεύμα, είναι απολύτως αποδεκτό ολόκληρη η λογική της εφαρμογής να τοποθετηθεί σε μεθόδους των αντίστοιχων widgets, ωστόσο κάτι τέτοιο αποτελεί κακή προγραμματιστική και αρχιτεκτονική πρακτική, μιας και οδηγεί γρήγορα σε πολύπλεξη του κώδικα και επαναχρησιμοποίηση τμημάτων λογικής. Για τον λόγο αυτό, έχουν αναπτυχθεί κάποιες αρχιτεκτονικές ή μοτίβα (design patterns), όπως το Model-View-Controller (MVC), το Model-View-ViewModel (MVVM) και το Model-View-Presenter (MVP) (Lou 2016).

Για τη δόμηση της εφαρμογής Horario επιλέξαμε την αρχιτεκτονική Model-View-ViewModel (MVVM). Το MVVM είναι μία παραλλαγή της γηραιότερης αρχιτεκτονικής MVC, που προτάθηκε το 2005 από αρχιτέκτονες λογισμικού της Microsoft, με σκοπό να κάνει την ανάπτυξη Windows προγραμμάτων, των μορφών WPF (Windows Presentation Foundation) και Silverlight, ευκολότερη (Gossman 2005). Η φιλοσοφία της αρχιτεκτονικής MVVM στηρίζεται στον διαχωρισμό των αρμοδιοτήτων (separation of concerns), μεταξύ των επιμέρους τμημάτων κώδικα.

Η αρχιτεκτονική MVVM αποτελείται από τα ακόλουθα 3 τμήματα (García 2023), (Anderson 2012):

- **Model:** είναι το τμήμα κώδικα που χειρίζεται τα δεδομένα και επικοινωνεί με τις μονάδες αποθήκευσης. Αρμοδιότητα του Model είναι να μεταφέρει δεδομένα από και προς τη βάση δεδομένων, επομένως εξαρτάται άμεσα από την επιλογή του συστήματος διαχείρισης βάσης

δεδομένων. Κάθε Model θα πρέπει να υλοποιεί ένα interface, ώστε τα δεδομένα που καθιστά προσβάσιμα να είναι μορφολογίας συμβατής με τα υπόλοιπα τμήματα του MVVM.

- **View:** είναι το τμήμα κώδικα με το οποίο έρχεται σε επαφή ο χρήστης. Το View σχεδιάζει τις γραφικές διεπαφές της εφαρμογής (δηλαδή τις επιμέρους οθόνες), φροντίζοντας πάντα να εμφανίζεται στον χρήστη ανανεωμένη πληροφορία. Το View χρησιμοποιεί δεδομένα συγκεκριμένης μορφής, τα οποία προετοιμάζει το Model (με βάση το interface που προαναφέρθηκε), χωρίς όμως αυτά τα δύο τμήματα να επικοινωνούν απευθείας μεταξύ τους.
- **ViewModel:** είναι το ενδιάμεσο τμήμα κώδικα μεταξύ Model και View. Το View συμπλέκεται σε δεδομένα του ViewModel (data binding), όπως ιδιότητες (properties) και πεδία (fields), και τα οποία καθορίζουν τη συμπεριφορά και τις λειτουργίες του View. Για παράδειγμα, ένα κουμπί του View συμπλέκεται σε μία μέθοδο του ViewModel, η οποία καλείται όταν ο χρήστης πατάει το κουμπί. Ένα αντίστοιχο data binding με ένα property του ViewModel μπορεί να καθορίσει εάν το κουμπί είναι ενεργοποιημένο ή απενεργοποιημένο.

Σκοπός της αρχιτεκτονικής MVVM είναι η πλήρης αποσύμπλεξη του Model από το View και η επικοινωνία τους μονάχα μέσω του ViewModel. Οι σχέσεις μεταξύ Model, View και ViewModel είναι πολλά-προς-πολλά (many-to-many). Έτσι, ένα View μπορεί να συμπλέκεται με περισσότερα από ένα ViewModels και σε ένα ViewModel να συμπλέκονται περισσότερα από ένα Views. Επιπλέον, με αυτόν τον τρόπο, όλο το front-end της εφαρμογής είναι πλήρως ανεξάρτητο από το back-end, με την έννοια ότι, εάν κάποια στιγμή οι αρχιτέκτονες αποφασίσουν να μεταναστεύσουν (migrate) σε ένα διαφορετικό σύστημα διαχείρισης βάσης δεδομένων, οι προγραμματιστές θα αρκεί να υλοποιήσουν το interface του Model για το αντίστοιχο σύστημα, αφήνοντας το View και το ViewModel αναλλοίωτα.

Ας εξετάσουμε, επί παραδείγματι, την Αρχική οθόνη της εφαρμογής **Horario** (βλ. Ενότητα 3.4.1). Θεωρώντας την ως ένα View, χρειάζεται ως δεδομένα τον ενεργό χρήστη και τις επερχόμενες βάρδιες του. Οι επερχόμενες βάρδιες αποτελούν ιδιότητα του ViewModel της Οθόνης προγράμματος, αφού όλες οι σχετικές με πρόγραμμα λειτουργίες επιτελούνται από το συγκεκριμένο ViewModel. Έτσι, θα ήταν εσφαλμένη πρακτική να προσθέσουμε business logic στο ViewModel της Αρχικής οθόνης για την ανάκτηση των επερχόμενων βαρδιών, μιας και θα οδηγούσε σε επαναλαμβανόμενο κώδικα. Αντίθετα, το Home-Page-View κάνει χρήση ιδιότητας του Schedule-Page-ViewModel, οδηγώντας σε ξεκάθαρο διαχωρισμό αρμοδιοτήτων (separation of concerns). Από την αντίθετη οπτική, στην ίδια ιδιότητα του Schedule-Page-ViewModel συμπλέκεται τόσο η Αρχική οθόνη, για τις επερχόμενες βάρδιες, αλλά και η Οθόνη προγράμματος, για την Ατομική της προβολή.

Ας εστιάσουμε σε ένα βασικό ζητούμενο της αρχιτεκτονικής MVVM. Η σύμπλεξη δεδομένων (data binding) μεταξύ View και ViewModel πρέπει να είναι διαρκώς ενημερωμένη. Αυτό σημαίνει ότι μία αλλαγή που προκαλείται σε μία ιδιότητα (property) του ViewModel, είτε προκαλείται από μέθοδο του ίδιου του ViewModel, είτε από δεδομένα που προήλθαν από το Model, θα πρέπει να αντικατοπτρίζεται αμέσως στο View, χωρίς να εμφανίζονται ασυμφωνίες.

4.3 Μοντέλο δεδομένων

Επομένως, το ViewModel πρέπει πάντα να “ειδοποιεί” (notify) το View σχετικά με αλλαγές στις ιδιότητές του, ή, με άλλα λόγια, οι ιδιότητες του ViewModel να είναι “παρατηρήσιμες” (observable) από το View. Η λειτουργικότητα αυτή αποτελεί βασικό στοιχείο της αρχιτεκτονικής MVVM και, γι' αυτό, υλοποιείται από βιβλιοθήκες στα περισσότερα frameworks που στηρίζονται στο MVVM.

Για το Flutter, η βιβλιοθήκη Riverpod μπορεί να χρησιμοποιηθεί για αυτόν τον σκοπό. Βασικός σκοπός της βιβλιοθήκης Riverpod είναι η αποδοτική διαχείριση της κατάστασης της εφαρμογής (state management), με τη χρήση Παρόχων (Providers), Καταναλωτών (Consumers) και Ειδοποιητών (Notifiers). Χρησιμοποιούμε αυτά τα χαρακτηριστικά για την υλοποίηση της αρχιτεκτονικής MVVM ως εξής: το κάθε View (η κάθε οθόνη της εφαρμογής) είναι ένας Consumer, ο οποίος παρακολουθεί αλλαγές των Providers (και, ως εκ τούτου, ενημερώνεται αυτόματα από το Riverpod). Η κάθε ιδιότητα (property) του κάθε ViewModel είναι ένας Provider (για την ακρίβεια, ένας NotifierProvider, δηλαδή ένας Provider που έχει τη δυνατότητα να ειδοποιεί τους καταναλωτές του). Αυτούς τους Providers παρακολουθούν τα Views για αλλαγές. Όταν, λοιπόν, το ViewModel επιθυμεί να τροποποιήσει μία ιδιότητά του, αλλάζει το περιεχόμενο ενός Provider, ενώ ο αντίστοιχος Notifier στέλνει σήμα τροποποίησης στο View, το οποίο ενημερώνεται αμέσως.

4.3 Μοντέλο δεδομένων

Όπως αναφέρθηκε στην Ενότητα 4.2, το πλεονέκτημα της αρχιτεκτονικής MVVM είναι η πλήρης ανεξαρτησία του front-end από το back-end. Παρ'όλα αυτά, η άψογη διαλειτουργικότητα προϋποθέτει τη συμβατότητα των δεδομένων που μεταφέρονται σε όλα τα τμήματα της εφαρμογής. Για τον σκοπό αυτό, το Model πρέπει να υλοποιεί ένα interface, το οποίο περιγράφει το μοντέλο δεδομένων, με βάση το οποία τα δεδομένα αποθηκεύονται στη βάση δεδομένων και διακινούνται σε όλη την εφαρμογή.

Για τη σχεδίαση του μοντέλου δεδομένων της εφαρμογής Horario, κατασκευάζουμε ένα **Διάγραμμα Οντοτήτων-Συσχετίσεων** (Entity-Relationship Diagram), το οποίο παρουσιάζεται στην Εικόνα 4.2. Το Διάγραμμα Οντοτήτων-Συσχετίσεων αποτελεί μία αναπαράσταση των τύπων δεδομένων που αποθηκεύονται σε μία βάση δεδομένων, καθώς και τις σχέσεις που αυτοί έχουν μεταξύ τους. Κάθε οντότητα (entity) παριστάνεται με ένα ορθογώνιο πλαίσιο, ενώ κάθε συσχέτιση (relationship) με έναν ρόμβο. Η κάθε οντότητα περιλαμβάνει χαρακτηριστικά (attributes), τα οποία παριστάνονται με ελλείψεις και τα οποία περιγράφουν τις ιδιότητες καθενός αντικειμένου της οντότητας.

Η κάθε οντότητα συνήθως ορίζεται μονοσήμαντα από ένα ή περισσότερα χαρακτηριστικά της, τα οποία είναι μοναδικά και ονομάζονται πρωτεύον κλειδί (primary key) της οντότητας. Τα χαρακτηριστικά που απαρτίζουν ένα πρωτεύον κλειδί σημειώνονται υπογραμμισμένα εντός της έλλειψης. Ένα χαρακτηριστικό μπορεί να λάβει περισσότερες από μία τιμές, οπότε ονομάζεται πλειότιμο χαρακτηριστικό (multivalued attribute) και παριστάνεται με δύο γραμμές στην περιφέρεια της έλλειψης. Ένα χαρακτηριστικό μπορεί να είναι προαιρετικό, ώστε κάποια αντικείμενα της

οντότητας να μην λάβουν τιμή για αυτό. Ένα προαιρετικό (optional) χαρακτηριστικό παριστάνεται με μία διακεκομένη γραμμή που ενώνει την έλλειψη με το ορθογώνιο πλαίσιο της οντότητας.

Μία συσχέτιση περιγράφει τη σχέση που υπάρχει μεταξύ δύο ή περισσότερων οντοτήτων. Η πληθικότητα (cardinality) της συσχέτισης ορίζει τον αριθμό των στιγμιοτύπων της οντότητας που συνδέονται με την άλλη οντότητα μέσω της συσχέτισης. Μία συσχέτιση μπορεί επίσης να περιλαμβάνει χαρακτηριστικά (attributes), τα οποία περιγράφουν ιδιότητες της συσχέτισης μεταξύ αντικειμένων της κάθε συμπεριλαμβανομένης οντότητας. Η πληθικότητα σε κάθε άκρο της συσχέτισης μπορεί να λάβει τις τιμές:

- 0 έως 1: απαιτείται το πολύ ένα αντικείμενο της οντότητας για τη συσχέτιση.
- 1 έως 1: απαιτείται ακριβώς ένα αντικείμενο της οντότητας για τη συσχέτιση.
- 0 έως πολλά: απαιτείται οσοσδήποτε αριθμός αντικειμένων της οντότητας για τη συσχέτιση.
- 1 έως πολλά: απαιτείται τουλάχιστον ένα αντικείμενο της οντότητας για τη συσχέτιση.

Μία οντότητα χαρακτηρίζεται **ασθενής οντότητα** (weak entity) και παριστάνεται με μία διπλή γραμμή στο περίγραμμα του ορθογωνίου πλαισίου της, όταν κάθε αντικείμενό της δεν ορίζεται μονοσήμαντα από ένα πρωτεύον κλειδί, αλλά, αντίθετα, ορίζεται μονοσήμαντα από ένα αντικείμενο μίας άλλης -ισχυρής- οντότητας, μέσω μίας συσχέτισης, η οποία ονομάζεται **ασθενής συσχέτιση** (weak relationship) και παριστάνεται με μία διπλή γραμμή στο περίγραμμα του ρόμβου της.

Τέλος, μία οντότητα μπορεί να εξειδικεύεται σε επιμέρους οντότητες μέσω μίας διαζευκτικής (disjoint) σχέσης. Η διάζευξη παριστάνεται με έναν κύκλο με το γράμμα “d”, στον οποίο η γενική οντότητα συνδέεται με μία γραμμή, ενώ η κάθε ειδική οντότητα με μία γραμμή, τεμνόμενη από μία καμπύλη.

Ο ενδιαφερόμενος γύρω από θέματα σχεδίασης βάσεων δεδομένων αναγνώστης μπορεί να ανατρέξει για περισσότερες λεπτομέρειες σε σχετικά συγγράμματα, όπως το (Elmasri και Navathe 2015) και το (Silberschatz, Korth, και Sudarshan 2010).

Θα εφαρμόσουμε τα παραπάνω για να περιγράψουμε το Διάγραμμα Οντοτήτων-Συσχετίσεων της Εικόνας 4.2. Μπορούμε να χωρίσουμε νοητά το Διάγραμμα στο άνω και κεντρικό τμήμα του, το οποίο περιγράφει έναν οργανισμό, τη διαχείρισή του και τους εργαζομένους του, και στο κάτω τμήμα του, το οποίο εστιάζει στον χρονοπρογραμματισμό της εργασίας για τον συγκεκριμένο οργανισμό.

Έτσι, έχουμε τις οντότητες *Organization*, *Location*, *Team* και *Employee* και τις μεταξύ τους συσχετίσεις, οι οποίες περιγράφουν την οργάνωση του οργανισμού. Κάθε οργανισμός αποτελείται από οποιονδήποτε αριθμό τοποθεσιών (καθεμία από τις οποίες ανήκει σε έναν μόνο οργανισμό) και κάθε τοποθεσία περιλαμβάνει οποιονδήποτε αριθμό από ομάδες (καθεμία από τις οποίες ανήκει σε μία μόνο τοποθεσία). Ο κάθε οργανισμός απασχολεί οποιονδήποτε αριθμό εργαζομένων (καθένας από τους οποίους ανήκει σε έναν μόνο οργανισμό). Επίσης, η κάθε ομάδα στελεχώνεται από οποιονδήποτε αριθμό εργαζομένων, καθένας από τους οποίους εργάζεται σε οποιονδήποτε αριθμό ομάδων. Ακόμη, η κάθε ομάδα διευθύνεται από τουλάχιστον έναν εργαζόμενο, ενώ ο κάθε εργαζόμενος μπορεί

4.4 Υλοποίηση της βάσης δεδομένων

να διευθύνει οποιονδήποτε αριθμό ομάδων. Ο κάθε εργαζόμενος είναι προσβάσιμος από έναν λογαριασμό στην εφαρμογή, ενώ ο κάθε λογαριασμός μπορεί να έχει πρόσβαση σε οποιονδήποτε αριθμό εργαζομένων.

