

Περιεχόμενα

1 Διαχείριση Εργασιών	1
1.1 Το πρόβλημα της διαχείρισης εργασιών	1
1.1.1 Ορισμός της εργασίας	2
1.1.2 Ορισμός της διαχείρισης εργασιών	2
1.1.3 Πεδίο εφαρμογής διαχείρισης εργασιών	2
1.1.4 Διαφορά με διαχείριση έργου	3
1.2 Ιστορική αναδρομή	3
1.2.1 Προφορικότητα και μνημονικές συσκευές	4
1.2.2 Πρώτα ημερολόγια	5
1.2.3 Σύγχρονη εποχή	7
1.3 Η συνδρομή της τεχνολογίας	8
1.3.1 Ψηφιακά εργαλεία	8
1.3.1.1 Todoist	9
1.3.1.2 Notion	9
1.3.1.3 Microsoft Project	10
1.3.1.4 Trello	12
1.4 Μεθοδολογίες	12
1.4.1 Διάγραμμα Γκαντ	13
1.4.2 Program evaluation and review technique (PERT)	14
1.4.3 Kanban	16
1.5 Διαχείριση εργασιών στο πανεπιστήμιο	17
1.5.1 Προβλήματα διαχείρισης που αντιμετωπίζουν οι φοιτητές	18
1.5.2 Χαρακτηριστικά που οι φοιτητές θα επιθυμούσαν σε μια εφαρμογή	18
2 Low-Code	20
2.1 Τι είναι ο χαμηλός κώδικας	20
2.1.1 Οπτικός προγραμματισμός (visual programming)	21
2.1.2 Καθόλου κώδικας (no-code)	23
2.1.3 Η έννοια του προγραμματιστή πολίτη (citizen developer)	24
2.1.3.1 Διαφορά με τους παραδοσιακούς προγραμματιστές	25
2.1.3.2 Χαρακτηριστικά των προγραμματιστών πολιτών	25
2.2 Πώς φτάσαμε στον χαμηλό κώδικα	26

2.2.1	Computer-Aided Software Engineering (CASE)	28
2.2.2	Model-driven Architecture (MDA)	30
2.2.3	Προγενέστερα λογισμικά οπτικού προγραμματισμού	31
2.3	Πλατφόρμες Ανάπτυξης Εφαρμογών σε Low-Code (LCDP)	32
2.3.1	Χαρακτηριστικά των LCDP	34
2.3.2	Δημοφιλή LCDP	34
2.3.2.1	Mendix	34
2.3.2.2	OutSystems	34
2.3.2.3	Microsoft Power Apps	34
3	Mendix	35
3.1	Τι είναι το Mendix;	35

Κεφάλαιο 1

Διαχείριση Εργασιών

“Plans are nothing; planning is everything”

—Dwight D. Eisenhower

Όπως αναφέρθηκε στην εισαγωγή, η παρούσα διπλωματική εργασία επικεντρώνεται στην ανάπτυξη μιας εφαρμογής για τον προγραμματισμό και την παρακολούθηση εργασιών. Οι διαδικασίες σχεδιασμού, υλοποίησης και παρακολούθησης αυτών των εργασιών εντάσσονται στο ευρύτερο πλαίσιο της διαχείρισης εργασιών (task management). Για τον λόγο αυτό, στο παρόν κεφάλαιο κρίνεται απαραίτητο να παρουσιαστεί μια πιο αναλυτική ερμηνεία του όρου, καθώς και οι θεμελιώδεις αρχές και μέθοδοι που σχετίζονται με τη συγκεκριμένη έννοια.

Η διαχείριση εργασιών αποτελεί ουσιώδες στοιχείο της καθημερινότητας, τόσο σε προσωπικό όσο και σε επαγγελματικό επίπεδο. Με την αυξανόμενη πολυπλοκότητα των σύγχρονων υποχρεώσεων και την ανάγκη για αποτελεσματικό συντονισμό πολλαπλών δραστηριοτήτων, αναδύονται νέες προκλήσεις που καθιστούν τη διαδικασία αυτή ολοένα και πιο απαιτητική.

Στο πλαίσιο του κεφαλαίου αυτού, θα εξεταστεί αναλυτικά η διαδικασία διαχείρισης εργασιών, η ιστορική της εξέλιξη, οι μεθοδολογίες που βελτιώνουν την αποδοτικότητά της και ο ρόλος που διαδραματίζει στην πανεπιστημιακή κοινότητα. Ο στόχος είναι να αναδειχθεί η σημασία της έννοιας αυτής για την καθημερινότητα, με ιδιαίτερη έμφαση στους φοιτητές, καθώς αποτελεί τη βάση της λογικής για την ανάπτυξη της εφαρμογής που θα παρουσιαστεί στη συνέχεια.

1.1 Το πρόβλημα της διαχείρισης εργασιών

Στην παρούσα ενότητα θα οριστεί η έννοια της διαχείρισης εργασιών, εστιάζοντας στις διαφορές της από τη διαχείριση έργου. Παράλληλα, θα αναλυθούν τα κύρια πεδία εφαρμογής της, προκειμένου να αποκτηθεί μια ολοκληρωμένη κατανόηση του όρου.

1.1.1 Ορισμός της εργασίας

Εργασία (task, project) στη διαχείριση εργασιών ονομάζεται μια προσωρινή δραστηριότητα που αναλαμβάνεται για τη δημιουργία ενός μοναδικού αποτελέσματος (προϊόντος, υπηρεσίας, αναφοράς ή κάποιου άλλου παραδοτέου). Η εργασία πραγματοποιείται σε μια προκαθορισμένη χρονική περίοδο και τερματίζεται όταν έχει γίνει η επίτευξη των στόχων, όταν δεν υπάρχει πλέον ανάγκη για την επίτευξη των στόχων ή όταν υπάρχει η σαφής εκτίμηση ότι οι στόχοι δεν μπορούν να επιτευχθούν. [14]

1.1.2 Ορισμός της διαχείρισης εργασιών

Η διαχείριση εργασιών (task management) ορίζεται ως η διαδικασία οργάνωσης, ιεράρχησης και παρακολούθησης των επιμέρους εργασιών καθ' όλη τη διάρκειά τους, από τον αρχικό σχεδιασμό έως την ολοκλήρωσή τους, με στόχο τη διασφάλιση της αποδοτικής και αποτελεσματικής εκτέλεσής τους. [14]

Η διαδικασία αυτή αποτελεί θεμελιώδες στοιχείο για τη βελτίωση της παραγωγικότητας, τόσο σε ατομικό όσο και σε συλλογικό επίπεδο. Αν και παραδοσιακά η διαχείριση εργασιών πραγματοποιούνται χειροκίνητα, η ραγδαία ανάπτυξη της τεχνολογίας έχει καταστήσει τα ψηφιακά εργαλεία τον κύριο τρόπο υλοποίησής της, προσφέροντας αυξημένη ευελιξία και ακρίβεια.