Στη συνέχεια, έχουμε τις οντότητες *Shift Rule* και *Shift Type*. Η κάθε ομάδα του οργανισμού οργανώνεται σε οποιονδήποτε αριθμό τύπων βαρδιών (καθένας από τους οποίους ανήκει σε μία μόνο ομάδα). Η ανάθεση του κάθε εργαζόμενου σε έναν τύπο βαρδιών περιορίζεται από οποιονδήποτε αριθμό κανόνων βαρδιών (καθένας από τους οποίους αντιστοιχίζεται σε έναν μόνο εργαζόμενο, από τον οποίο ορίζεται και μονοσήμαντα - ασθενής οντότητα). Ο κάθε εργαζόμενος μπορεί να εκφράσει την προτίμησή του για οποιονδήποτε αριθμό τύπων βαρδιών και για οποιονδήποτε αριθμό συναδέλφων του, με τους αντίστοιχους συντελεστές βαρύτητας.

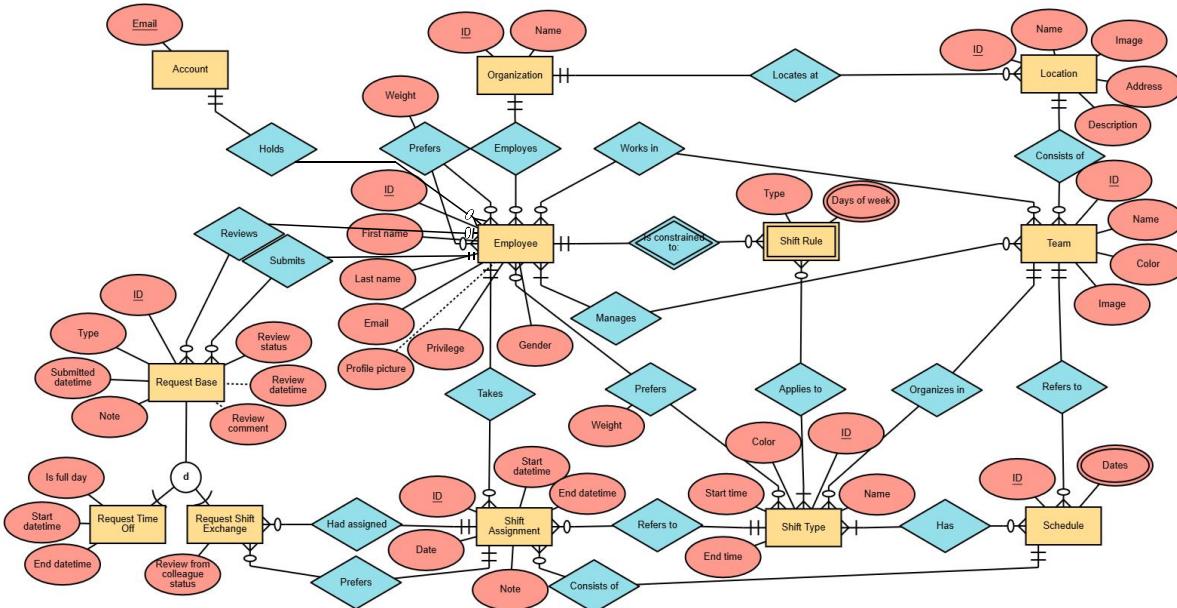
Έπειτα, οι οντότητες *Schedule* και *Shift Assignment* περιλαμβάνουν τα δεδομένα του χρονοπρογραμματισμού της εργασίας. Το κάθε πρόγραμμα αναφέρεται σε μία μόνο ομάδα του οργανισμού (καθεμία από τις οποίες περιλαμβάνει οποιονδήποτε αριθμό προγραμμάτων) και περιλαμβάνει τουλάχιστον έναν τύπο βαρδιών της συγκεκριμένης ομάδας. Το κάθε πρόγραμμα αποτελείται από οποιονδήποτε αριθμό αναθέσεων σε βάρδιες, ενώ κάθε ανάθεση περιλαμβάνεται σε ένα μόνο πρόγραμμα. Η κάθε ανάθεση αναλαμβάνεται από έναν μόνο εργαζόμενο (καθένας από τους οποίους αναλαμβάνει οποιονδήποτε αριθμό αναθέσεων), ενώ αναφέρεται σε έναν μόνο τύπο βαρδιών (καθένας από τους οποίους σχετίζεται με οποιονδήποτε αριθμό αναθέσεων).

Τέλος, η οντότητα *Request Base* περιλαμβάνει τα δεδομένα αιτημάτων των εργαζομένων. Ο κάθε εργαζόμενος υποβάλλει οποιονδήποτε αριθμό αιτημάτων (καθένα από τα οποία υποβάλλεται από μόνο έναν εργαζόμενο) και εγκρίνει οποιονδήποτε αριθμό αιτημάτων (καθένα από τα οποία εγκρίνεται από το πολύ έναν εργαζόμενο). Η οντότητα *Request Base* εξειδικεύεται στις οντότητες *Request Time Off* και *Request Shift Exchange*. Το κάθε αίτημα ανταλλαγής βαρδιών διαθέτει ανατεθειμένη μόνο μία ανάθεση βάρδιας (καθεμία από τις οποίες μπορεί να σχετίζεται με οποιονδήποτε αριθμό αιτημάτων ανταλλαγής βαρδιών) και καθορίζει ως προτιμώμενη μόνο μία, διαφορετική, ανάθεση βάρδιας (καθεμία από τις οποίες μπορεί επίσης να σχετίζεται με οποιονδήποτε αριθμό αιτημάτων ανταλλαγής βαρδιών).

4.4 Υλοποίηση της βάσης δεδομένων

Έχοντας σχεδιάσει τη βάση δεδομένων, καλούμαστε να επιλέξουμε ένα **σύστημα διαχείρισης βάσης δεδομένων (database management system - DBMS)**. Ένα σύστημα διαχείρισης βάσης δεδομένων είναι, ουσιαστικά, ένα σύστημα λογισμικού που επιτρέπει σε χρήστες και εφαρμογές να αλληλεπιδρούν με βάσεις δεδομένων. Οι λειτουργίες που επιτελεί ένα σύστημα διαχείρισης βάσης δεδομένων περιλαμβάνουν (MongoDB, χ.χ.):

- **Αποθήκευση δεδομένων:** δυνατότητα αποθήκευσης δεδομένων σε μία κεντρική τοποθεσία



Εικόνα 4.2. Το Διάγραμμα Οντοτήτων-Συσχετίσεων για την εφαρμογή Horario

- **Ανάκτηση δεδομένων:** δυνατότητα αποδοτικής ανάκτησης δεδομένων μέσω των κατάλληλων ερωτημάτων
- **Διαχείριση δεδομένων:** προσθήκη, διαγραφή και τροποποίηση δεδομένων σε μία βάση δεδομένων
- **Οργάνωση δεδομένων:** δόμηση των δεδομένων με έναν συμπαγή και λογικό τρόπο

Αρχικά, τα δεδομένα διαχειρίζονται σε αρχεία εντός αρχείων συστημάτων (file systems), πρακτική που γρήγορα κατέστη μη αποδοτική, λόγω της περιορισμένης λειτουργικότητας και αποδοτικότητας στη διαχείριση μεγάλων όγκων δεδομένων. Έτσι, εμφανίστηκαν οι πρώτες σχεσιακές βάσεις δεδομένων (relational databases), συστήματα τα οποία βασίζονται στη διάσημη πλέον γλώσσα προγραμματισμού SQL (Structured Query Language). Σήμερα, τα συστήματα αυτά έχουν εξελιχθεί πολύ περισσότερο, με τεχνολογίες όπως η NoSQL και οι βάσεις δεδομένων γράφων (MongoDB, χ.χ.).

Σε αυτό το σημείο, επιλέξαμε για την υποστήριξη της εφαρμογής Horario την ολοκληρωμένη πλατφόρμα νέφους Firebase. To Firebase είναι μία ολοκληρωμένο σύστημα που βασίζεται στην Πλατφόρμα Νέφους της Google (Google Cloud Platform) και περιλαμβάνει δεκάδες προϊόντα, για την υποστήριξη εφαρμογών που βασίζονται στο νέφος. To Firestore είναι ένα NoSQL σύστημα διαχείρισης βάσης δεδομένων που περιλαμβάνεται στα προϊόντα του Firebase. Στόχος του είναι η εύκολη και αποδοτική διαχείριση των δεδομένων εφαρμογών για κινητές συσκευές και ιστού, ώστε να είναι εύκολα επεκτάσιμες παγκοσμίως.

Οι εφαρμογές επικοινωνούν με το Firestore, καθώς και με όλα τα προϊόντα του Firebase, χρησιμοποιώντας το Firebase SDK, το οποίο έχει αναπτυχθεί για πολλές γλώσσες προγραμματισμού,

4.5 Τοπική αποθήκευση δεδομένων και μεγάλα δυαδικά αρχεία

βασική από τις οποίες αποτελεί η Dart για το Flutter. Ουσιαστικά, η επικοινωνία γίνεται μέσω REST APIs, πραγματοποιώντας αιτήματα και λαμβάνοντας αποκρίσεις HTTP, τα οποία όμως πραγματοποιεί το SDK, για διευκόλυνση των προγραμματιστών.

Στο Firestore, τα δεδομένα αποθηκεύονται σε μορφή **συλλογών (collections)**, οι οποίες αντιστοιχούν στις οντότητες του Διαγράμματος Οντοτήτων-Συσχετίσεων, και οι οποίες περιλαμβάνουν **έγγραφα (documents)**, δηλαδή τα αντικείμενα των οντοτήτων. Η NoSQL φύση του Firestore δεν επιβάλλει ένα αυστηρό σχήμα για τη μορφή των εγγράφων, ωστόσο δομούμε τα δεδομένα της εφαρμογής μας με τον τρόπο που παριστάνεται στις Εικόνες 4.3 και 4.4.

4.5 Τοπική αποθήκευση δεδομένων και μεγάλα δυαδικά αρχεία

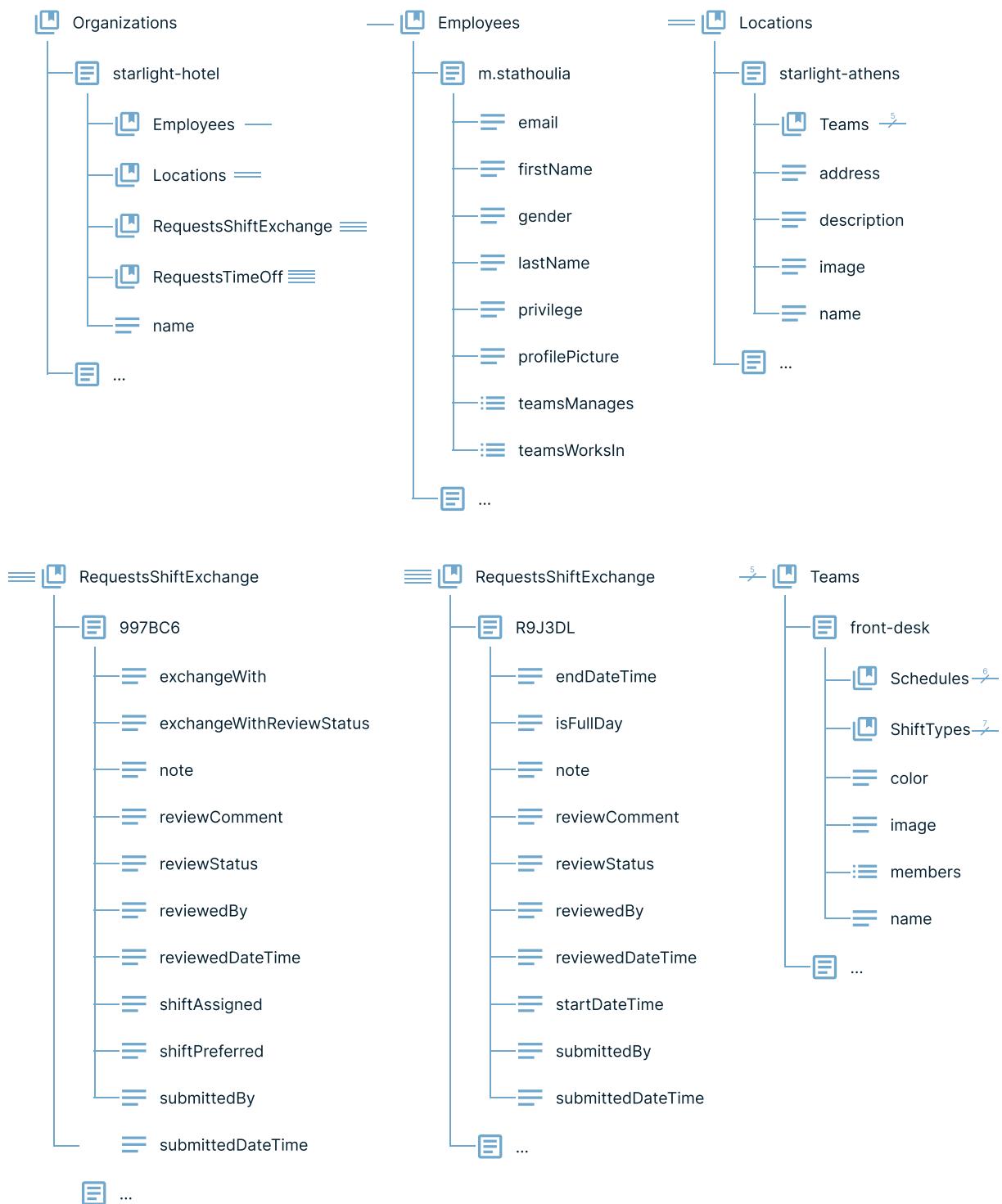
Η βάση δεδομένων που προσφέρει το Firestore είναι κατάλληλη για τα δεδομένα της εφαρμογής, τα οποία πρέπει να είναι προσβάσιμα από όλους τους πελάτες (clients), δηλαδή όλες τις συσκευές και τους φυλλομετρητές. Ωστόσο, υπάρχουν στοιχεία προς αποθήκευση, τα οποία δεν μπορούν ή δεν ενδείκνυται να αποθηκευτούν με αυτή τη μορφή στο Firestore.

Αρχικά, ρυθμίσεις της εφαρμογής, όπως είναι η γλώσσα και η εμφάνιση δεν εξαρτώνται από τον λογαριασμό του χρήστη, ούτε από τον τρέχοντα οργανισμό. Αντίθετα, ανήκουν σε κάθε συσκευή και πρέπει να αποθηκεύονται τοπικά σε αυτή. Για τον σκοπό αυτό, χρησιμοποιούμε τη βιβλιοθήκη **Shared Preferences** του Flutter, η οποία επιτρέπει την τοπική αποθήκευση δεδομένων στη συσκευή, όπως καθορίζεται από την υποκείμενη πλατφόρμα.

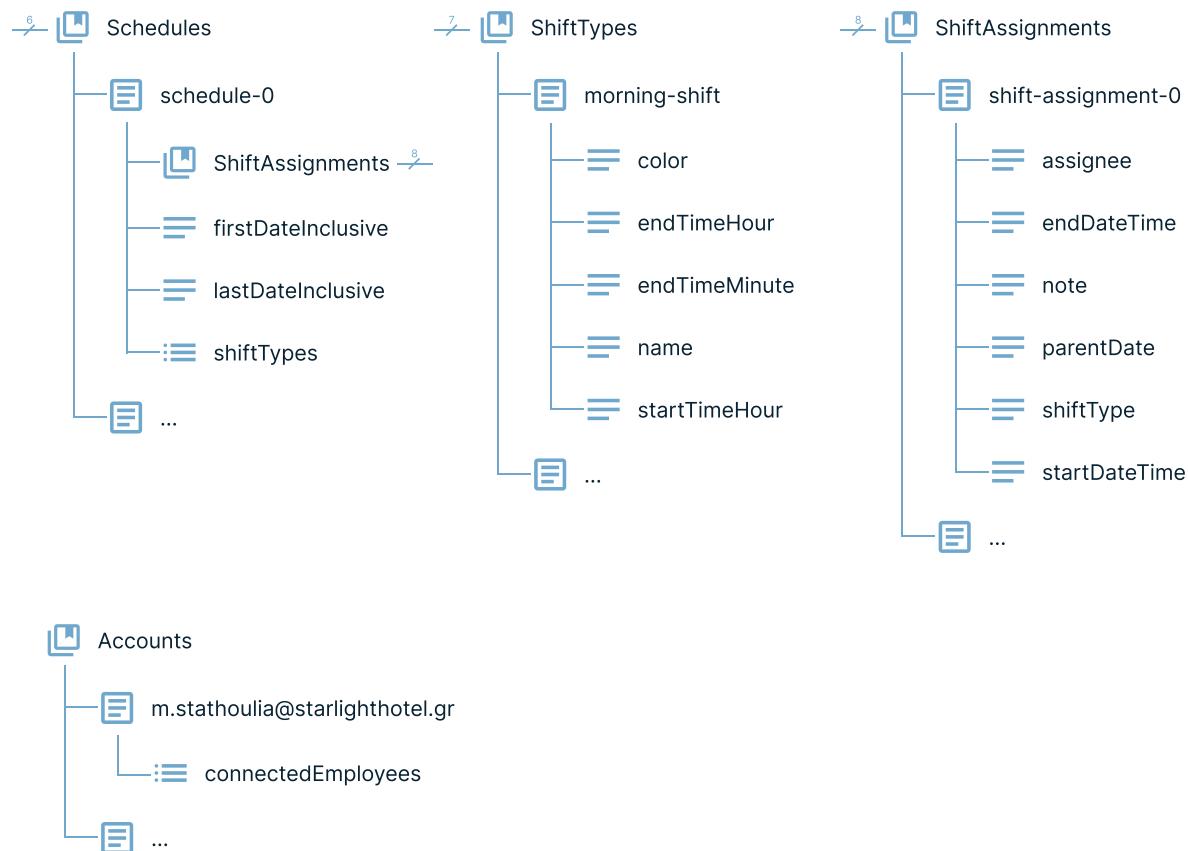
Επιπλέον, δεδομένα που δεν ενδείκνυται να αποθηκευτούν στο Firestore αποτελούν και τα μεγάλα δυαδικά αρχεία (binary large objects - BLOB), όπως οι εικόνες προφίλ των εργαζομένων, οι εικόνες ομάδων και τοποθεσιών. Για την αποθήκευση τέτοιων αρχείων, χρησιμοποιούμε το **Storage**, ένα προϊόν του Firebase για αυτόν τον σκοπό. Η επικοινωνία με το Storage πραγματοποιείται επίσης μέσω του Firebase SDK. Επίσης, στο Firestore αποθηκεύουμε ένα αναγνωριστικό των αρχείων στα αντίστοιχα έγγραφα (documents), ώστε να μπορούμε να κάνουμε τις σωστές ανακτήσεις δεδομένων (π.χ. στο πεδίο *profilePicture* του εγγράφου *m.stathoulia*, αποθηκεύουμε το αναγνωριστικό από το Storage της εικόνας προφίλ του χρήστη *Μαργαρίτα Σταθούλια*).

4.6 Αυθεντικοποίηση χρηστών

Οι περισσότερες εφαρμογές χρειάζεται να γνωρίζουν την ταυτότητα του τρέχοντος χρήστη, ώστε να επιτρέπουν την ασφαλή αποθήκευση δεδομένων στο νέφος και να προσφέρουν την ίδια εμπειρία σε όλες τις συσκευές για τον ίδιο χρήστη. Το Firebase προσφέρει το προϊόν **Authentication**, που αποτελεί ένα ολοκληρωμένο σύστημα αυθεντικοποίησης χρηστών με διάφορους τρόπους:



Εικόνα 4.3. Η δόμηση των δεδομένων της εφαρμογής Horario στο Firestore



Εικόνα 4.4. Η δόμηση των δεδομένων της εφαρμογής Horario στο Firestore (συνέχεια)

- Email και password
- Αριθμό τηλεφώνου
- Συνεργαζόμενους παρόχους ταυτότητας (federated identity providers), όπως το Google Sign-in και το Facebook Login.

Το Authentication ενσωματώνεται στην εφαρμογή μέσω του Firebase SDK και επιτρέπει την αυθεντικοποίηση χωρίς την ανάγκη κώδικα στην πλευρά του εξυπηρετητή.

4.7 Μηχανή αυτόματης ανάθεσης βαρδιών

4.7.1 Υλοποίηση αλγορίθμου

Κεντρικό σημείο της εφαρμογής **Horario** αποτελεί η μηχανή αυτόματης ανάθεσης βαρδιών, η οποία εκτελείται όταν ο χρήστης δημιουργεί ένα νέο ή επεξεργάζεται ένα υπάρχον πρόγραμμα εργασίας και επιλέξει τουλάχιστον μία αυτόματη ανάθεση βάρδιας (βλ. Ενότητα 3.4.6). Ο σχηματισμός του αλγορίθμου από μαθηματικής πλευράς έχει αναλυθεί εκτενώς στην Ενότητα 2.3.4. Στην Ενότητα αυτή, υλοποιούμε τον αλγόριθμο προγραμματιστικά, και τον ενσωματώνουμε στο front-end της εφαρμογής.

Από αρχιτεκτονικής άποψης, επιλέγουμε η μηχανή να είναι κεντρικοποιημένη, δηλαδή να εκτελείται απομακρυσμένα σε έναν εξυπηρετητή, και όχι στην εκάστοτε συσκευή του χρήστη, παράλληλα με το front-end. Η επιλογή αυτή γίνεται κυρίως για δύο λόγους.

Πρώτον, η πιο δημοφιλής γλώσσα που επιλέγεται για την επίλυση προβλημάτων γραμμικού προγραμματισμού είναι συνήθως η Python, λόγω της πληθώρας σχετικών βιβλιοθηκών. Έτσι, υπάρχει αναντιστοιχία με τη γλώσσα προγραμματισμού του front-end, που είναι η Dart, και άρα εμφανίζεται η ανάγκη υλοποίησης ενός API, που θα πραγματοποιεί τη διασύνδεση των δύο τμημάτων.

Δεύτερον, το περιβάλλον εκτέλεσης του front-end της εφαρμογής είναι πλήρως μη ελεγχόμενο, τόσο από πλευράς πλατφόρμας (και άρα υποστήριξης της Python), όσο και από πλευράς υπολογιστικής ισχύος. Έτσι, εάν επιλέγαμε να υλοποιήσουμε τη μηχανή αποκεντρωμένα, παράλληλα με το front-end σε κάθε συσκευή, θα έπρεπε πάντα να εξασφαλίζουμε ότι η Python είναι εγκατεστημένη στο σύστημα, μαζί με όλες τις απαραίτητες βιβλιοθήκες. Όσον αφορά την υπολογιστική ισχύ, η φύση του αλγορίθμου και η πολυπλοκότητα του προβλήματος δεν έχουν ιδιαίτερες απαιτήσεις. Παρ' όλα αυτά, θα ήταν καλή πρακτική να απομονώσουμε την εκτέλεση της μηχανής σε έναν εξυπηρετητή που λειτουργεί καθαρά επί τούτου, κυρίως λόγω της αβεβαιότητας περί της υφιστάμενης συσκευής και πλατφόρμας.

Επιλέγουμε, λοιπόν, να υλοποιήσουμε τη μηχανή σε Python. Η **PuLP** είναι μία βιβλιοθήκη της Python, που υλοποιεί εσωτερικά δημοφιλείς αλγορίθμους γραμμικού προγραμματισμού, με σκοπό την επίλυση τέτοιων προβλημάτων. Το μοντέλο της μηχανής είναι πολύ απλούστερο από αυτό του front-end και

4.7 Μηχανή αυτόματης ανάθεσης βαρδιών

περιλαμβάνει μονάχα τις απαραίτητες για τον ορισμό του προβλήματος γραμμικού προγραμματισμού οντότητες του μοντέλου δεδομένων (βλ. Ενότητα 4.3), όπως τις *Employee*, *ShiftType*, *ShiftAssignment* και *Schedule*, τα απαιτούμενα χαρακτηριστικά και οι συσχετίσεις τους.

Για την επίλυση του προβλήματος, εισάγουμε τους απαραίτητους περιορισμούς, όπως περιγράφονται στην Ενότητα 2.3.4. Υπενθυμίζουμε ότι οι περιορισμοί αυτοί εκφράζονται με τη μορφή αθροισμάτων των τιμών των μεταβλητών απόφασης, προστίθενται στο μοντέλο γραμμικού προγραμματισμού, επηρεάζοντας την αντικειμενική συνάρτηση και είναι οι ακόλουθοι:

- Το πολύ μία ανάθεση ανά εργαζόμενο, ανά ημέρα (ή καμία, αν έχει ζητηθεί ρεπό ή άδεια).
- Μέγιστος αριθμός αναθέσεων εντός του χρονικού ορίζοντα (του εύρους του αντικειμένου *Schedule*).
- Μέγιστος αριθμός αναθέσεων σε συνεχόμενες ημέρες
- Απαγόρευση ανάθεσης σε νυχτερινή βάρδια και στην αμέσως επόμενη πρωινή

Οι περιορισμοί αυτοί είναι ενσωματωμένοι στον κώδικα στην πλευρά του εξυπηρετητή και ισχύουν για όλα τα προγράμματα εργασίας όλων των οργανισμών, χωρίς δυνατότητα επεξεργασίας.

Ο αλγόριθμος δέχεται ως είσοδο τις ημέρες του προγράμματος εργασίας, τους τύπους βαρδιών, τον αριθμό αναθέσεων που πρέπει να πραγματοποιηθούν, τους διαθέσιμους εργαζομένους και τις προτιμήσεις τους. Ο αλγόριθμος παράγει ως έξοδο τις τιμές των μεταβλητών απόφασης που δημιουργήθηκαν, και οι οποίες καθορίζουν τις αναθέσεις που πραγματοποιήθηκαν, με τη μορφή *True* / *False* (True εάν ο συγκεκριμένος εργαζόμενος ανατέθηκε στη συγκεκριμένη βάρδια, αλλιώς False).

4.7.2 Υλοποίηση εξυπηρετητή

Για την ενσωμάτωση της μηχανής στο front-end, κατασκευάζουμε ένα REST API με [Flask](#), μία από τις πιο γνωστές βιβλιοθήκες της Python για αυτόν τον σκοπό. Το front-end πραγματοποιεί αιτήματα HTTP σε ένα endpoint αυτού του API. Στη συνέχεια, το API, αφού διαχειριστεί κατάλληλα τα δεδομένα που έλαβε από το αίτημα HTTP, καλεί τον αλγόριθμο αυτόματης ανάθεσης, με τις κατάλληλες εισόδους. Έπειτα, με την ολοκλήρωση της εκτέλεσης του αλγορίθμου, το API λαμβάνει την έξοδο και, αφού τη διαχειριστεί κατάλληλα, στέλνει στο front-end μία απόκριση HTTP με τα αποτελέσματα της αυτόματης ανάθεσης. Καθ' όλη αυτή τη διάρκεια, το front-end παραμένει στάσιμο, υποδεικνύοντας στον χρήστη ότι εκτελείται λειτουργία στο παρασκήνιο, με έναν δείκτη προόδου (progress indicator).

Ο εξυπηρετητής αυτός, καθώς πρέπει να είναι πάντα προσβάσιμος από όλες τις συσκευές, πρέπει να φιλοξενείται σε μία υπηρεσία νέφους (cloud hosting). Για τον σκοπό αυτό, επιλέγουμε την υπηρεσία [Fly](#). Το Fly φιλοξενεί full-stack εφαρμογές ιστού και παρέχει υποστήριξη για διασύνδεση με βάσεις δεδομένων, εικονικές μηχανές, μονάδες επεξεργασίας γραφικών (GPU) και άλλα. Επιλέξαμε τη συγκεκριμένη υπηρεσία λόγω προϋπάρχουσας εμπειρίας από εργασίες εξαμήνων.

4.8 Σύνοψη

Στο Κεφάλαιο αυτό, συζητήθηκε ο τρόπος με τον οποίο τα σχέδια, όπως προέκυψαν στο τέλος του Κεφαλαίου 3, μετατράπηκαν σε μία λειτουργική εφαρμογή. Εστιάσαμε τόσο στο front-end τμήμα, με το οποίο έρχεται σε επαφή ο χρήστης της εφαρμογής και το οποίο σχεδιάσαμε με βάση τις αρχές του ανθρωποκεντρικού σχεδιασμού, αλλά και στο back-end τμήμα, το οποίο υποστηρίζει και ολοκληρώνει το άλλο, χειριζόμενο τα δεδομένα της εφαρμογής.

Στο επόμενο Κεφάλαιο, θα συζητηθούν οι μέθοδοι που εφαρμόστηκαν για την αξιολόγηση της εφαρμογής, μία διαδικασία απαραίτητη και πολύ χρήσιμη στα πλαίσια του ανθρωποκεντρικού σχεδιασμού.

5 Αξιολόγηση της εφαρμογής

5.1 Περί αξιολόγησης ευχρηστίας στον ανθρωποκεντρικό σχεδιασμό

Η αξιολόγηση της ευχρηστίας και, γενικότερα, της εμπειρίας χρήσης ενός λογισμικού είναι ένα πολύ σημαντικό τμήμα της ανάπτυξης λογισμικού. Με βάση τις αρχές του ανθρωποκεντρικού σχεδιασμού, όπως αυτός αναλύεται στην Ενότητα 2.2, αξιολόγηση πρέπει να εκτελείται σε τακτική βάση, ώστε να διορθώνονται τυχόντα σχεδιαστικά σφάλματα και να βελτιώνονται οι επιλογές, οδηγώντας σε μία καλύτερη εμπειρία χρήσης του προϊόντος (Αβούρης κ.ά. 2018).

Σύμφωνα με τα (Nielsen 1994) και (Αβούρης κ.ά. 2018), οι μέθοδοι αξιολόγησης μπορούν να οργανωθούν σε τρεις μεγάλες κατηγορίες:

- **Αξιολογήσεις από ειδικούς (expert-based evaluation methods):** στην κατηγορία αυτή ανήκουν οι μέθοδοι που εκτελούνται από αξιολογητές, δηλαδή άτομα που γνωρίζουν εκτενώς τους κανόνες και τις μεθοδολογίες του ανθρωποκεντρικού σχεδιασμού.
- **Αξιολογήσεις από χρήστες (user-based evaluation methods):** στην κατηγορία αυτή ανήκουν οι μέθοδοι που εκτελούνται από τελικούς χρήστες, δηλαδή άτομα στα οποία απευθύνεται το προϊόν.
- **Μέθοδοι ανάλυσης δεδομένων από συστήματα που έχουν ήδη υλοποιηθεί.**

Οι expert-based μέθοδοι αξιολόγησης λειτουργούν συνήθως ως **διαμορφωτικές αξιολογήσεις (formative evaluations)**, δηλαδή αυτές που συμβάλλουν στη διαμόρφωση του τελικού προϊόντος, ενώ οι user-based μέθοδοι λειτουργούν τόσο ως διαμορφωτικές, όσο και ως **συμπερασματικές αξιολογήσεις (summative evaluations)**, δηλαδή αποτιμούν την επιτυχία του συστήματος σε όρους ευχρηστίας και εμπειρίας χρήσης.