1.1.3 Πεδίο εφαρμογής διαχείρισης εργασιών

Η διαχείριση εργασιών διαθέτει ένα ευρύ πεδίο εφαρμογής, καλύπτοντας δραστηριότητες που κυμαίνονται από απλές καθημερινές υποχρεώσεις έως τη διαχείριση σύνθετων και απαιτητικών εργασιών.

Μια από τις πιο άμεσες και προσιτές εφαρμογές της συναντάται στον καθημερινό προσωπικό προγραμματισμό. Σε αυτό το πλαίσιο, περιλαμβάνει τη χρήση εργαλείων όπως λίστες υποχρεώσεων (to-do lists), ημερολόγια ή ψηφιακές εφαρμογές¹ (π.χ. Todoist [36], Notion [41], Google Keep), τα οποία διευκολύνουν την οργάνωση προσωπικών υποχρεώσεων, τον προγραμματισμό ραντεβού ή δραστηριοτήτων αναψυχής, καθώς και τη διαχείριση στόχων. Μέσα από τέτοιες πλατφόρμες, δίνεται η δυνατότητα τόσο για μακροπρόθεσμο όσο και για μακροπρόθεσμο σχεδιασμό, ενώ παρέχεται και η ευκολία παρακολούθησης της προόδου.

Στα επαγγελματικά περιβάλλοντα, η διαχείριση εργασιών διαδραματίζει καθοριστικό ρόλο, συμβάλλοντας στη βελτίωση της συνεργασίας και της συνολικής αποδοτικότητας. Κεντρικός στόχος της είναι η ορθολογική κατανομή του φόρτου εργασίας, η διασφάλιση του συντονισμού μεταξύ των μελών μιας ομάδας και η οργάνωση των καθημερινών δραστηριοτήτων. Μέσω της διαχείρισης εργασιών καθίσταται δυνατή η διευθέτηση παράλληλων εργασιών, η ιεράρχησή τους με βάση την προτεραιότητα (επείγουσες ή μη), καθώς και η δίκαιη και στοχευμένη ανάθεση καθηκόντων στους

¹Θα αναφερθούμε πιο εκτεταμένα σε ψηφιακά εργαλεία στην ενότητα 1.3.1

εργαζομένους. Ένα από τα πλέον δημοφιλή εργαλεία που χρησιμοποιούνται σε ομαδικά επαγγελματικά περιβάλλοντα για τον σκοπό αυτό είναι το Trello [20], το οποίο παρέχει ευελιξία και οπτική οργάνωση των εργασιών.

Τέλος, η ραγδαία εξέλιξη της τεχνολογίας έχει οδηγήσει στη δημιουργία προηγμένων πλατφορμών που αξιοποιούν σύγχρονες τεχνολογίες, όπως η τεχνητή νοημοσύνη και η ανάλυση δεδομένων, για τη βελτιστοποίηση της διαχείρισης εργασιών. Αυτές οι πλατφόρμες υπερβαίνουν τις παραδοσιακές μεθόδους οργάνωσης καθώς είναι σε θέση να αναγνωρίζουν πρότυπα, να προβλέπουν χρονικές απαιτήσεις, να ανιχνεύουν πιθανές συγκρούσεις στον χρονοπρογραμματισμό και να προτείνουν την ιδανική σειρά εκτέλεσης των εργασιών.

1.1.4 Διαφορά με διαχείριση έργου

Συχνά η έννοια της διαχείρισης εργασιών (task management) συγχέεται με αυτή της διαχείρισης έργου (project management). Η αλήθεια είναι πως πρόκειται για έννοιες που όντως συσχετίζονται αλλά στην πραγματικότητα η καθεμία εστιάζει σε διαφορετικά αντικείμενα.

Η διαχείριση εργασιών, όπως έχει αναφέρθηκε, αφορά την παρακολούθηση διαφορετικών, μεμονομένων δραστηριοτήτων οι οποίες χρειάζεται να ολοκληρωθούν. Η διαχείριση εργασιών επικεντρώνεται στο μικροεπίπεδο, στη διαχείριση καθημερινών υποχρεώσεων, στα διαφορετικά deadlines που μπορεί να υπάρχουν, την εξέλιξή τους ανά το χρόνο κ.α. Τα εργαλεία που αφορούν τη διαχείριση εργασιών περιλαμβάνουν ημερολόγια, υπενθυμίσεις ή χρονοδιαγράμματα.

Αντίθετα, η διαχείριση έργου περιγράφει τον σχεδιασμό, την εκτέλεση και την ολοκλήρωση ενός ολόκληρου έργου. Ένα έργο αποτελείται και αυτό από διαφορετικές εργασίες, οι οποίες όμως είναι οργανωμένες προς έναν ευρύτερο στόχο. Επομένως, η έννοια της διαχείρισης έργου συμπεριλαμβάνει τη διαχείριση εργασιών, αλλά επίσης προϋποθέτει επιπλέον απαιτήσεις όπως τη σωστή κατανομή πόρων (resource allocation) ή την αξιολόγηση κινδύνου (risk assessment). Τα λογισμικά διαχείρισης έργου έχουν λειτουργικότητες όπως διαγράμματα Γκαντ ή παρακολούθηση εξαρτήσεων.

Στην παρούσα διπλωματική εργασία για λόγους πληρότητας θα αναλυθούν και κάποιες έννοιες που αφορούν τη διαχείριση έργου, έχοντας όμως υπόψην ότι η υλοποίηση της εφαρμογής αφορά τη διαχείριση εργασιών.

1.2 Ιστορική αναδρομή

Η διαχείριση και ο προγραμματισμός εργασιών αποτελούν έννοιες που υπήρχαν ήδη από την αρχαιότητα, πολύ πριν την ανάπτυξη ψηφιακών εργαλείων και αυτοματισμών, με τις πρώτες μορφές οργάνωσης να βασίζονται κυρίως στον προφορικό λόγο και την ανθρώπινη μνήμη. Με την ανάγκη για καταγραφή και παρακολούθηση, αναπτύχθηκαν και χρησιμοποιήθηκαν διάφορες μνημονικές συσκευές, οι οποίες προσέ-

φεραν στους πολιτισμούς της εποχής τρόπους οργάνωσης και αποθήκευσης κρίσιμων πληροφοριών.

Στην ενότητα αυτή, θα ασχοληθούμε με τις πρώτες καταγεγραμμένες μορφές διαχείρισης εργασιών, την εξέλιξή τους μέχρι σήμερα όπως επίσης και τις σημαντικές καινοτομίες και μεθοδολογίες που αναπτύχθηκαν στην πορεία της ιστορίας, όπως το διάγραμμα Γκαντ και άλλες οργανωτικές τεχνικές, οι οποίες εξέλιξαν τη διαχείριση και τον προγραμματισμό των εργασιών.

1.2.1 Προφορικότητα και μνημονικές συσκευές

Στην αρχαιότητα, η διαχείριση των εργασιών βασιζόταν κυρίως στον προφορικό λόγο, ο οποίος αποτελούσε το κύριο μέσο μετάδοσης πληροφοριών και οδηγιών. Έτσι οι εργασίες αναθέτονταν μέσω προφορικών οδηγιών, ενώ η ακρίβεια της εκτέλεσης εξαρτιόταν σε μεγάλο βαθμό από την αξιοπιστία της ανθρώπινης μνήμης. Αυτό το σύστημα, αν και ήταν η μόνη επιλογή που υπήρχε εκείνη την εποχή, παρουσίαζε σοβαρά μειονεκτήματα, καθώς η μνήμη είναι επιρρεπής σε λάθη, ειδικά σε καταστάσεις που απαιτούν τη διαχείριση πολλαπλών ή σύνθετων εργασιών. Με λίγα λόγια, η εξάρτηση από την ανθρώπινη μνήμη περιόριζε την ακρίβεια και τη δυνατότητα αποτελεσματικής οργάνωσης, ιδίως σε περιπτώσεις που οι εργασίες ήταν περίπλοκες, έπρεπε να εκτελεστούν σε μεγάλα χρονικά διαστήματα ή αφορούσαν πολλά άτομα. [13]

Αυτή η αναγκαιότητα οδήγησε στην ανάπτυξη τεχνικών απομνημόνευσης και συστημάτων καταγραφής, που είχαν ως στόχο τη βελτίωση της διαχείρισης πληροφοριών και την αύξηση της αξιοπιστίας. Τέτοιες τεχνικές περιλαμβαναν τη χρήση επαναλαμβανόμενων φράσεων και ρυθμικών μοτίβων για την ευκολότερη απομνημόνευση οδηγιών. Επιπλέον, η δημιουργία χειροκίνητων συσκευών, όπως το λουκάσα (lukasa) από τους Λούμπα του Κονγκό και το κουίπου (quipu) από τους Ίνκα, εισήγαγε συστήματα που υποκαθιστούσαν εν μέρει την ανθρώπινη μνήμη, παρέχοντας μια οπτικοποιημένη αναπαράσταση πληροφοριών. Αυτές οι συσκευές δεν ήταν απλώς εργαλεία καταγραφής, αλλά αποτελούσαν καινοτόμες λύσεις διαχείρισης εργασιών, ενισχύοντας την ικανότητα ανάκλησης και οργάνωσης πληροφοριών.

Το λουκάσα αποτελούνταν από πολύχρωμες χάντρες τοποθετημένες σε συγκεκριμένες θέσεις πάνω σε ξύλινες ή δερμάτινες επιφάνειες, προσφέροντας στους χειριστές έναν τρόπο να αποθηκεύουν, να οργανώνουν και να ανακαλούν πληροφορίες. [27] Το κουίπου ήταν μια κατασκευή με χορδές από βαμβάκι ή μαλλί. Οι χορδές ήταν πολύχρωμες με κόμπους, επιτρέποντας έτσι την κατηγοριοποίηση και αποθήκευση πληροφοριών βάσει χρώματος, διάταξης και αριθμού. Οι Ίνκα δημιουργούσαν κόμπους στις χορδές και τις χρησιμοποιούσαν για τη συλλογή και παρακολούθηση των υποχρεώσεων τους ή και για την αποθήκευση άλλων πληροφοριών όπως δεδομένα απογραφής, φορολογικών υποχρεώσεων και άλλα. [25]

Η δημιουργία τέτοιων τεχνικών και συστημάτων καταγραφής αναδεικνύει την ανθρώπινη ικανότητα να προσαρμόζεται σε πρακτικές ανάγκες και να δημιουργεί και-



Σχήμα 1.1: Η συσκευή λουκάσα



Σχήμα 1.2: Η συσκευή κουίπου

νοτόμες λύσεις. Αυτά τα πρώιμα μέσα διαχείρισης εργασιών όχι μόνο κάλυψαν τις απαιτήσεις της εποχής, αλλά έθεσαν τα θεμέλια για τη μεταγενέστερη ανάπτυξη γραπτών και, τελικά, φηφιακών συστημάτων, που επανακαθόρισαν τον τρόπο με τον οποίο οργανώνουμε και εκτελούμε εργασίες στις μέρες μας.

1.2.2 Πρώτα ημερολόγια

Κατασκευές όπως τα ηλιακά ρολόγια αποτέλεσαν ένα από τα πρώτα εργαλεία που έδωσαν στους ανθρώπους τη δυνατότητα να διαιρέσουν την ημέρα σε διαχριτά τμήματα. Η ανακάλυψη αυτών των εργαλείων αποτέλεσε καθοριστικό βήμα στην κατανόηση του χρόνου ως δομημένου και μετρήσιμου πόρου, επιτρέποντας στους πληθυσμούς να οργανώσουν καλύτερα τις καθημερινές τους δραστηριότητες. Η δυνατότητα αυτή οδήγησε σε μια σαφή διάκριση μεταξύ υποχρεώσεων και ελεύθερου χρόνου, καθώς οι

άνθρωποι άρχισαν να προγραμματίζουν τις ώρες της ημέρας με μεγαλύτερη ακρίβεια. Αυτός ο διαχωρισμός ήταν θεμελιώδης για την ανάπτυξη πιο σύνθετων συστημάτων διαχείρισης του χρόνου, καθώς οι κοινότητες αντιλήφθηκαν τη σημασία του χρονοπρογραμματισμού για τη βελτιστοποίηση των συλλογικών τους δραστηριοτήτων.