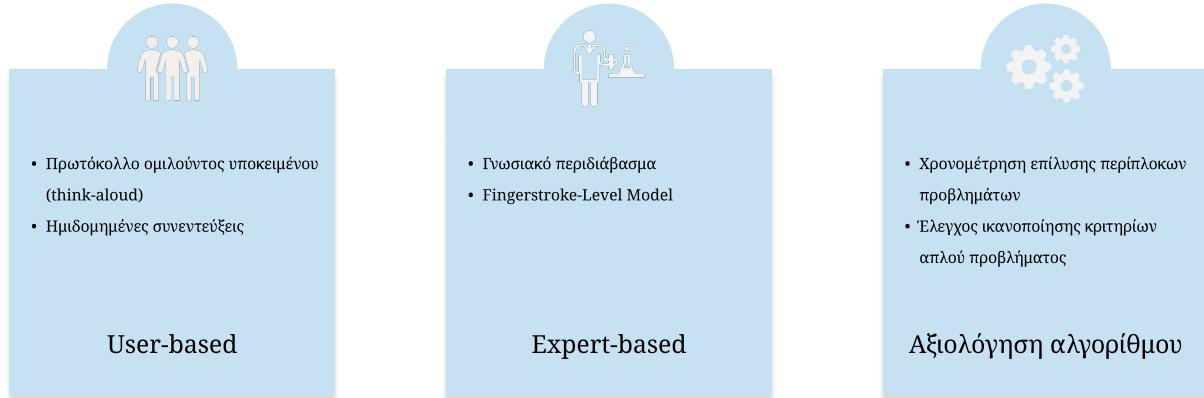
5.2 Μέθοδοι που χρησιμοποιήθηκαν

Στα πλαίσια αξιολόγησης της εφαρμογής Horario εφαρμόστηκαν τόσο expert-based, όσο και user-based μέθοδοι αξιολόγησης. Ως expert-based μεθόδους εφαρμόσαμε τη μέθοδο του γνωσιακού περιδιαβάσματος (cognitive walkthrough, Ενότητα 5.3) και τη μέθοδο Fingerstroke-Level Model (FLM, Ενότητα 5.4). Ως user-based μεθόδους, ζητήσαμε από πραγματικούς χρήστες της εφαρμογής μας να

5.3 Γνωσιακό περιδιάβασμα (Cognitive walkthrough)

εκτελέσουν ένα σενάριο χρήσης, με βάση το πρωτόκολλο του ομιλούντος υποκειμένου (think-aloud), διαδικασία που ακολουθήθηκε από ημιδομημένες συνεντεύξεις, που αναλύονται στην Ενότητα 5.5.

Τέλος, αξιολογήσαμε την αποδοτικότητα του αλγορίθμου αυτόματης ανάθεσης βαρδιών, εκτελώντας μία σειρά πειραμάτων που εξηγούνται στην Ενότητα 5.6.



Εικόνα 5.1. Μέθοδοι αξιολόγησης της εφαρμογής

Εκτελέσαμε δύο κύκλους αξιολόγησης της εφαρμογής. Ο πρώτος κύκλος έλαβε χώρα μετά τη σχεδίαση των πρωτοτύπων και είχε τη μορφή διαμορφωτικής αξιολόγησης, ώστε να ερευνηθεί η ευχρηστία της διεπαφής που είχε σχεδιαστεί. Ο δεύτερος κύκλος έλαβε χώρα όταν πλέον υλοποιήθηκε η εφαρμογή, σε μία προσπάθεια συμπερασματικής αξιολόγησης.

5.3 Γνωσιακό περιδιάβασμα (Cognitive walkthrough)

Το γνωσιακό περιδιάβασμα (cognitive walkthrough) είναι μία αυστηρή τεχνική αξιολόγησης, που αναπτύχθηκε αρχικά από τους Lewis κ.ά. το 1990 (Lewis κ.ά. 1990). Η μέθοδος αυτή ελέγχει τη λογική των βημάτων που θα πρέπει να ακολουθήσει ο χρήστης της εφαρμογής, ώστε να επιτύχει έναν στόχο.

Στα πλαίσια αξιολόγησης της εφαρμογής Horario, εφαρμόσαμε το απλοποιημένο γνωσιακό περιδιάβασμα (cognitive jogthrough), που προτάθηκε το 2000 από τον Spencer (Spencer 2000). Σύμφωνα με αυτό, υλοποιήσαμε ένα σενάριο αλληλεπίδρασης με το σύστημα. Σε κάθε διακριτό βήμα της αλληλεπίδρασης, θέτουμε τα δύο ακόλουθα ερωτήματα και καταγράφουμε τις απαντήσεις μας:

- Γνωρίζω τι πρέπει να κάνω σε αυτό το βήμα;
- Αν κάνω το σωστό, θα γνωρίζω ότι έκανα το σωστό και ότι σημειώνω πρόοδο προς τον στόχο μου;

Για την εφαρμογή της μεθόδου του γνωσιακού περιδιαβάσματος, χρησιμοποιήσαμε τους στόχους που τέθηκαν κατά την ιεραρχική ανάλυση εργασιών και τις χαμηλού επιπέδου εργασίες που θα πρέπει να ακολουθηθούν ώστε αυτοί να επιτευχθούν (βλ. Ενότητα 3.1.6). Στον Πίνακα 5.1 φαίνονται

τα αποτελέσματα της διαδικασίας για τη δημιουργία προγράμματος εργασίας, ενώ στον Πίνακα 5.2 τα αντίστοιχα αποτελέσματα για τη δημιουργία αιτήματος από εργαζόμενο.

Πίνακας 5.1: Αποτελέσματα γνωσιακού περιδιαβάσματος για τη δημιουργία προγράμματος εργασίας

<i>α/α</i>	<i>Γνωρίζω τι πρέπει να κάνω;</i>	<i>Εφόσον έκανα το σωστό, καταλαβαίνω ότι έκανα το σωστό και σημειώνω πρόοδο;</i>
Βήμα 0: Έναρξη διαδικασίας	Ναι , πρέπει να μεταβώ στην καρτέλα <i>Schedule</i> και να πατήσω το κουμπί <i>New schedule</i> .	Ναι , βλέπω τον οδηγό δημιουργίας προγράμματος να εμφανίζεται.
Βήμα 1: Επιλογή ομάδας	Ναι , βλέπω ένα σύντομο κείμενο με οδηγία και τη λίστα με τις ομάδες προς επιλογή.	Ναι , βλέπω το επόμενο βήμα στον οδηγό και τη γραμμή προόδου να γεμίζει σταδιακά.
Βήμα 2: Επιλογή ημερομηνιών	Ναι , βλέπω ένα σύντομο κείμενο με οδηγία και τα πεδία ημερομηνιών έναρξης και λήξης.	Ναι , οι επιλεγμένες ημερομηνίες φαίνονται στην οθόνη και το κουμπί <i>Συνέχεια</i> είναι ορατό και έντονο, υποδεικνύοντας προτροπή για ενέργεια (call-to-action, CTA). Όταν πατηθεί, βλέπω το επόμενο βήμα στον οδηγό και τη γραμμή προόδου να γεμίζει σταδιακά.
Βήμα 3.0: Προσθήκη βαρδιών	Ναι , βλέπω την Ομαδική προβολή του προγράμματος, με εικονίδια προσθήκης σε κάθε κελί. Καταλαβαίνω ότι πρέπει να τα πατήσω.	Ναι , βλέπω το Φύλλο βάρδιας με ορατές επιλογές επεξεργασίας.
Βήμα 3.1: Επιλογή ημέρας	Ναι , καταλαβαίνω ότι την επέλεξα ήδη από την Ομαδική προβολή του προγράμματος και δεν το ξεχνώ γιατί είναι ορατή στο Φύλλο βάρδιας.	Ναι , είναι ορατή στο Φύλλο βάρδιας.

5.3 Γνωσιακό περιδιάβασμα (Cognitive walkthrough)

α/α	<i>Γνωρίζω τι πρέπει να κάνω;</i>	<i>Εφόσον έκανα το σωστό, καταλαβαίνω ότι έκανα το σωστό και σημειώνω πρόοδο;</i>
Βήμα 3.2: Επιλογή βάρδιας	Ναι, καταλαβαίνω ότι την επέλεξα ήδη από την Ομαδική προβολή του προγράμματος και δεν το ξεχνώ γιατί είναι ορατή στο Φύλλο βάρδιας.	Ναι, είναι ορατή στο Φύλλο βάρδιας.
Βήμα 3.3: Επιλογή εργαζόμενου ή επιλογή αυτόματης ανάθεσης	Ναι, βλέπω προεπιλεγμένο το κουμπί αυτόματης ανάθεσης και, αν το αποεπιλέξω, βλέπω το μενού επιλογής εργαζομένου.	Ναι, η επιλογή μου απεικονίζεται στο Φύλλο βάρδιας (κατάσταση κουμπιού ή περιεχόμενο μενού επιλογής εργαζομένου) και στην Ομαδική προβολή του προγράμματος εργασίας.
Βήμα 3.4: Επεξεργασία ώρας έναρξης	Ναι, βλέπω την επιλογή, την επιλεγμένη ώρα έναρξης και τα κουμπιά <i>Nωρίτερα, Αργότερα</i>	Ναι, η επιλεγμένη ώρα έναρξης ενημερώνεται στο Φύλλο βάρδιας.
Βήμα 3.5: Επεξεργασία ώρας λήξης	Ναι, βλέπω την επιλογή, την επιλεγμένη ώρα λήξης και τα κουμπιά <i>Nωρίτερα, Αργότερα</i>	Ναι, η επιλεγμένη ώρα λήξης ενημερώνεται στο Φύλλο βάρδιας.
Βήμα 3.6: Προσθήκη σημείωσης	Ναι, βλέπω στο πεδίο σημείωσης στο Φύλλο βάρδιας και την προτροπή <i>Leave a note</i> ...	Ναι, η σημείωση που άφησα φαίνεται στην οθόνη.
Βήμα 4: Αυτόματη ανάθεση βαρδιών	Ναι, σε περίπτωση που έχω επιλέξει τουλάχιστον μία αυτόματη ανάθεση, το κουμπί CTA αναγράφει <i>Auto-assign.</i>	Ναι, εάν πατήσω το κουμπί, εμφανίζεται μήνυμα επιβεβαίωσης, ένδειξη προόδου της εργασίας στο παρασκήνιο και, τελικά, τα εικονίδια αυτόματων αναθέσεων στην Ομαδική προβολή του προγράμματος εργασίας αντικαθίστανται από τις εικόνες του προσωπικού.

α/α	<p><i>Γνωρίζω τι πρέπει να κάνω;</i></p>	<p><i>Εφόσον έκανα το σωστό, καταλαβαίνω ότι έκανα το σωστό και σημειώνω πρόοδο;</i></p>
Βήμα 5: Δημοσίευση προγράμματος	<p>Ναι, το κουμπί CTA αναγράφει <i>Publish.</i></p>	<p>Ναι, εάν πατήσω το κουμπί, εμφανίζεται μήνυμα επιβεβαίωσης, ένδειξη προόδου της εργασίας στο παρασκήνιο και, τελικά, μήνυμα επιτυχίας.</p>

Πίνακας 5.2: Αποτελέσματα γνωσιακού περιδιαβάσματος για τη δημιουργία αιτήματος από εργαζόμενο

α/α	<p><i>Γνωρίζω τι πρέπει να κάνω;</i></p>	<p><i>Εφόσον έκανα το σωστό, καταλαβαίνω ότι έκανα το σωστό και σημειώνω πρόοδο;</i></p>
Βήμα 0: Έναρξη διαδικασίας	<p>Ναι, πρέπει να μεταβώ στην καρτέλα <i>Requests</i> και να πατήσω το κουμπί <i>New request.</i></p>	<p>Ναι, βλέπω τον οδηγό νέου αιτήματος να εμφανίζεται.</p>
Βήμα 1: Επιλογή τύπου	<p>Ναι, βλέπω ένα σύντομο κείμενο με οδηγία και τη λίστα με τους τύπους προς επιλογή.</p>	<p>Ναι, βλέπω το επόμενο βήμα στον οδηγό και τη γραμμή προόδου να γεμίζει σταδιακά.</p>
Βήμα 2: (Αίτημα αδείας) Καθορισμός λεπτομερειών	<p>Ναι, βλέπω ένα σύντομο κείμενο με οδηγία και τη φόρμα καθορισμού των λεπτομερειών. Ζητείται ο τύπος άδειας (μονοήμερη, πολυήμερη), οι ημερομηνίες έναρξης και λήξης και οι ώρες έναρξης και λήξης, σε περίπτωση που δε ζητείται ολοήμερη.</p>	<p>Ναι, οι επιλογές μου φαίνονται στην οθόνη και το κουμπί <i>Συνέχεια</i> είναι ορατό και έντονο, υποδεικνύοντας προτροπή για ενέργεια (call-to-action, CTA). Όταν πατηθεί, βλέπω το επόμενο βήμα στον οδηγό και τη γραμμή προόδου να γεμίζει σταδιακά.</p>

5.4 Τα μοντέλα KLM - FLM

α/α	<i>Γνωρίζω τι πρέπει να κάνω;</i>	<i>Εφόσον έκανα το σωστό, καταλαβαίνω ότι έκανα το σωστό και σημειώνω πρόοδο;</i>
Βήμα 2.1: (Αίτημα ανταλλαγής βαρδιών) Επιλογή βάρδιας προς ανταλλαγή	Ναι , βλέπω ένα σύντομο κείμενο με οδηγία και την Ατομική προβολή προγράμματος εργασίας, ώστε να επιλέξω τη βάρδια που θέλω να ανταλλάξω.	Ναι , βλέπω το επόμενο βήμα στον οδηγό και τη γραμμή προόδου να γεμίζει σταδιακά.
Βήμα 2.2: (Αίτημα ανταλλαγής βαρδιών) Επιλογή επιθυμητής βάρδιας	Ναι , βλέπω ένα σύντομο κείμενο με οδηγία και την Ομαδική προβολή προγράμματος εργασίας, ώστε να επιλέξω τη βάρδια που επιθυμώ να αναλάβω.	Ναι , βλέπω το επόμενο βήμα στον οδηγό και τη γραμμή προόδου να γεμίζει σταδιακά.
Βήμα 3: Προσθήκη σχολίου και έλεγχος	Ναι , βλέπω ένα σύντομο κείμενο με οδηγία και την επισκόπηση του αιτήματος. Βλέπω το πεδίο σχολίου και την προτροπή <i>Leave a comment</i>	Ναι , η σημείωση που άφησα φαίνεται στην οθόνη.
Βήμα 4: Υποβολή αιτήματος	Ναι , το κουμπί CTA αναγράφει <i>Submit</i> .	Ναι , εάν πατήσω το κουμπί, εμφανίζεται μήνυμα επιβεβαίωσης, ένδειξη προόδου της εργασίας στο παρασκήνιο και, τελικά, μήνυμα επιτυχίας.

Παρατηρούμε πως έχουμε θετικά αποτελέσματα σε όλες τις ερωτήσεις.

5.4 Τα μοντέλα KLM - FLM

Το μοντέλο πληκτρολογήσεων (Keystroke-Level Model) δημοσιεύτηκε το 1983 από τους Card, Moran και Newell (Card, Newell, και Moran 1983), (Αβούρης κ.ά. 2018). Στόχος αυτής της expert-based μεθόδου αξιολόγησης είναι η πρόβλεψη του χρόνου ακολουθίας των ενεργειών του χρήστη. Αποτελεί μία απλοποιημένη έκδοση του μοντέλου GOMS, η οποία κάνει την παραδοχή ότι ο χρήστης είναι

έμπειρος και αλάνθαστος, αλλά μονόχειρας.