Σχήμα 1.3: Το παλαιότερο γνωστό ηλιακό ρολόι από τους Αιγυπτίους· χρησιμοποιούνταν για να μετράει τις ώρες εργασίας τους

Παράλληλα με τα ηλιακά ρολόγια, οι πρώιμες προσπάθειες δημιουργίας ημερολογίων συνέβαλαν καθοριστικά στην εξέλιξη της διαχείρισης εργασιών και χρόνου. Πολιτισμοί όπως οι Αιγύπτιοι, οι Ρωμαίοι και οι Μάγια ανέπτυξαν πολύπλοκα συστήματα ημερολογίων που βασίζονταν στις κινήσεις του ήλιου, της σελήνης και των άστρων. Αυτά τα ημερολόγια όχι μόνο προσδιόριζαν τον χρόνο για γεωργικές δραστηριότητες, όπως η σπορά και η συγκομιδή, αλλά χρησίμευαν και ως οδηγός για θρησκευτικές τελετές, κοινωνικές εκδηλώσεις και άλλες τελετουργίες. Μέσω αυτών των εργαλείων, έγινε εφικτός ο διαχωρισμός του χρόνου, προσφέροντας μια πρώιμη μορφή συστηματικής οργάνωσης που συνέβαλε στην ανάπτυξη των κοινωνιών. [28]

Η τεχνολογική πρόοδος έφερε σημαντικές εξελίξεις στον τρόπο μέτρησης και διαχείρισης του χρόνου. Τα ηλιακά ρολόγια, τα οποία ήταν εξαρτώμενα από την παρουσία του ήλιου, σταδιακά εξελίχθηκαν σε μηχανικά ρολόγια, τα οποία μπορούσαν να λειτουργούν ανεξάρτητα από τις καιρικές συνθήκες ή την ώρα της ημέρας. Αυτή η μετάβαση στα μηχανικά ρολόγια σηματοδότησε μια νέα εποχή για τον χρονοπρογραμματισμό. Τα μηχανικά ρολόγια προσέφεραν μεγαλύτερη ακρίβεια και έθεσαν τα θεμέλια για την ανάπτυξη πιο σύνθετων εργαλείων διαχείρισης εργασιών, που θα μπορούσαν να εξυπηρετήσουν τις αυξανόμενες απαιτήσεις των κοινωνιών.

1.2.3 Σύγχρονη εποχή

Η οργανωμένη διαχείριση εργασιών έχει τις ρίζες της βαθιά μέσα στην ιστορία, καθώς οι άνθρωποι πάντα αναζητούσαν τρόπους να οργανώσουν καλύτερα τις δραστηριότητές τους. Ωστόσο, οι πρώτες προσπάθειες τυποποίησης αυτής της διαδικασίας εντοπίζονται στον 18ο αιώνα, όταν η ανάγκη για μια συστηματική προσέγγιση έγινε πιο έντονη λόγω της ανάπτυξης των κοινωνιών και της αυξανόμενης πολυπλοκότητας των έργων. Στα τέλη του 19ου αιώνα, η βιομηχανική επανάσταση επέφερε τεράστιες αλλαγές στον τρόπο παραγωγής και κατασκευής. Τα έργα μεγάλης κλίμακας, όπως σιδηροδρομικά δίκτυα, γέφυρες και εργοστάσια, απαιτούσαν πιο οργανωμένες προσεγγίσεις στη διαχείριση ανθρώπινου δυναμικού και πόρων. Αυτές οι νέες απαιτήσεις οδήγησαν στην ανάγκη για πιο λεπτομερή και αποτελεσματική διαχείριση των εργασιών. Όμως η οργάνωση χιλιάδων εργατών, η διαχείριση μεγάλων ποσοτήτων πρώτων υλών και η τήρηση αυστηρών χρονοδιαγραμμάτων ήταν προκλήσεις που δε θα μπορούσαν να αντιμετωπιστούν με τις παραδοσιακές τεχνικές.

Μηχανικοί όπως ο Henry Gantt εισήγαγαν πρωτοποριακές μεθόδους οργάνωσης, όπως το **διάγραμμα Γκαντ**. Το διάγραμμα Γκαντ είναι ένα εργαλείο που παρέχει οπτικοποίηση, αναπαριστώντας όλες τις επιμέρους εργασίες ενός έργου κατά μήκος ενός χρονικού άξονα, παρέχοντας μια καθαρή εικόνα των φάσεων υλοποίησής του. Με αυτόν τον τρόπο, οι υπεύθυνοι έργων μπορούσαν να παρακολουθούν την πρόοδο κάθε φάσης, να εντοπίζουν πιθανές καθυστερήσεις και να αναπροσαρμόζουν τον προγραμματισμό όπου ήταν απαραίτητο. Ένα από τα μεγαλύτερα πλεονεκτήματα του διαγράμματος Γκαντ ήταν η δυνατότητα προσδιορισμού της κρίσιμης διαδρομής του έργου, δηλαδή της αλληλουχίας των εργασιών που πρέπει να ολοκληρωθούν εντός συγκεκριμένων χρονικών ορίων για να διασφαλιστεί η έγκαιρη ολοκλήρωση του έργου. Θα αναφερθούμε με μεγαλύτερη λεπτομέρεια στο διάγραμμα Γκαντ στην ενότητα 1.4. Πάντως, το εργαλείο αυτό ήταν καθοριστικό σε μεγάλα έργα υποδομών, όπως η κατασκευή της Διώρυγας του Παναμά, που αποτέλεσε ένα από τα πιο φιλόδοξα και απαιτητικά έργα της εποχής, καθώς και το φράγμα Χούβερ, το οποίο απαίτησε σχολαστικό σχεδιασμό και συντονισμό πόρων σε πρωτόγνωρη κλίμακα. [15]

Η εισαγωγή μεθοδολογικών εργαλείων όπως το διάγραμμα Γκαντ δεν περιορίστηκε μόνο στη βιομηχανία και τα έργα υποδομών, αποτέλεσε την έμπνευση για νέες έρευνες και πρακτικές που επεκτάθηκαν σε διάφορους τομείς. Ένα από τα πιο χαρακτηριστικά παραδείγματα είναι το Πρότζεκτ Μανχάταν (Manhattan Project), το οποίο σχεδιάστηκε κατά τη διάρκεια του Β' Παγκοσμίου Πολέμου για το σχεδιασμό πυρηνικών όπλων. Αυτό το ιδιαίτερα απαιτητικό έργο οδήγησε στην ανάπτυξη δύο νέων μοντέλων διαχείρισης, του PERT (Program Evaluation and Review Technique) και του CPM (Critical Path Method). Το PERT σχεδιάστηκε για να αντιμετωπίσει την αβεβαιότητα στις εκτιμήσεις του χρόνου υλοποίησης των εργασιών, ενώ το CPM επικεντρώθηκε στην ανάλυση και τη βελτιστοποίηση της κρίσιμης διαδρομής του έργου. [6]. Θα αναφερθούμε και σε αυτά τα μοντέλα στην ενότητα 1.4.