Οι στοιχειώδεις ενέργειες του μοντέλου KLM, οι συμβολισμοί τους και οι εμπειρικοί μέσοι χρόνοι τους φαίνονται στον πίνακα 5.3.

Πίνακας 5.3: Κατηγορίες ενέργειών του μοντέλου KLM

Ενέργεια	Συμβολισμός	Εμπειρικός μέσος χρόνος
Πληκτρολόγηση (keystroking)	T_K	0.08 sec πεπειραμένη δακτυλογράφος, 0.28 sec δακτυλογράφος μέσης ικανότητας, 1.2 sec άπειρος δακτυλογράφος
Κλικ ποντικιού (mouse button press)	T_B	0.1 sec down/up, 0.2 sec κλικ
Δείξη (δηλαδή μετακίνηση του κέρσορα, τυπικά με χρήση ποντικιού) (pointing)	T_P	1.1 sec
Μετακίνηση χεριού μεταξύ πληκτρολογίου και ποντικιού (hand movement)	T_H	0.4 sec
Νοητική προετοιμασία (mental preparation)	T_M	1.35 sec
Χρόνος απόκρισης συστήματος (system response time)	T_R	περίπου 0.05 sec

Στην περίπτωση αξιολόγησης της εφαρμογής Horario σε κινητή συσκευή, το μοντέλο KLM αποδεικνύεται ανεπαρκές. Στο (A. Lee κ.ά. 2015) προτείνεται ένα νέο μοντέλο για κινητές συσκευές με οιόνη αφής (Fingerstroke-Level Model, FLM) που βασίζεται στο KLM και σε προσπάθειες επικαιροποίησής του.

Οι στοιχειώδεις ενέργειες του μοντέλου FLM, οι συμβολισμοί τους και οι εμπειρικοί μέσοι χρόνοι τους φαίνονται στον πίνακα 5.4.

Πίνακας 5.4: Κατηγορίες ενεργειών του μοντέλου FLM

Ενέργεια	Συμβολισμός	Εμπειρικός μέσος χρόνος
Άγγιγμα (tapping)	T_T	0.31 sec
Δείξη (pointing)	T_P	0.43 sec
Σύρσιμο (dragging)	T_D	0.17 sec
Γρήγορη μετακίνηση - Τίναγμα (flicking)	T_F	Από αριστερά προς δεξιά: 0.11 sec, Από δεξιά προς αριστερά: 0.12 sec
Νοητική προετοιμασία (mental preparation)	T_M	1.35 sec
Χρόνος απόκρισης συστήματος (system response time)	T_R	περίπου 0.05 sec

Θα εφαρμόσουμε το μοντέλο FLM για τους στόχους που τέθηκαν κατά την ιεραρχική ανάλυση εργασιών (βλ. Ενότητα 3.1.6).

Δημιουργία προγράμματος εργασίας

Έστω T_1 ο χρόνος προετοιμασίας προγράμματος, μέχρι πριν την έναρξη προσθήκης μεμονωμένων βαρδιών. Αποτελείται από τα βήματα:

- Μετακίνηση και άγγιγμα της καρτέλας *Schedule*: $M + P + T$
- Μετακίνηση και άγγιγμα του κουμπιού *New schedule*: $M + P + T$
- Μετακίνηση και άγγιγμα της επιθυμητής ομάδας: $M + P + T$
- Μετακίνηση και άγγιγμα πεδίου ημερομηνίας: $M + P + T$
- Μετακίνηση και άγγιγμα ημερομηνίας έναρξης και λήξης: $2 \times (M + P + T)$
- Μετακίνηση και άγγιγμα του κουμπιού *Save*: $M + P + T$
- Έλεγχος επιλεγμένων ημερομηνιών: R
- Μετακίνηση και άγγιγμα του κουμπιού *Continue*: $M + P + T$

Άρα:

$$T_1 = 8 \times (M + P + T) + R$$

Έστω T_2 ο χρόνος που απαιτείται για την προσθήκη n μεμονωμένων βαρδιών με αυτόματη ανάθεση. Για κάθε βάρδια με αυτόματη ανάθεση, αν υποθέσουμε ότι δεν τροποποιούμε τις ώρες έναρξης και λήξης και δεν προσθέτουμε σχόλιο, απαιτούνται τα βήματα:

- Μετακίνηση και άγγιγμα του κελιού βάρδιας: $M + P + T$
- Μετακίνηση και άγγιγμα του κουμπιού *Add staff*: $M + P + T$
- Κύλιση προς τα κάτω: F
- Μετακίνηση και άγγιγμα του κουμπιού *Add staff*: $M + P + T$
- ...

Άρα:

$$T_2 = \sum_{i=1}^n \{(M + P + T) + s_i \times [2 \times (M + P + T) + F]\}$$

όπου s_i είναι ο αριθμός του προσωπικού που απαιτείται για τη βάρδια i .

Έστω T_3 ο χρόνος που απαιτείται για την προσθήκη m μεμονωμένων βαρδιών με χειροκίνητη ανάθεση. Για κάθε βάρδια με χειροκίνητη ανάθεση, αν υποθέσουμε ότι βρίσκουμε τον επιθυμητό εργαζόμενο χωρίς αναζήτηση μέσω πληκτρολόγησης, δεν τροποποιούμε τις ώρες έναρξης και λήξης και δεν προσθέτουμε σχόλιο, απαιτούνται τα βήματα:

- Μετακίνηση και άγγιγμα του κελιού βάρδιας: $M + P + T$
- Μετακίνηση και άγγιγμα του κουμπιού *Add staff*: $M + P + T$
- Μετακίνηση και άγγιγμα του πεδίου επιλογής εργαζόμενου: $M + P + T$
- Κύλιση έως τον επιθυμητό εργαζόμενο: F
- Μετακίνηση και άγγιγμα του επιθυμητού εργαζόμενου: $M + P + T$
- Κύλιση προς τα κάτω: F
- Μετακίνηση και άγγιγμα του κουμπιού *Add staff*: $M + P + T$
- ...

Άρα:

$$T_3 = \sum_{i=1}^m \{(M + P + T) + s_i \times [4 \times (M + P + T) + 2 \times F]\}$$

όπου s_i είναι ο αριθμός του προσωπικού που απαιτείται για τη βάρδια i .

Έστω T_4 ο χρόνος που απαιτείται για την αυτόματη ανάθεση και τη δημοσίευση του προγράμματος. Υποθέτουμε ότι λαμβάνουμε τα αποτελέσματα της αυτόματης ανάθεσης αυτούσια, χωρίς να πραγματοποιήσουμε αλλαγές. Απαιτούνται τα βήματα:

- Μετακίνηση και áγγιγμα του κουμπιού *Auto-assign*: $M + P + T$
- Μετακίνηση και áγγιγμα του κουμπιού *Auto-assign* (στην επιβεβαίωση): $M + P + T$
- Αναμονή αυτόματης ανάθεσης: R
- Μετακίνηση και áγγιγμα του κουμπιού *Publish*: $M + P + T$
- Μετακίνηση και áγγιγμα του κουμπιού *Publish* (στην επιβεβαίωση): $M + P + T$
- Αναμονή δημοσίευσης: R

Αρο:

$$T_4 = 2 \times (M + P + T) + 2 \times R$$

Στην περίπτωση που δεν έχει επιλεγεί καμία αυτόματη ανάθεση, ο χρόνος T_4 μειώνεται σε:

$$T_4 = (M + P + T) + R$$

Τελικά, είναι:

$$T_{\text{total}} = T_1 + T_2 + T_3 + T_4$$

Ενδεικτικά, για την προσθήκη 3 βαρδιών ανά ημέρα σε μία εβδομάδα με 2 αυτόματες αναθέσεις η καθεμία (άρα $n = 21$, $m = 0$) και $s_i = 2$ για κάθε βάρδια i , έχουμε:

$$T_{\text{total}} \approx 235.5 \text{ sec.} \approx 4 \text{ min.}$$

Υποβολή αιτήματος áδειας

Η διαδικασία υποβολής αιτήματος áδειας, έστω για πολυήμερη áδεια, αποτελείται από τα βήματα:

- Μετακίνηση και áγγιγμα της καρτέλας *Requests*: $M + P + T$
- Μετακίνηση και áγγιγμα του κουμπιού *New request*: $M + P + T$
- Μετακίνηση και áγγιγμα του κουμπιού *Time-off*: $M + P + T$
- Μετακίνηση και áγγιγμα του κουμπιού *Multiple days (leave)*: $M + P + T$
- Μετακίνηση και áγγιγμα πεδίου ημερομηνίας: $M + P + T$
- Μετακίνηση και áγγιγμα ημερομηνίας έναρξης και λήξης: $2 \times (M + P + T)$
- Μετακίνηση και áγγιγμα του κουμπιού *Save*: $M + P + T$
- Μετακίνηση και áγγιγμα του κουμπιού *Continue*: $M + P + T$
- Έλεγχος αιτήματος: M
- Μετακίνηση και áγγιγμα πεδίου σχολίου: $M + P + T$

- Πληκτρολόγηση σχολίου (έστω 40 χαρακτήρες): $M + 40P + 40T$
- Μετακίνηση και άγγιγμα του κουμπιού *Submit*: $M + P + T$
- Αναμονή υποβολής: R

Άρα:

$$T_{\text{total}} = 11 \times (M + P + T) + M + M + 40P + 40T + R \approx 55 \text{ sec.}$$

Υποβολή αιτήματος ανταλλαγής βαρδιών

Η διαδικασία υποβολής αιτήματος ανταλλαγής βαρδιών αποτελείται από τα βήματα:

- Μετακίνηση και άγγιγμα της καρτέλας *Requests*: $M + P + T$
- Μετακίνηση και άγγιγμα του κουμπιού *New request*: $M + P + T$
- Μετακίνηση και άγγιγμα του κουμπιού *Shift exchange*: $M + P + T$
- Κύλιση έως την επιθυμητή βάρδια προς ανταλλαγή: F
- Μετακίνηση και άγγιγμα της επιθυμητής βάρδιας προς ανταλλαγή: $M + P + T$
- Κύλιση έως την επιθυμητή βάρδια: F
- Μετακίνηση και άγγιγμα του κελιού της επιθυμητής βάρδιας: $M + P + T$
- Επιλογή συναδέλφου: $M + P + T$
- Έλεγχος αιτήματος: M
- Μετακίνηση και άγγιγμα πεδίου σχολίου: $M + P + T$
- Πληκτρολόγηση σχολίου (έστω 40 χαρακτήρες): $M + 40P + 40T$
- Μετακίνηση και άγγιγμα του κουμπιού *Submit*: $M + P + T$
- Αναμονή υποβολής: R

Άρα:

$$T_{\text{total}} = 8 \times (M + P + T) + 2 \times F + M + M + 40P + 40T + R \approx 50 \text{ sec.}$$

5.5 Συνεντεύξεις με πραγματικούς χρήστες

Τις παραπάνω –expert-based –μεθόδους αξιολόγησης, συμπληρώνουμε και ενισχύουμε με την εμπλοκή πραγματικών τελικών χρηστών στη διαδικασία αξιολόγησης της εφαρμογής. Για τον σκοπό αυτό, πραγματοποιήσαμε μία σειρά πειραμάτων με τελικούς χρήστες, στα οποία εφαρμόσαμε τόσο το πρωτόκολλο think-aloud, όσο και τις ημιδομημένες συνεντεύξεις.

Το πρωτόκολλο ομιλούντος υποκειμένου (think-aloud protocol) προτάθηκε από τον Clayton Lewis (Lewis κ.ά. 1990). Με τη μέθοδο αυτή, ο σχεδιαστής - ερευνητής ζητά από τον χρήστη - αξιολογητή

5.5 Συνεντεύξεις με πραγματικούς χρήστες

να εκφράζει με προφορικό λόγο τις σκέψεις του, όσο χρησιμοποιεί το σύστημα. Με τον τρόπο αυτό, ο σχεδιαστής έχει τη δυνατότητα να κατανοήσει τον τρόπο με τον οποίο ο χρήστης αντιλαμβάνεται τη λειτουργία της εφαρμογής και να εντοπίσει παρερμηνείες και σχεδιαστικά λάθη (Αβούρης κ.ά. 2018).

Οι συνεντεύξεις (*interviews*) είναι ο κατ’ εξοχήν τρόπος επικοινωνίας ενός σχεδιαστή με τους τελικούς χρήστες, ώστε να λάβει ανάδραση σχετικά με το υπό αξιολόγηση προϊόν. Ειδικότερα, στις ημιδομημένες συνεντεύξεις, ο σχεδιαστής διαθέτει μία λίστα με ερωτήσεις τις οποίες θέτει στον χρήστη, ωστόσο η συζήτηση είναι όσο το δυνατόν πιο ελεύθερη, ούτως ώστε ο χρήστης να έχει τη δυνατότητα να την κατευθύνει και σε άλλους τομείς, δίνοντας στον σχεδιαστή πληροφορίες που, ενδεχομένως, να μην προέκυπταν από την αυστηρή λίστα ερωτήσεων.

Αξίζει να αναφέρουμε, ότι επιλέγοντας να διεξάγουμε συνεντεύξεις αντί για ερωτηματολόγια, ουσιαστικά επιλέξαμε την ποιοτική αξιολόγηση, δηλαδή την ανάλυση υποκειμενικών νοημάτων των χρηστών, έναντι της ποσοτικής, δηλαδή της αναζήτησης αριθμητικών δεδομένων. Έτσι, μπορεί το πλήθος των συνεντεύξεων που διεξήγαμε να μην είναι αρκετά μεγάλο για να κάνουμε στατιστική ανάλυση –άλλωστε τα δεδομένα που προέκυψαν δεν ήταν ποσοτικά –, ωστόσο τα ποιοτικά δεδομένα που λάβαμε μας έδωσαν μία σαφή εικόνα του τρόπου με τον οποίο σκέφτονται και λειτουργούν οι χρήστες. Επομένως, οι συνεντεύξεις μας φάνηκαν εξαιρετικά χρήσιμες και τις θεωρούμε ίσως το πιο σημαντικό κομμάτι της αξιολόγησης.

Στα πλαίσια αξιολόγησης της εφαρμογής **Horario**, εκτελέσαμε 3 τέτοια πειράματα ημιδομημένων συνεντεύξεων, όπου ζητήσαμε από τους χρήστες μας να υλοποιήσουν το σενάριο που βρίσκεται στο Παράρτημα και φαίνεται στις Εικόνες 7.4 και 7.5.

Με την ολοκλήρωση της διαδικασίας, ακολούθησαν ημιδομημένες συνεντεύξεις, οι οποίες βασίζονταν, αλλά δεν περιορίζονταν, στην ακόλουθη λίστα ερωτήσεων ανοιχτού τύπου:

1. Σας φάνηκαν εύκολες ή περίπλοκες οι επιμέρους διαδικασίες στην εφαρμογή; Γιατί;
2. Θεωρείτε ότι έγιναν γρήγορα ή ήταν χρονοβόρες;
3. Είχαν λογική συνέχεια οι διαδικασίες ή υπήρξε κάποιο σημείο που μπερδευτήκατε;
4. Τι σας άρεσε περισσότερο στην εφαρμογή;
5. Τι πρέπει να αλλάξει επειγόντως;
6. Τι παραπάνω θα θέλατε να μπορείτε να κάνετε μέσα από την εφαρμογή;
7. Υπάρχουν άλλες παρατηρήσεις που θέλετε να κάνετε;

Αξίζει να σημειώσουμε πως εκτελέσαμε τις συνεντεύξεις με πραγματικούς χρήστες ως ένα είδος διαμορφωτικής αξιολόγησης (formative evaluation, βλ. Ενότητα 5.1). Οι απαντήσεις των χρηστών χρησιμοποιήθηκαν για τον επανασχεδιασμό ορισμένων οθονών της εφαρμογής, καθώς κάποιες κρίθηκαν μη εύχρηστες και προκαλούσαν σύγχυση στους χρήστες. Στο Παράρτημα της εργασίας, παρατίθενται τα πρωτότυπα που δε χρησιμοποιήθηκαν στο τελικό προϊόν, αφού επανασχεδιάστηκαν, με διαφορετική λογική.

5.6 Αξιολόγηση αποδοτικότητας αλγορίθμου αυτόματης ανάθεσης βαρδιών

Πραγματοποιήσαμε, επίσης, πειράματα, ώστε να μετρήσουμε την αποδοτικότητα του αλγορίθμου αυτόματης ανάθεσης βαρδιών. Για τον σκοπό αυτό, ορίσαμε προγραμματιστικά, μέσω Python, 6 διαφορετικά προβλήματα, για να δοκιμάσουμε τις επιδόσεις του αλγορίθμου μας:

- Πρόβλημα 1: 7 ημέρες, 100 εργαζόμενοι
- Πρόβλημα 2: 14 ημέρες, 100 εργαζόμενοι
- Πρόβλημα 3: 28 ημέρες, 100 εργαζόμενοι
- Πρόβλημα 4: 7 ημέρες, 1000 εργαζόμενοι
- Πρόβλημα 5: 14 ημέρες, 1000 εργαζόμενοι
- Πρόβλημα 6: 28 ημέρες, 1000 εργαζόμενοι

Στα προβλήματα αυτά, αγνοούμε χάριν απλότητας τις προτιμήσεις περί συναδέλφων.

Χρονομετρούμε την εκτέλεση των προβλημάτων. Στον Πίνακα 5.5 φαίνονται τα αποτελέσματα του πειράματος, τα οποία παριστάνονται γραφικά στην Εικόνα 5.2¹.

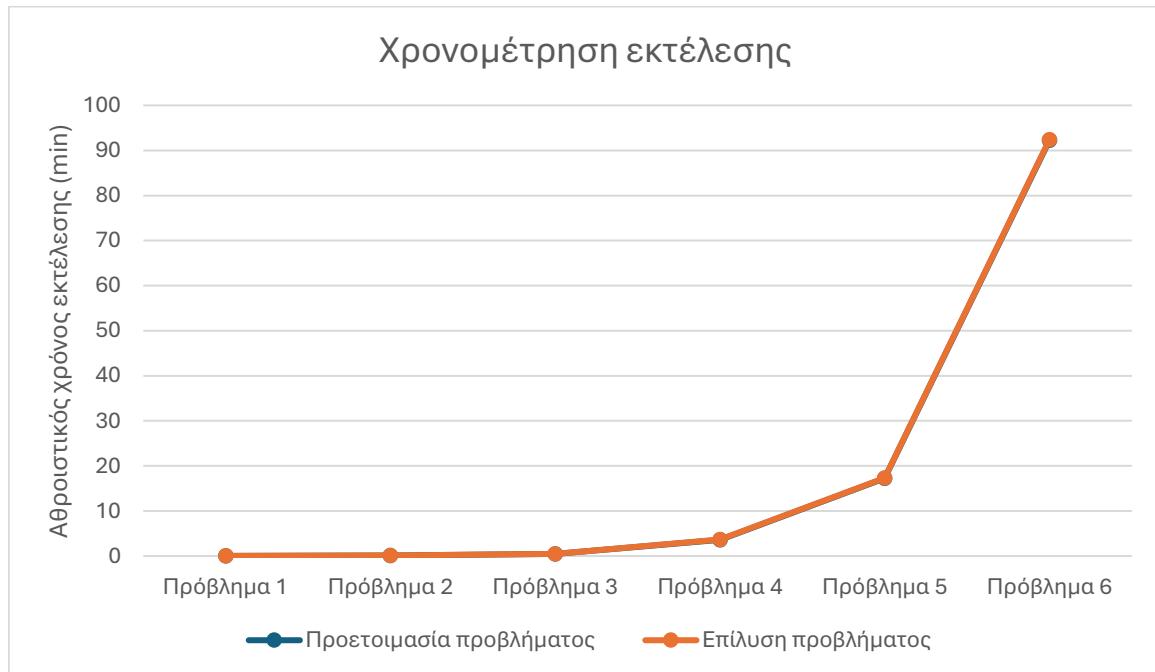
Πίνακας 5.5: Αποτελέσματα χρονομέτρησης αλγορίθμου αυτόματης ανάθεσης βαρδιών

Πρόβλημα	Χρόνος προετοιμασίας προβλήματος	Χρόνος επίλυσης προβλήματος
Πρόβλημα 1	1.67 sec	0.26 sec
Πρόβλημα 2	6.19 sec	0.46 sec
Πρόβλημα 3	26.99 sec	0.87 sec
Πρόβλημα 4	216.65 sec	0.87 sec
Πρόβλημα 5	1034.95 sec	4.37 sec
Πρόβλημα 6	5535.15 sec	9.78 sec

Παρατηρούμε ότι όσο αυξάνεται η πολυπλοκότητα του προβλήματος, αυξάνεται σημαντικά ο χρόνος που απαιτείται για την προετοιμασία του προβλήματος, δηλαδή τον ορισμό των μεταβλητών απόφασης, της αντικειμενικής συνάρτησης και των περιορισμών. Παρατηρούμε επίσης ότι ο χρόνος επίλυσης του προβλήματος παραμένει ικανοποιητικά χαμηλός, ακόμα και για αρκετά πολύπλοκα προβλήματα.

¹Για τη χρονομέτρηση, χρησιμοποιήθηκε σύστημα με επεξεργαστή Intel®Core™i5-10400H και μνήμη 16GB.

5.6 Αξιολόγηση αποδοτικότητας αλγορίθμου αυτόματης ανάθεσης βαρδιών



Εικόνα 5.2. Αποτελέσματα χρονομέτρησης αλγορίθμου αυτόματης ανάθεσης βαρδιών

Αξίζει να αναφέρουμε ότι τα μεγέθη αυτών των προβλημάτων σπάνια θα είναι της τάξεως ενός πραγματικού προβλήματος. Ο λόγος είναι ότι το πρόγραμμα βαρδιών δημιουργείται για κάθε ομάδα ξεχωριστά και οι ομάδες μίας επιχείρησης αποτελούνται από λιγότερα άτομα. Εντούτοις, οι παραπάνω χρόνοι εκτέλεσης αποτελούν χρήσιμες μετρήσεις της απόδοσης του αλγορίθμου μας.

Επιλύουμε επίσης, ένα μικρού μεγέθους πρόβλημα, ώστε να εξετάσουμε την ορθότητα της λύσης που θα προκύψει.

Έστω η ομάδα της υποδοχής ενός ξενοδοχείου, η οποία αποτελείται από 7 εργαζόμενους: Ν. Καλαμιώτης, Χ. Κολοβίνου, Ι. Νικοπολίδη, Δ. Οικονόμου, Κ. Πέτρου, Μ. Σταθούλια, Α. Τζανίδης.

Κάθε ημέρα χωρίζεται σε τρεις βάρδιες: πρωινή, απογευματινή, νυχτερινή. Μόνο οι άνδρες εκτελούν νυχτερινές βάρδιες (εκτός αν δεν υπάρχει άλλη δυνατότητα). Ο Ν. Καλαμιώτης αναλαμβάνει μόνο νυχτερινές βάρδιες. Όλοι οι υπόλοιποι προτιμούν την πρωινή βάρδια, έπειτα την απογευματινή και τέλος τη νυχτερινή. Εξαίρεση αποτελεί η Χ. Κολοβίνου, που εκτελεί μόνο πρωινές βάρδιες (εκτός αν δεν υπάρχει άλλη δυνατότητα).

Θέλουμε να ορίσουμε το πρόγραμμα της επόμενης εβδομάδας. Τις καθημερινές, η πρωινή βάρδια χρειάζεται 2 εργαζόμενους (εκτός από την Τετάρτη και την Πέμπτη, που, λόγω πολλών αναχωρήσεων, χρειάζονται 3 εργαζόμενοι), η απογευματινή 2 και η νυχτερινή 1. Το Σαββατοκύριακο, όλες οι βάρδιες χρειάζονται 1 εργαζόμενο. Έχουν ζητηθεί οι ακόλουθες άδειες: τη Δευτέρα, έχει ζητήσει ο Ν. Καλαμιώτης και η Χ. Κολοβίνου, την Τρίτη, ο Ν. Καλαμιώτης και η Μ. Σταθούλια, το Σάββατο και

5.6 Αξιολόγηση αποδοτικότητας αλγορίθμου αυτόματης ανάθεσης βαρδιών

την Κυριακή, η Ι. Νικοπολίδη και η Μ. Σταθούλια.

Ως βέλτιστη λύση του προβλήματος, προκύπτει αυτή που φαίνεται στον Πίνακα 5.6.

Πίνακας 5.6: Αποτελέσματα αναθέσεων συγκεκριμένου προβλήματος

	Δευτέρα	Τρίτη	Τετάρτη	Πέμπτη	Παρασκευή	Σάββατο	Κυριακή
Πρωί	Πέτρου, Τζανίδης	Κολοβίνου, Πέτρου	Κολοβίνου, Οικονόμου,	Κολοβίνου, Σταθούλια,	Κολοβίνου, Σταθούλια	Κολοβίνου Τζανίδης	Τζανίδης
Απόγ.	Νικοπολίδη, Σταθούλια	Οικονόμου, Νικοπολίδη	Νικοπολίδη, Σταθούλια	Πέτρου, Νικοπολίδη	Πέτρου, Νικοπολίδη	Οικονόμου, Τζανίδης	Οικονόμου
Νύχτα	Οικονόμου	Τζανίδης	Καλαμιώτης	Καλαμιώτης	Καλαμιώτης	Καλαμιώτης	Καλαμιώτης

Μπορούμε να ελέγξουμε και να επιβεβαιώσουμε ότι ικανοποιούνται όλοι οι περιορισμοί και επιπλέον ικανοποιούνται και οι προτιμήσεις των εργαζόμενων στις επιμέρους βάρδιες.

Ανάλογες τροποποιήσεις της βέλτιστης λύσης θα παίρναμε, αν καθορίζαμε προτιμήσεις συναδέλφων. Για παράδειγμα, για το ίδιο πρόβλημα, αν η Χ. Κολοβίνου έχει στη λίστα προσφιλών συναδέλφων την Κ. Πέτρου, η βέλτιστη λύση θα ήταν αυτή που φαίνεται στον Πίνακα 5.7.

Πίνακας 5.7: Αποτελέσματα αναθέσεων συγκεκριμένου προβλήματος με προτιμήσεις συναδέλφων

	Δευτέρα	Τρίτη	Τετάρτη	Πέμπτη	Παρασκευή	Σάββατο	Κυριακή
Πρωί	Πέτρου, Σταθούλια	Κολοβίνου, Πέτρου	Κολοβίνου, Πέτρου,	Κολοβίνου, Πέτρου,	Κολοβίνου, Πέτρου	Τζανίδης	Κολοβίνου
Απόγ.	Οικονόμου, Νικοπολίδη	Οικονόμου, Νικοπολίδη	Νικοπολίδη, Τζανίδης	Οικονόμου, Νικοπολίδη	Οικονόμου, Νικοπολίδη	Οικονόμου, Σταθούλια	Τζανίδης
Νύχτα	Τζανίδης	Τζανίδης	Καλαμιώτης	Καλαμιώτης	Καλαμιώτης	Καλαμιώτης	Καλαμιώτης

Παρατηρούμε ότι το πρόγραμμα έχει τροποποιηθεί έτσι, ώστε η Χ. Κολοβίνου και η Κ. Πέτρου να εργάζονται μαζί, όσο αυτό είναι δυνατόν.

6 Συμπεράσματα

Στην παρούσα διπλωματική εργασία πραγματοποιήθηκε μία πλήρης μελέτη για ένα σύστημα προγραμματισμού βάρδιας για μεγάλες επιχειρήσεις. Το ζήτημα αυτό περιλαμβάνει ένα πρόβλημα βελτιστοποίησης, το οποίο έχει στόχο την εύρεση του βέλτιστου ωρολογίου προγράμματος εργασίας των εργαζομένων σε μία επιχείρηση. Το βέλτιστο αυτό ωρολόγιο πρόγραμμα καλύπτει πλήρως τις ανάγκες της επιχείρησης σε προσωπικό, χωρίς να απασχολεί περισσότερους υπαλλήλους από όσους πραγματικά χρειάζονται, ενώ ταυτόχρονα ικανοποιεί τις προτιμήσεις κάθε εργαζόμενου, στον βαθμό που αυτό είναι δυνατό. Το σύστημα αυτό βρίσκει εφαρμογή σε οποιαδήποτε επιχείρηση οι εργαζόμενοι δουλεύουν με βάρδιες.

6.1 Σύνοψη εργασίας

Η μελέτη περιστράφηκε γύρω από δύο βασικούς άξονες: ο πρώτος ήταν η κατάστρωση του αλγορίθμου του χρονοπρογραμματισμού εργασίας από μαθηματικής πλευράς (Ενότητα 2.3). Χρησιμοποιώντας τις μεθόδους του γραμμικού προγραμματισμού, δημιουργήθηκε ένα μοντέλο, το οποίο έχει σκοπό την εύρεση αυτού του προγράμματος, το οποίο μεγιστοποιεί τις προτιμήσεις των εργαζομένων, ενώ ταυτόχρονα ικανοποιεί τα αιτήματά τους και πληροί τις ανάγκες της επιχείρησης σε προσωπικό.

Ο δεύτερος, και κυριότερος, άξονας της μελέτης αποτέλεσε η δημιουργία μίας cross-platform εφαρμογής, η οποία θα υποστηρίζει τον χρονοπρογραμματισμό εργασίας των εργαζομένων σε μία επιχείρηση. Το πρώτο βήμα ήταν η σχεδίαση της εφαρμογής, με βάση τις πρακτικές του ανθρωποκεντρικού σχεδιασμού. Αρχικά, πραγματοποιήθηκε ανάλυση των απαιτήσεων της εφαρμογής (Ενότητα 3.1). Στα πλαίσια αυτής, πραγματοποιήθηκε ανάλυση χρηστών, με τεχνικές όπως το πλαίσιο PACT, τη μέθοδο των περσόνων και τις ιστορίες χρηστών (user stories), αλλά και ανάλυση εργασιών, με τις μεθόδους της εθνογραφίας, με διερεύνηση εντός πλαισίου, και της ιεραρχικής ανάλυσης εργασιών.

Στη συνέχεια, αφού μελετήθηκαν υπάρχουσες, ανταγωνιστικές λύσεις (Ενότητα 3.2), πραγματοποιήθηκαν σχεδιαστικές επιλογές (Ενότητα 3.3), με βάση τις προδιαγραφές της καλής εμπειρίας χρήστη (user experience), λαμβάνοντας υπόψη διαφορετικές θεωρίες, όπως αυτή της θεωρίας χρωμάτων. Έπειτα, σχεδιάστηκαν σκαριφήματα και πρωτότυπα για τις επιμέρους οθόνες της εφαρμογής (Ενότητα 3.4),

6.2 Μελλοντικά βήματα

οι οποίες καλύπτουν τους στόχους των χρηστών που τέθηκαν κατά τη φάση της ανάλυσης, με τρόπο που επιτυγχάνει τη βέλτιστη εμπειρία χρήστη.