Σχήμα 1.4: Το φράγμα Χούβερ [16]

1.3 Η συνδρομή της τεχνολογίας

Από τη δεκαετία του '60 και έπειτα, οι επιχειρήσεις άρχισαν να αναγνωρίζουν την αξία της συστηματικής και μεθοδικής οργάνωσης της εργασίας. Η ψηφιακή επανάσταση που ακολούθησε δε θα μπορούσε παρά να γιγαντώσει αυτή τη νέα πραγματικότητα. Η είσοδο των υπολογιστών, επέφερε και νέες δυνατότητες αποθήκευσης και ανάλυσης δεδομένων, δυνατότητες που άλλαξαν ριζικά τη διαχείριση των εργασιών. Έτσι, διαδικασίες που προηγουμένως απαιτούσαν χρονοβόρα χειρωνακτική εργασία και εκτεταμένη χρήση χαρτιού, έγιναν πλέον πιο γρήγορες και πιο ακριβείς.

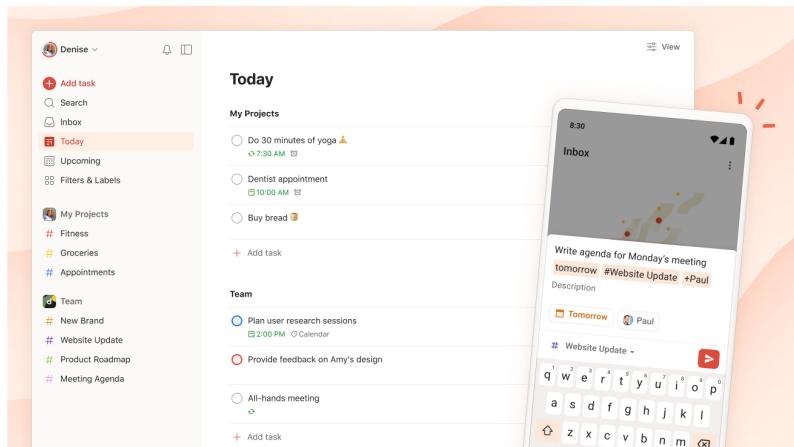
Η τεχνολογική πρόοδος έφερε νέα εργαλεία και λογισμικά που ενίσχυσαν τη συνεργασία μεταξύ ομάδων και τμημάτων, όπως το Microsoft Project και αργότερα οι πλατφόρμες συνεργασίας τύπου Trello και Asana, επέτρεψαν σε ομάδες διαφορετικών γεωγραφικών περιοχών να συνεργάζονται απρόσκοπτα, μειώνοντας τα εμπόδια επικοινωνίας, ενώ πρόσθισε και αυτοματισμούς στην κατανομή εργασιών και στη δημιουργία χρονοδιαγραμμάτων. Η τεχνολογία όχι μόνο βελτίωσε τη λειτουργικότητα των εργαλείων διαχείρισης αλλά τα έκανε επίσης πιο προσιτά σε μικρότερες επιχειρήσεις, που προηγουμένως δεν είχαν τη δυνατότητα να επενδύσουν σε τέτοιες λύσεις.

1.3.1 Ψηφιακά εργαλεία

Με την εξέλιξη της τεχνολογίας, οι σημειώσεις και η οργάνωση μεταφέρθηκε από τις χειρόγραφες σημειώσεις, τα ημερολόγια και τα έγγραφα των γραφομηχανών σε ψηφιακά εργαλεία. Παραδείγματα αυτών σε ατομικό επίπεδο είναι το Todoist ή το Notion και σε επαγγελματικό επίπεδο προγράμματα σαν το Microsoft Project.

1.3.1.1 Todoist

Αν και έχει χάσει πλέον την πρωτοκαθεδρία του, το Todoist [36] παραμένει μια από τις πιο δημοφιλείς εφαρμογές διαχείρισης εργασιών, σχεδιασμένη για να βοηθάει τόσο απλούς χρήστες όσο και επαγγελματίες να οργανώνουν τις υποχρεώσεις τους και να βελτιώνουν την παραγωγικότητά τους. Η εφαρμογή επιτρέπει τη δημιουργία λιστών εργασιών με δυνατότητα ομαδοποίησης σε έργα, υποέργα ή ετικέτες (tags). Οι χρήστες μπορούν να καθορίσουν ημερομηνίες και προθεσμίες καθώς και να ρυθμίσουν υπενθυμίσεις, καταφέροντας έτσι την αποδοτικότερη οργάνωση του χρόνου τους, ενώ οι ειδοποιήσεις τους διασφαλίζουν ότι δε θα παραλείψουν καμία σημαντική εργασία. Επιπλέον, περιλαμβάνει σύστημα επιβράβευσης (“Karma”) ενθαρρύνοντας τη συνέπεια και την ολοκλήρωση των εργασιών, ενώ υπάρχει η δυνατότητα ενσωμάτωσης με εξωτερικές εφαρμογές όπως το Google Calendar.



Σχήμα 1.5: Η εφαρμογή Todoist.

1.3.1.2 Notion

Το Notion [41] είναι αυτή τη στιγμή ίσως η πιο δημοφιλής πλατφόρμα προσωπικής οργάνωσης. Συνδυάζει στοιχεία για task-management, note-taking, βάσεις δεδομένων και συνεργατικότητας σε μία ενιαία εφαρμογή, καθιστώντας το ιδανικό για χρήστες που επιζητούν έναν κεντρικό χώρο για τη διαχείριση της εργασιακής ή προσωπικής τους ζωής. Το κύριο χαρακτηριστικό του Notion είναι η δυνατότητα δημιουργίας προσαρμοσμένων σελίδων χρησιμοποιώντας Markdown γλώσσα, στις οποίες οι χρήστες μπορούν να μορφοποιήσουν το κείμενο, να προσθέσουν πίνακες ή να ενσωματώσουν διάφορα στοιχεία όπως λίστες εργασιών, πίνακες Kanban, ημερολόγια, checklists, ή ακόμη και embedded αρχεία. Αυτή η ευελιξία επιτρέπει τη δημιουργία εξατομικευμένων συστημάτων διαχείρισης προσαρμοσμένων στις μοναδικές ανάγκες του κάθε χρήστη, λειτουργώντας ως μια προσωπική εγκυροπαίδεια.

Επίσης, υπάρχει η δυνατότητα δημιουργίας βάσεων δεδομένων (οι οποίες μπορούν

The screenshot shows the Notion Projects interface. At the top, there are navigation icons (dots, arrows) and a search bar labeled 'Projects'. Below the header are tabs for 'Active' (selected), 'Timeline', 'Mine', and '2 more...'. To the right are buttons for 'Filter', 'Sort', 'Search', and a 'New' dropdown. A search bar and a 'New' button are also present. The main area displays two projects under the 'Planning' status: 'New Emojis dont render' and 'Better commenting experience'. Under the 'In Progress' status, there is one project: 'Dogfood new mobile experience'. On the right, a sidebar shows 'Hidden groups': 'Paused' (1), 'Backlog' (0), 'Done' (4), and 'Canceled' (0). A 'New' button is located at the bottom left of the main project list.