Με την ολοκλήρωση της φάσης της σχεδίασης, και αφού πραγματοποιήθηκε ένας κύκλος διαμορφωτικής αξιολόγησης, προχωρήσαμε στη φάση της υλοποίησης (Κεφάλαιο 4). Για την υλοποίηση του front-end της εφαρμογής, έγινε χρήση της αρχιτεκτονικής Model-View-ViewModel (MVVM), της οποίας απώτερος σκοπός είναι να κάνει τη διαδικασία της ανάπτυξης εύκολη και ευέλικτη, ώστε η εφαρμογή να είναι στο μέλλον εύκολα επεκτάσιμη. Παράλληλα, σχηματίστηκε το μοντέλο δεδομένων, το οποίο χρησιμοποιήθηκε για την υλοποίηση της βάσης δεδομένων, η οποία υποστηρίζει τη λειτουργία της εφαρμογής, φιλοξενώντας τα δεδομένα της στο νέφος (cloud). Για τον σκοπό αυτό, χρησιμοποιήθηκε η υπηρεσία Firebase από την Google, μέσω της οποίας πραγματοποιήσαμε επίσης και τη διαχείριση των χρηστών της εφαρμογής (αυθεντικοποίηση). Ολοκληρώνοντας τη φάση της υλοποίησης, δημιουργήσαμε έναν εξυπηρετητή που περιλαμβάνει τη μηχανή αυτόματης ανάθεσης βαρδιών σε εργαζομένους. Ο εξυπηρετητής φιλοξενήθηκε στην υπηρεσία Fly, και επικοινωνεί με την εφαρμογή μέσω ενός API, το οποίο αναπτύχθηκε με τη γλώσσα Python και τη χρήση του framework Flask.

Σημαντικό στάδιο της διαδικασίας ανάπτυξης της εφαρμογής, ωστόσο, αποτέλεσε η αξιολόγηση της προόδου, με βάση τις αρχές του ανθρωποκεντρικού σχεδιασμού (Κεφάλαιο 5). Η αξιολόγηση πραγματοποιήθηκε με μεθόδους που ανήκουν σε δύο κατηγορίες: τις μεθόδους αξιολόγησης από ειδικούς (expert-based evaluation) και τις μεθόδους αξιολόγησης από χρήστες (user-based evaluation). Συγκεκριμένα, από τις μεθόδους αξιολόγησης από ειδικούς, εφαρμόσαμε τη μέθοδο του γνωσιακού περιδιαβάσματος (cognitive walkthrough) για συγκεκριμένα σενάρια χρήσης της εφαρμογής, αλλά και το μοντέλο πληκτρολογήσεων (Keystroke-Level Model, KLM), προσαρμοσμένο για κινητές συσκευές με οθόνη αφής (Fingerstroke-Level Model, FLM). Τις μεθόδους αυτές συμπλήρωσαν και ενίσχυσαν οι μέθοδοι αξιολόγησης από πραγματικούς χρήστες της εφαρμογής, με χρήση ημιδομημένων συνεντεύξεων, τα αποτελέσματα των οποίων ήταν καθοριστικά για τη διαδικασία σχεδίασης της εφαρμογής. Τέλος, πραγματοποιήθηκε ένας κύκλος αξιολόγησης της αποδοτικότητας του αλγορίθμου αυτόματης ανάθεσης βαρδιών, με χρονομέτρηση της εκτέλεσής του σε τυχαία σενάρια, αλλά και έλεγχο των αποτελεσμάτων του σε συγκεκριμένα, απλά, σενάρια, ώστε να επιβεβαιωθεί η σωστή λειτουργία του.

6.2 Μελλοντικά βήματα

Η εφαρμογή Horario αποτελεί ένα ολοκληρωμένο σύστημα διαχείρισης βάρδιας για μεγάλες επιχειρήσεις, το οποίο μπορεί να υιοθετηθεί από ένα μεγάλο εύρος οργανισμών. Το επόμενο βήμα της ανάπτυξης της θα αποτελούσε η δοκιμαστική χρήση της σε πραγματικές συνθήκες από τους εργαζομένους μίας επιχείρησης, ώστε να παρατηρηθούν πιο διεξοδικά συγκεκριμένες πλευρές της και

να πραγματοποιηθούν βελτιώσεις. Επιπλέον, με την καθημερινή χρήση της, μπορούν να προκύψουν επιπλέον ανάγκες από τους χρήστες, που δεν περιλαμβάνονται στα πλαίσια της παρούσας εργασίας. Οι ανάγκες αυτές θα μπορούσαν να υλοποιηθούν μελλοντικά, με αποτέλεσμα η εφαρμογή **Horario** να μετατραπεί σε ένα απαραίτητο σύστημα για επιχειρήσεις πολλών κλάδων.

7 Βιβλιογραφία

- Aguinis, Martin. 2019. ‘How is Flutter different for app development’ . Google For Developers. <https://www.youtube.com/watch?v=l-YO9CmaSUM>.
- Aickelin, Uwe, και Kathryn A. Dowsland. 2004. ‘An indirect Genetic Algorithm for a nurse-scheduling problem’ . *Computers & Operations Research* 31 (5): 761–78. [https://doi.org/https://doi.org/10.1016/S0305-0548\(03\)00034-0](https://doi.org/10.1016/S0305-0548(03)00034-0).
- Anderson, Chris. 2012. ‘The Model-View-ViewModel (MVVM) Design Pattern’ . Στο *Pro Business Applications with Silverlight 5*, 461–99. Berkeley, CA: Apress. https://doi.org/10.1007/978-1-4302-3501-9_13.
- Batagoda, Muditha. 2018. ‘Usability for designers, P-A-C-T framework’ . *Medium*. UX Planet. <https://uxplanet.org/usability-for-designers-p-a-c-t-framework-20509afc5f57>.
- Berrada, Ilham, Jacques A. Ferland, και Philippe Michelon. 1996. ‘A multi-objective approach to nurse scheduling with both hard and soft constraints’ . *Socio-Economic Planning Sciences* 30 (3): 183–93. [https://doi.org/https://doi.org/10.1016/0038-0121\(96\)00010-9](https://doi.org/https://doi.org/10.1016/0038-0121(96)00010-9).
- Beyer, Hugh, και Karen Holtzblatt. 1997. *Contextual Design: Defining Customer-Centered Systems*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.
- Brown, Gerald G., και Robert F. Dell. 2007. ‘Formulating Integer Linear Programs: A Rogues’ Gallery’ . *INFORMS Transactions on Education* 7 (2): 153–59. <https://doi.org/10.1287/ited.7.2.153>.
- Card, Stuart K., Allen Newell, και Thomas P. Moran. 1983. *The Psychology of Human-Computer Interaction*. USA: L. Erlbaum Associates Inc.
- Choy, Murphy, και Michelle Cheong. 2012. ‘A Flexible Mixed Integer Programming framework for Nurse Scheduling’ . arXiv. <https://doi.org/10.48550/ARXIV.1210.3652>.
- Copperchips. 2023. ‘What is flutter, and what are its uses?’ *Medium*. Copperchips. <https://medium.com/@copperchips/what-is-flutter-and-what-are-its-uses-90ae2ff29e71>.
- Delia, Lisandro, Nicolas Galdamez, Pablo Thomas, Leonardo Corbalan, και Patricia Pesado. 2015. ‘Multi-platform mobile application development analysis’ . Στο *2015 IEEE 9th International Conference on Research Challenges in Information Science (RCIS)*. IEEE. <https://doi.org/10.1109/rcis.2015.7128878>.

-
- Elmasri, Ramez, και Shamkant B Navathe. 2015. *Fundamentals of database systems*. 7ο έκδ. Upper Saddle River, NJ: Pearson.
- ‘Flutter Official Website’ . χχ. Google. <https://flutter.dev/>.
- García, Raúl Ferrer. 2023. ‘The Model-View-ViewModel (MVVM) Design Pattern’ . Στο *iOS Architecture Patterns*, 145–224. Apress. https://doi.org/10.1007/978-1-4842-9069-9_4.
- Gavalas, Damianos, και Daphne Economou. 2011. ‘Development Platforms for Mobile Applications: Status and Trends’ . *IEEE Software* 28 (1): 77–86. <https://doi.org/10.1109/MS.2010.155>.
- Google, Community. χχ. ‘Material Design’ . <https://m3.material.io/>.
- Gossman, John. 2005. ‘Tales from the Smart Client: Introduction to Model/View/ViewModel pattern for building WPF apps’ . <https://learn.microsoft.com/en-us/archive/blogs/johngossman/introduction-to-modelviewviewmodel-pattern-for-building-wpf-apps>.
- Gregersen, Erik. 2023. ‘Application software’ . Στο *Encyclopaedia Britannica*. <https://www.britannica.com/technology/application-software>.
- IBM Topics. χχ. ‘What is an API?’ <https://www.ibm.com/topics/api>.
- Jafari, Hamed, και Nasser Salmasi. 2015. ‘Maximizing the nurses’ preferences in nurse scheduling problem: mathematical modeling and a meta-heuristic algorithm’ . *Journal of Industrial Engineering International* 11 (3): 439–58. <https://doi.org/10.1007/s40092-015-0111-0>.
- Jaumard, Brigitte, Frédéric Semet, και Tsevi Vovor. 1998. ‘A generalized linear programming model for nurse scheduling’ . *European Journal of Operational Research* 107 (1): 1–18. [https://doi.org/https://doi.org/10.1016/S0377-2217\(97\)00330-5](https://doi.org/https://doi.org/10.1016/S0377-2217(97)00330-5).
- Kemp, Simon. 2022. ‘Digital 2022: Time Spent Using Connected Tech Continues To Rise’ . <https://datareportal.com/reports/digital-2022-time-spent-with-connected-tech>.
- Keshav. 2021. ‘Beginners Guide for Flutter.’ *Medium*. Nerd for Tech. <https://medium.com/nerd-for-tech/beginners-guide-for-flutter-dc27ee7d78a3>.
- Kumar, Mr. B.Satheesh, Ms. G Nagalakshmi, και Dr. S Kumaraguru. 2014. ‘A Shift Sequence for Nurse Scheduling Using Linear Programming Problem’ . *IOSR Journal of Nursing and Health Science* 3 (6): 24–28. <https://doi.org/10.9790/1959-03612428>.
- Lee, Ahreum, Kibum Song, Hokyung Blake Ryu, Jieun Kim, και Gyuhyun Kwon. 2015. ‘Fingerstroke time estimates for touchscreen-based mobile gaming interaction’ . *Human Movement Science* 44: 211–24. <https://doi.org/https://doi.org/10.1016/j.humov.2015.09.003>.
- Lee, JaYoung. 2023. ‘Flutter 2023 Q1 survey —API breaking changes, deep linking, and more’ . *Medium*. Flutter. <https://medium.com/flutter/flutter-2023-q1-survey-api-breaking-changes-deep-linking-and-more-7ff692f974e0>.

-
- Legrain, Antoine, Hocine Bouarab, και Nadia Lahrichi. 2014. ‘The Nurse Scheduling Problem in Real-Life’ . *Journal of Medical Systems* 39 (1). <https://doi.org/10.1007/s10916-014-0160-8>.
- Leonard, J. 1999. *Systems engineering fundamentals: supplementary text*. AD-a372 635. Defense Systems Management College. <https://books.google.gr/books?id=qixAUYaA6CUC>.
- Lewis, Clayton, Peter Poison, Cathleen Wharton, και John Rieman. 1990. ‘Testing a walkthrough methodology for theory-based design of walk-up-and-use interfaces’ . Conference paper. *Conference on Human Factors in Computing Systems - Proceedings*, 235–42. <https://doi.org/10.1145/97243.97279>.
- Lou, Tian. 2016. ‘A comparison of Android Native App Architecture: MVC, MVP and MVVM’ . MA. thesis, Eindhoven University of Technology.
- Mascetti, Sergio, Mattia Ducci, Niccolò Cantù, Paolo Pecis, και Dragan Ahmetovic. 2021. ‘Developing Accessible Mobile Applications with Cross-Platform Development Frameworks’ . Στο *Proceedings of the 23rd International ACM SIGACCESS Conference on Computers and Accessibility*. ASSETS ’21. New York, NY, USA: Association for Computing Machinery. <https://doi.org/10.1145/3441852.3476469>.
- McGrath, Stephen Keith. 2021. ‘Stakeholders’ . Στο *Speaking Management: How to Spot Language Traps and Resolve Contested Management Terms*, 301–16. Singapore: Springer Nature Singapore. https://doi.org/10.1007/978-981-16-2213-7_17.
- MongoDB. χ.χ. ‘Database Management Systems’ . <https://www.mongodb.com/resources/basics/database-management-system>.
- Morville, Peter. χ.χ. ‘User Experience Design’ . Semantic Studios. https://semanticstudios.com/user_experience_design/.
- Nielsen, Jakob. 1994. *Usability Engineering*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.
- Qiany, Shirley. 2020. ‘A Personas Guideline, From What They Are to How To Use’ . Medium. UX Collective. <https://uxdesign.cc/while-we-are-talking-about-personas-what-exactly-are-we-talking-525a645eb61a>.
- Rehkopf, Max. χ.χ. ‘User stories with examples and a template’ . <https://www.atlassian.com/agile/project-management/user-stories>.
- Ries, A., και J. Trout. 2001. *Positioning: The Battle for Your Mind*. Economia e discipline aziendali. McGraw-Hill Education. <https://books.google.gr/books?id=QupddZiRKkUC>.
- Robertson, Kim. 1989. ‘Strategically Desirable Brand Name Characteristics’ . *Journal of Consumer Marketing* 6 (4): 61–71. <https://doi.org/10.1108/eum0000000002563>.
- Rosenbloom, E. S., και N. F. Goertzen. 1987. ‘Cyclic nurse scheduling’ . *European Journal of Operational Research* 31 (1): 19–23. [https://doi.org/10.1016/0377-2217\(87\)90131-7](https://doi.org/10.1016/0377-2217(87)90131-7).

-
- Shah, Kewal, Harsh Sinha, και Payal Mishra. 2019. ‘Analysis of Cross-Platform Mobile App Development Tools’ . Στο *2019 IEEE 5th International Conference for Convergence in Technology (I2CT)*, 1–7. <https://doi.org/10.1109/I2CT45611.2019.9033872>.
- Silberschatz, Abraham, Henry F Korth, και S Sudarshan. 2010. *Database System Concepts*. 6ο έκδ. New York, NY: McGraw-Hill Professional.
- Spencer, Rick. 2000. ‘The Streamlined Cognitive Walkthrough Method, Working around Social Constraints Encountered in a Software Development Company’ . Στο *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 353–59. CHI ’ 00. New York, NY, USA: Association for Computing Machinery. <https://doi.org/10.1145/332040.332456>.
- StatCounter GlobalStats. 2023. ‘Operating System Market Share Worldwide - November 2023’ . <https://gs.statcounter.com/os-market-share>.
- Statista Research Department. 2023. ‘Operating systems - statistics & facts’ . <https://www.statista.com/topics/1003/operating-systems/#topicOverview>.
- Trilling, Lorraine, Alain Guinet, και Dominique Le Magny. 2006. ‘NURSE SCHEDULING USING INTEGER LINEAR PROGRAMMING AND CONSTRAINT PROGRAMMING’ . *IFAC Proceedings Volumes* 39 (3): 671–76. <https://doi.org/10.3182/20060517-3-FR-2903.00340>.
- Valouxis, C, και E Housos. 2000. ‘Hybrid optimization techniques for the workshift and rest assignment of nursing personnel’ . *Artificial Intelligence in Medicine* 20 (2): 155–75. [https://doi.org/10.1016/S0933-3657\(00\)00062-2](https://doi.org/10.1016/S0933-3657(00)00062-2).
- Van den Bergh, Jorne, Jeroen Beliën, Philippe De Bruecker, Erik Demeulemeester, και Liesje De Boeck. 2013. ‘Personnel scheduling: A literature review’ . *European Journal of Operational Research* 226 (3): 367–85. <https://doi.org/10.1016/j.ejor.2012.11.029>.
- Xanthopoulos, Spyros, και Stelios Xinogalos. 2013. ‘A Comparative Analysis of Cross-Platform Development Approaches for Mobile Applications’ . Στο *Proceedings of the 6th Balkan Conference in Informatics*, 213–20. BCI ’ 13. New York, NY, USA: Association for Computing Machinery. <https://doi.org/10.1145/2490257.2490292>.
- Yasar, Kinza, και Linda Rosencrance. 2022. ‘What is a software development kit (SDK)?’ Οκτώβριος 2022. <https://www.techtarget.com/whatis/definition/software-developers-kit-SDK>.
- You, Dongliang, και Minjie Hu. 2021. ‘A comparative study of cross-platform mobile application development’ . *Wellington, New Zealand* 66.
- Αβούρης, Νικόλαος, Χρήστος Κατσάνος, Νικόλαος Τσέλιος, και Κωνσταντίνος Μουστάκας. 2018. *Εισαγωγή στην Αλληλεπίδραση Ανθρώπου-Υπολογιστή*. Εκδόσεις Πανεπιστημίου Πατρών.
- Κέντρο Πληροφόρησης Εργαζομένων και Ανέργων, Γενική Συνομοσπονδία Εργατών Ελλάδος. χ.χ. ‘Βάρδιες και εργασία’ . <https://www.kepea.gr/aarticle.php?id=152>.