Σχήμα 1.6: Στιγμιότυπο του Notion

να λειτουργήσουν ως λίστες εργασιών, παρακολούθησης προόδου για έργα κ.α.) τις οποίες μπορούν να φιλτράρουν, να ταξινομούν και να χειρίζονται όπως θέλουν. Τέλος, υπάρχει δυνατότητα για συνεργατική κοινή χρήση σελίδων, την ενσωμάτωση με άλλα εργαλεία όπως το Google Drive ή το Slack.

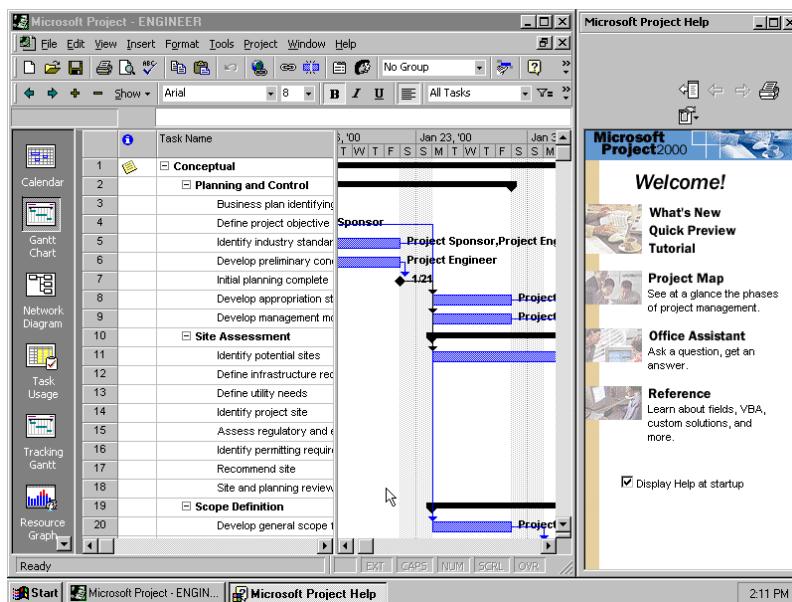
1.3.1.3 Microsoft Project



Σχήμα 1.7: Στιγμιότυπο από το Microsoft Project 3.0 (σε DOS) [40]

Πρόκειται για ένα από τα πρώτα λογισμικά διαχείρισης έργων, σχεδιασμένα για το κοινό. Η ιδέα για τη δημιουργία του προήλθε από μια φάρσα του Ron Bredehoeft,

ο οποίος, θέλοντας να αναπαραστήσει τη διαδικασία παρασκευής αυγών μπένεντικτ σε όρους διαχείρισης έργων, ανέπτυξε την ιδέα για ένα εργαλείο που θα μπορούσε να χρησιμοποιηθεί για την οργάνωση και τον προγραμματισμό οποιουδήποτε έργου. Το Microsoft Project παρουσιάστηκε για πρώτη φορά το 1984 ως εφαρμογή για DOS, κερδίζοντας αμέσως την προσοχή των επαγγελματιών. Σήμερα, αποτελεί ένα από τα πιο καθιερωμένα εργαλεία για την οργάνωση, τον χρονοπρογραμματισμό και την παρακολούθηση έργων, με εφαρμογές σε πλήθος βιομηχανιών, από την κατασκευή μέχρι την πληροφορική και τη διαχείριση ανθρώπινων πόρων.



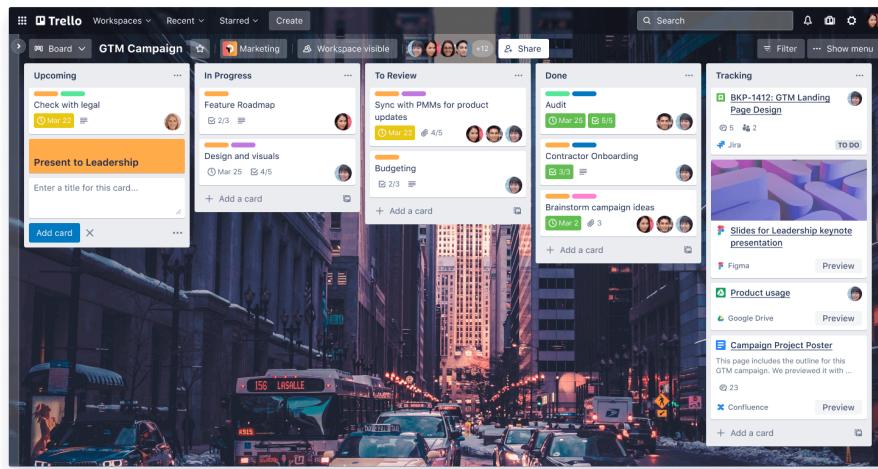
Σχήμα 1.8: Στιγμιότυπο από το Microsoft Project 2000 [40]

Η κεντρική οθόνη του λογισμικού χωρίζεται σε δύο βασικές περιοχές: το διάγραμμα Γκαντ (Gantt chart) και τον πίνακα εισαγωγής εργασιών (input table). Ο πίνακας εισαγωγής επιτρέπει την εισαγωγή λεπτομερών πληροφοριών σχετικά με κάθε εργασία, όπως η διάρκεια, οι εξαρτήσεις και οι πόροι.

Παρέχονται λειτουργικότητες όπως η δυνατότητα ιεράρχησης των εργασιών μέσω της τοποθέτησης εσοχών (indents) (δημιουργώντας μιας σαφούς δομής του έργου, διευκολύνοντας τη διαχείριση πολύπλοκων έργων με πολλές υποκατηγορίες), η δημιουργία εξαρτήσεων μεταξύ των εργασιών (predecessors) (για παράδειγμα, μια εργασία μπορεί να προγραμματιστεί να ξεκινήσει μόνο όταν ολοκληρωθεί μια άλλη), η δυνατότητα αυτόματου προγραμματισμού των εργασιών (auto-scheduling), η δυνατότητα δημιουργίας αλυσιδωτών εργασιών (linked tasks), στις οποίες μια εργασία εκτελείται αμέσως μετά την ολοκλήρωση μιας άλλης, διευκολύνοντας τον προγραμματισμό μεγάλων και σύνθετων έργων. Επιπλέον, το λογισμικό προσφέρει ευελιξία στην προβολή των δεδομένων, επιτρέποντας την αποτύπωση των εργασιών πέρα από το διάγραμμα Γκαντ σε μορφή ημερολογίου, φύλλου εργασίας (task sheet) ή ακόμη και

η δημιουργία στατιστικών αναφορών. Έτσι, για παράδειγμα μια ομάδα μπορεί να επιλέξει το ημερολόγιο για να βλέπει τις ημερήσιες υποχρεώσεις της, ενώ ένας διευθυντής μπορεί να επιλέξει τις στατιστικές αναφορές για να αξιολογήσει τη συνολική πρόοδο.

1.3.1.4 Trello



Σχήμα 1.9: Στιγμιότυπο του Trello

Το Trello [20] είναι μια πλατφόρμα διαχείρισης εργασιών που βασίζεται στους πίνακες Kanban (θα αναλυθούν στην ενότητα 1.4), επιτρέποντας μια εύληπτη οργάνωση της καθημερινότητας. Με τη χρήση πινάκων, λιστών και καρτών, οι χρήστες μπορούν συνεργατικά να παρακολουθούν την πρόοδο των εργασιών τους, ιδιαίτερα για εργασίες που χρειάζονται ομαδική συνεργασία, καθιστώντας το ιδιαίτερα δημοφιλές σε ομάδες όλων των μεγεθών και σε διάφορους τομείς. Κάθε πίνακας αντιπροσωπεύει ένα πρότζεκτ, ενώ οι λίστες μπορούν να χρησιμοποιηθούν για την κατηγοριοποίηση των εργασιών σε στάδια (“To Do”, “In Progress”, “Done”). Οι κάρτες, που τοποθετούνται μέσα στις λίστες, αντιπροσωπεύουν συγκεκριμένες εργασίες που πρέπει να ολοκληρωθούν. Οι χρήστες μπορούν να προσθέτουν περιγραφές, checklists, προθεσμίες, συνημμένα αρχεία, και ετικέτες (labels) στις κάρτες, επιτρέποντας την εξατομίκευση και την οργάνωση κάθε εργασίας σύμφωνα με τις ανάγκες τους. Προσφέρονται εργαλεία αυτοματοποίησης (λειτουργία “Butler”) και υπάρχουν δυνατότητες ενσωμάτωσης με άλλες εφαρμογές.

1.4 Μεθοδολογίες

Στις προηγούμενες ενότητες, αναφερθήκαμε στην ιστορική εξέλιξη της οργάνωσης των εργασιών και του χρόνου μας και το πως η ραγδαία πρόοδος της τεχνολογίας

έθεσαν τα θεμέλια για τις σύγχρονες προσεγγίσεις που ακολουθούμε σήμερα στη διαχείριση εργασιών. Επίσης, έγινε αναφορά σε φηφιακά εργαλεία, όπως το Notion, το Trello και το Todoist, και το πώς παρέχουν λειτουργικότητα που διευκολύνει τη ροή των εργασιών και ενισχύει την παραγωγικότητα των ομάδων.

Παράλληλα με την ανάπτυξη των εργαλείων, ωστόσο, αναπτύχθηκαν και υιοθετήθηκαν και αντίστοιχες μεθοδολογίες, οι οποίες παρέχουν βασικές αρχές για την αποτελεσματική διαχείριση σύνθετων έργων. Παραδείγματα αυτών των μεθοδολογιών που θα αναλυθούν είναι το διάγραμμα Γκαντ, το PERT και το Kanban.

Στο πλαίσιο της παρούσας διπλωματικής, η οποία επικεντρώνεται στην ανάπτυξη μιας εφαρμογής διαχείρισης εργασιών που θα περιλαμβάνει και πίνακα Kanban, η αναφορά στις μεθοδολογίες αυτές είναι απαραίτητη για λόγους πληρότητας ώστε να αναδειχθούν οι αρχές που καθοδηγούν τον σχεδιασμό τέτοιων εργαλείων.

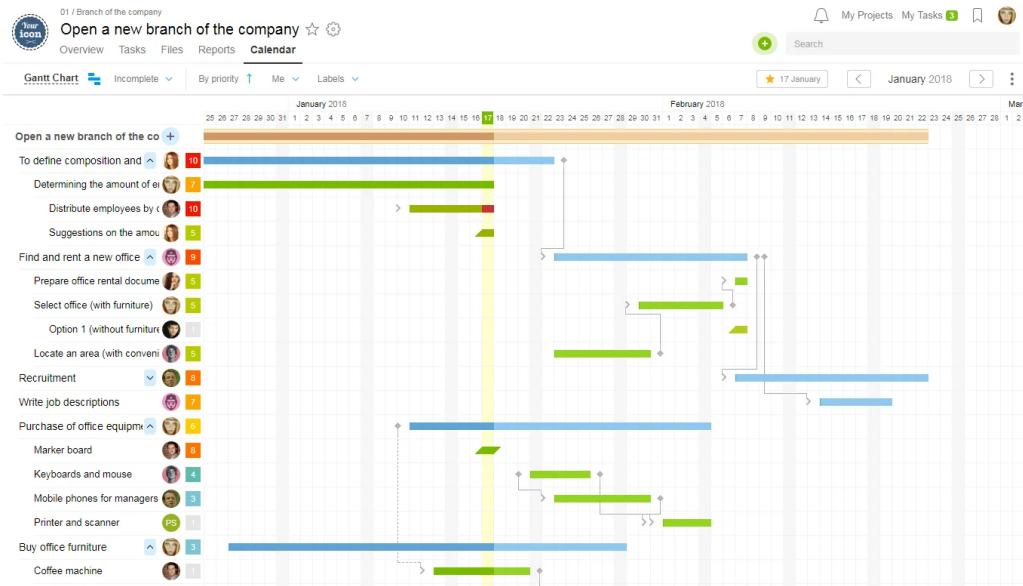
1.4.1 Διάγραμμα Γκαντ

Το διάγραμμα Γκαντ (Gantt chart) έχει καθιερωθεί ως ένα από τα πιο χρήσιμα εργαλεία στη διαχείριση έργων, καθώς παρέχει μια εύληπτη γραφική αναπαράσταση των επιμέρους εργασιών ενός έργου. Στον οριζόντιο άξονα απεικονίζεται ο χρόνος, ενώ στον κατακόρυφο άξονα παρατίθενται οι διαφορετικές εργασίες που συνθέτουν το έργο. Για τη δημιουργία ενός διαγράμματος Γκαντ, είναι απαραίτητος ο αρχικός καταμερισμός του έργου σε επιμέρους εργασίες. Αυτό περιλαμβάνει την αναλυτική καταγραφή κάθε δραστηριότητας που πρέπει να ολοκληρωθεί και την εκτίμηση της χρονικής διάρκειας που θα απαιτηθεί για την ολοκλήρωσή της. Αφού γίνει ο καταμερισμός, οι εργασίες τοποθετούνται στο διάγραμμα με τέτοιο τρόπο ώστε αυτές που ολοκληρώνονται νωρίτερα να βρίσκονται φηλότερα, διατηρώντας μια σαφή δομή που διευκολύνει την ανάγνωση και την κατανόηση του χρονοδιαγράμματος.