Σίσκος, Γιάννης. 2000. *Γραμμικός Προγραμματισμός*. Εκδόσεις Νέων Τεχνολογιών.

Παράρτημα

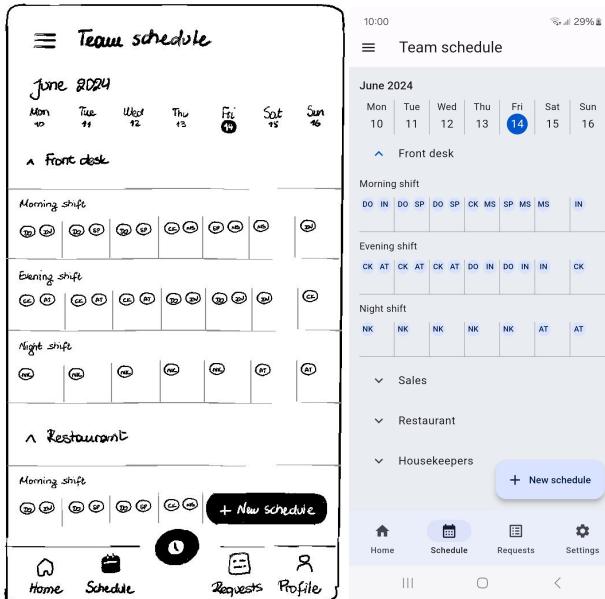
Πρωτότυπα που επανασχεδιάστηκαν μετά την αξιολόγηση

Στην Ενότητα αυτή, παρατίθενται τα σκαριφήματα και τα πρωτότυπα των οθονών, τα οποία κατά την πρώτη φάση της αξιολόγησης κρίθηκαν ότι προσφέρουν χαμηλή εμπειρία χρήστη. Για τον λόγο αυτό, επανασχεδιάστηκαν με διαφορετική λογική, λαμβάνοντας υπόψη τις προτάσεις των χρηστών που συμμετείχαν ως αξιολογητές.

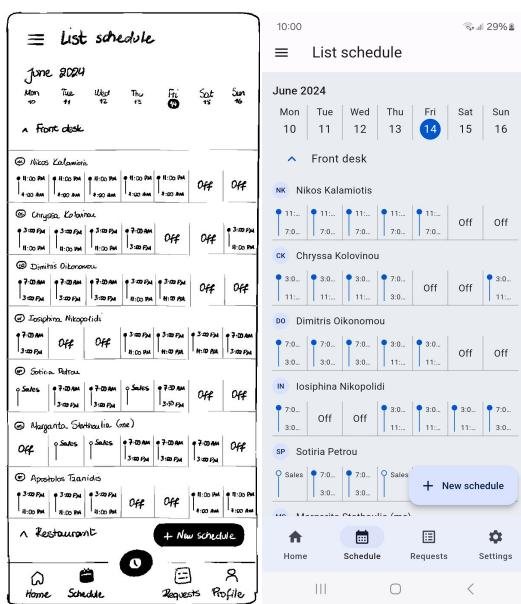
Στην Εικόνα 7.1 βλέπουμε το σκαρίφημα και το πρωτότυπο της Ομαδικής προβολής του προγράμματος εργασίας. Σε αυτή τη φάση του σχεδιασμού, η ομαδική προβολή περιελάμβανε σε μία ενιαία οθόνη όλες τις ομάδες της επιχείρησης σε μορφή ακορντεόν, ώστε ο χρήστης να μπορεί να επιλέγει ποιες θα είναι ορατές κάθε στιγμή. Η αξιολόγηση έδειξε ότι οι χρήστες δυσκολεύονταν να αντιληφθούν ότι πρόκειται για έναν ενιαίο πίνακα, του οποίου ορισμένα κελιά είναι συγχωνευμένα (οι γραμμές του κουμπιών ανάπτυξης και σύμπτυξης του ακορντεόν). Επίσης, τα ονόματα των βαρδιών εμφανίζονταν πάνω από τις αντίστοιχες γραμμές του πίνακα, για εξοικονόμηση πλάτους, γεγονός που επίσης προκαλούσε σύγχυση στους χρήστες. Για τον λόγο αυτό, η Ομαδική προβολή επανασχεδιάστηκε και απλούστεύτηκε, ώστε να περιλαμβάνει μόνο μία ομάδα κάθε φορά, προσφέροντας λιγότερη και πιο ξεκάθαρη πληροφορία.

Στην Εικόνα 7.2 βλέπουμε το σκαρίφημα και το πρωτότυπο της Προβολής λίστας του προγράμματος εργασίας, η οποία απορρίφθηκε σε επόμενο στάδιο της σχεδίασης. Η προβολή αυτή είχε σκοπό να συνδυάσει την πληροφορία που προσφέρουν η Ατομική και η Ομαδική προβολή, ώστε να μπορεί να γίνει γρήγορος έλεγχος πότε δουλεύει ο κάθε εργαζόμενος ξεχωριστά εντός μίας ομάδας. Η προβολή αυτή περιλαμβάνει στοιχεία της Ομαδικής προβολής, όπως το ακορντεόν, αλλά οργανώνει την πληροφορία με βάση τους εργαζόμενους εντός κάθε ομάδας. Αποδείχθηκε ότι οι χρήστες δεν μπορούσαν να αντιληφθούν την πληροφορία που προσφέρει αυτή η οθόνη και την συνέχειαν με αυτή της Ομαδικής προβολής. Για αυτόν τον λόγο, απορρίφθηκε και ενσωματώθηκε στην Ατομική προβολή, στην οποία πλέον μπορεί να επιλεγεί ο επιθυμητός χρήστης (πέρα από τον τρέχοντα λογαριασμό), μέσω των κουμπιών της οθόνης επιλογής προβολής.

Στην Εικόνα 7.3 βλέπουμε το σκαρίφημα και το πρωτότυπο ενός βήματος του Οδηγού δημιουργίας και επεξεργασίας προγράμματος. Πρόκειται για το βήμα της προσθήκης βαρδιών, το οποίο βασίζεται στο πρωτότυπο της Ομαδικής προβολής της Εικόνας 7.1. Επιπλέον, ορατό είναι το Φύλλο

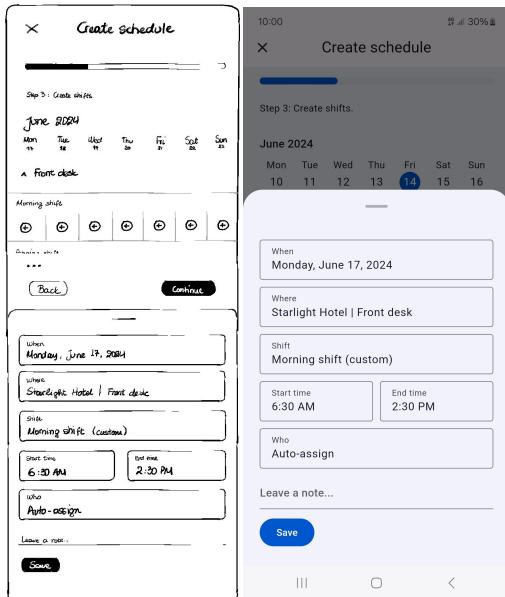


Εικόνα 7.1. Παλαιό πρωτότυπο της Ομαδικής προβολής του προγράμματος εργασίας



Εικόνα 7.2. Απορριφθέν πρωτότυπο της Προβολής λίστας του προγράμματος εργασίας

προσθήκης βάρδιας, το οποίο περιλαμβάνει πεδία καθορισμού λεπτομερειών για τη συγκεκριμένη βάρδια. Ωστόσο, για λόγους συνοχής της γραφικής διεπαφής σε όλη την έκταση της εφαρμογής, το συγκεκριμένο πρωτότυπο απορρίφθηκε και συγχωνεύθηκε με το Φύλλο βάρδιας, στο οποίο προστέθηκε δυνατότητα επεξεργασίας με τη μορφή καρτών.



Εικόνα 7.3. Απορριφθέν πρωτότυπο του Φύλλου προσθήκης βάρδιας στον Οδηγό δημιουργίας / επεξεργασίας προγράμματος εργασίας

Σενάριο αξιολόγησης εμπειρίας χρήστη με πραγματικούς χρήστες

Στην Ενότητα αυτή, παρατίθεται το έντυπο που μοιράστηκε στους αξιολογητές της εφαρμογής, κατά τη διάρκεια της αξιολόγησης από τελικούς χρήστες (user-based evaluation).

HORARIO

Scheduling made easy.

Αξιολόγηση εμπειρίας χρήστη



**Μαργαρίτα
Σταθούλια**

FRONT OFFICE MANAGER

Ηλικία: 28
Φέντα: Γυναίκα
Οράδες: Front desk, Sales



ΒΙΟΓΡΑΦΙΑ

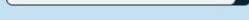
- Εργάζομαι στο Starlight Hotel τα τελευταία 4 χρόνια.
- Φροντίζω πάντα οι πελάτες και το προσωπικό να είναι ευχαριστημένοι.
- Καταρτίζω το εβδομαδιαίο πρόγραμμα εργασίας σε βάρδιες.

ΚΕΡΔΗ (GAINS)

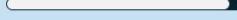
- Χαίρομαι όταν το πρόγραμμα που φτάγων υιοντούει το προσωπικό.
- Μου αρέσει να δουλεύω το πρωί, αλλά κατ' εκάρεση κάνω και απογευματινές βάρδιες, ώστε να εξυπηρετήσω όλους.
- Απολαμβάνω να δουλεύω με τη Χρήστα, είμαστε καλές φίλες.

ΠΡΟΣΩΠΙΚΟΤΗΤΑ

Εξουειδώστη με την τεχνολογία



Υπομονή



Προσφημοστικότητα σε ξαφνικές αλλαγές

ΔΥΣΚΟΛΙΕΣ (PAINS)

- Έχω πολλές αρμοδιότητες. Συνήθως πιέζω μα τη φτάνω το πρόγραμμα.
- Δυσκολεύομαι όταν κάποιος μου αλλάζει τα πλάνα τελευταία στιγμή. Θέλω να τα έχω όλα οργανωμένα.
- Δε μου αρέσει να δουλεύω με τον Δημήτρη. Νομίζει ότι τα ξέρει όλα.

“

Σήμερα είναι Πέμπτη, οπότε όλοι περιμένουν να τους ανακοινώσω το πρόγραμμα της επόμενης εβδομάδας. Πρέπει να ελέγχω τα σημειώματα που μου έχουν αφήσει, για να ξέρω πότε να βάλω εργάσιμη τημέρα στον καθένα. Έχει ήδη βραδιάσει και τα σημειώματα είναι πάρα πολλά. Πότε θα προλάβω;

Υποθέστε πως είστε η Μαργαρίτα, Front Office Manager του ξενοδοχείου Starlight Hotel. Επισκέπτεστε σήμερα την εφαρμογή **Horario**, προκειμένου να ολοκληρώσετε τις παρακάτω εργασίες:

Εργασία 1: Αλλαγή γλώσσας

Η εφαρμογή ξεκινά σε μία γλώσσα που δεν καταλαβαίνετε. Αλλάξτε τη γλώσσα σε Αγγλικά ή Ελληνικά.

Εργασία 2: Έλεγχος αιτημάτων προσωπικού

Ελέγχτε τα αιτήματα που έχει πραγματοποιήσει το προσωπικό για ρεπό και άδειες. Αποδεχτείτε το αίτημα με κωδικό **3PP473** και σχολιάστε: «Είσαι ελεύθερη να ξεκουραστείς!».

Εργασία 3: Καταχώριση αιτήματος

Επιθυμείτε να καταχωρίσετε ένα αίτημα για άδεια, την περίοδο από 12 Αυγούστου 2024 έως 18 Αυγούστου 2024. Αφήστε το σχόλιο: «Παρακαλώ αποδεχτείτε το αίτημά μου, καθώς πρέπει να παραστώ σε γάμο συγγενικού μου προσώπου.».

Εικόνα 7.4. Σενάριο αξιολόγησης εμπειρίας χρήστη με πραγματικούς χρήστες



Εργασία 4: Έλεγχος υπάρχοντος προγράμματος εργασίας

Ελέγχετε με ποιον δουλεύατε μαζί την Πέμπτη 13 Ιουνίου 2024.

Ελέγχετε ποιος εργαζόταν στην υποδοχή το πρωί της Τρίτης 11 Ιουνίου 2024.

Ελέγχετε πού και τι ώρα εργάζεται η Κ. Πέτρου σήμερα.

Εργασία 5: Έκδοση προγράμματος εργασίας

Έκδωστε ένα πρόγραμμα εργασίας για την εβδομάδα 24 Ιουνίου 2024 – 30 Ιουνίου 2024. Θα χρειαστείτε τις ακόλουθες βάρδιες:

Στην υποδοχή:

- Δευτέρα πρωί: Μ. Σταθούλια, Χ. Κολοβίνου
- Δευτέρα απόγευμα: Ι. Νικοπολίδη, Ν. Καλαμιώτης
- Δευτέρα βράδυ: Δ. Οικονόμου
- Τρίτη πρωί: Μ. Σταθούλια, Χ. Κολοβίνου
- Τρίτη απόγευμα: Ι. Νικοπολίδη, Ν. Καλαμιώτης
- Τρίτη βράδυ: Δ. Οικονόμου
- ...

Στις πωλήσεις:

- Δευτέρα: Κ. Πέτρου
- Τρίτη: Κ. Πέτρου
- ...

Λόγω αυξημένης κίνησης, την Τρίτη η Ιωσηφίνα θα χρειαστεί να έρθει 2 ώρες νωρίτερα από την προκαθορισμένη έναρξη της βάρδιάς της. Θα σχολάσει όμως και 2 ώρες νωρίτερα.

Εργασία 6: Επεξεργασία κανόνων και προτιμήσεων προσωπικού

Μόλις κλείσατε μία νέα συμφωνία με τη Χ. Κολοβίνου, η οποία ορίζει ότι απαγορεύεται να εργάζεται σε νυχτερινές βάρδιες, σε οποιαδήποτε ομάδα. Δημιουργήστε έναν νέο κανόνα για τον αλγόριθμο αυτόματης ανάθεσης βαρδιών.

Ορίστε επίσης για εσάς τους προσφίλείς και μη προσφίλείς σας συναδέλφους.

Η Α. Ηλιοπούλου πήρε προαγωγή! Πέρα από Housekeeping Manager, έγινε μόλις διευθύντρια και του εστιατορίου. Ενημερώστε το σύστημα για αυτή την αλλαγή.

Εικόνα 7.5. Σενάριο αξιολόγησης εμπειρίας χρήστη με πραγματικούς χρήστες (συνέχεια)