Η ευκολία και η ταχύτητα με την οποία μπορεί να κατασκευαστεί ένα διάγραμμα Γκαντ αποτελούν έναν από τους κύριους λόγους για τη δημοτικότητά του. Παρέχει μια σαφή απεικόνιση της χρονικής διάρκειας και της αλληλουχίας των εργασιών, κάνοντας τη χρήση του προσιτή ακόμη και για άτομα που δεν έχουν εξειδικευμένες γνώσεις στη διαχείριση έργων.

Παρόλα αυτά, το διάγραμμα Γκαντ έχει και ορισμένους περιορισμούς. Ένας από αυτούς είναι η αδυναμία του να αποτυπώσει τις εξαρτήσεις μεταξύ των εργασιών και την επίδραση της καθυστέρισης μιας εργασίας στο συνολικό έργο. Για παράδειγμα, δεν είναι εμφανές ποιες εργασίες πρέπει να ολοκληρωθούν πριν ξεκινήσει μια άλλη, κάτι που μπορεί να οδηγήσει σε παρανοήσεις ή καθυστερήσεις αν δεν υπάρχει κατάλληλος συντονισμός. Επιπλέον, η στατική του φύση περιορίζει τη δυνατότητα αναπροσαρμογής όταν οι συνθήκες αλλάξουν, όπως σε περιπτώσεις μεταβολής της διάρκειας μιας εργασίας ή προσθήκης νέων δραστηριοτήτων. Αυτό σημαίνει ότι, ενώ είναι εξαιρετικό για την αρχική φάση σχεδιασμού και την παρακολούθηση, μπορεί να μην επαρκεί σε δυναμικά περιβάλλοντα όπου απαιτείται συνεχής αναπροσαρμογή.

Παρόλα αυτά, παραμένει ένα από τα πιο χρήσιμα εργαλεία για την κατανόηση της χρονικής διάστασης ενός έργου και τη συνολική εποπτεία της προόδου του. [42]



Σχήμα 1.10: Παράδειγμα διαγράμματος Γκαντ

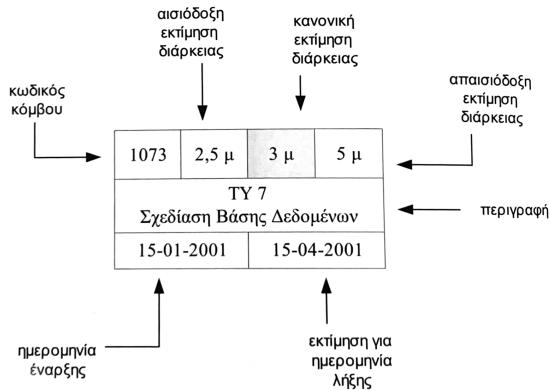
1.4.2 Program evaluation and review technique (PERT)

Το **Program Evaluation and Review Technique (PERT)** συνδυαστικά και με τη μέθοδο ακίσιμης διαδρομής (Critical Path Method – CPM), είναι μια μεθοδολογία προγραμματισμού και ελέγχου έργων που επικεντρώνεται στον υπολογισμό και την αξιολόγηση του χρόνου ολοκλήρωσης ενός έργου, ενώ παράλληλα παρέχει σαφή εικόνα των σχέσεων και των εξαρτήσεων μεταξύ των δραστηριοτήτων του. Αναπτύχθηκε τη δεκαετία του 1950 για την υποστήριξη σύνθετων έργων με υψηλή αβεβαιότητα, όπως ο προγραμματισμός στρατιωτικών προγραμμάτων.

Σε ένα διάγραμμα PERT (διάγραμμα αξιολόγησης έργου), το έργο αναλύεται σε επιμέρους δραστηριότητες, καθεμία από τις οποίες απεικονίζεται ως κόμβος σε ένα γράφημα. Αυτή η δομή επιτρέπει την οπτικοποίηση των σχέσεων και των εξαρτήσεων μεταξύ των δραστηριοτήτων, καθιστώντας σαφές ποιες πρέπει να ολοκληρωθούν πρώτα και ποιες μπορούν να εκτελούνται παράλληλα.

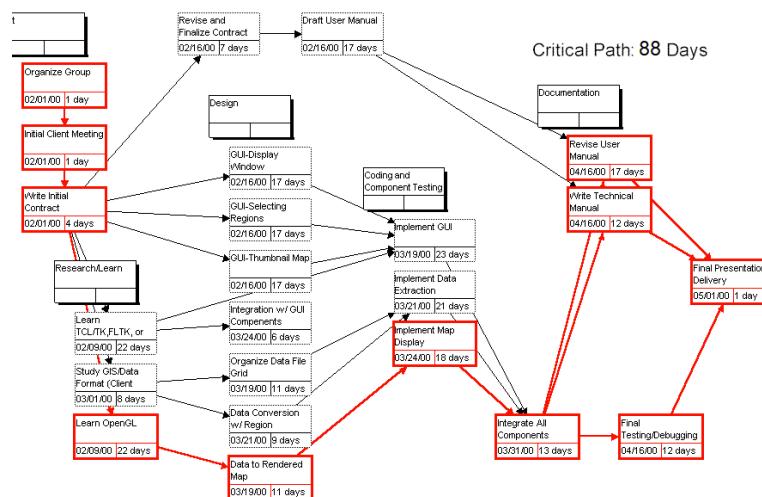
Το PERT βασίζεται στην εκτίμηση του χρόνου για κάθε δραστηριότητα χρησιμοποιώντας τρεις παραμέτρους: τον αισιόδοξο χρόνο (ο ελάχιστος χρόνος ολοκλήρωσης), τον πιο πιθανό χρόνο και τον απαισιόδοξο χρόνο (ο μέγιστος χρόνος ολοκλήρωσης). Με βάση αυτές τις εκτιμήσεις, υπολογίζεται ο αναμενόμενος χρόνος για κάθε δραστηριότητα, λαμβάνοντας υπόψη την αβεβαιότητα.

Μια σημαντική έννοια στο PERT είναι ο υπολογισμός της ακίσιμης διαδρομής, δηλαδή της αλληλουχίας δραστηριοτήτων που καθορίζει τη συνολική διάρκεια του



Σχήμα 1.11: Παράδειγμα κόμβου σε διάγραμμα PERT [42]

έργου. Οι δραστηριότητες που βρίσκονται στην κρίσιμη διαδρομή δεν επιτρέπουν καθυστερήσεις, καθώς οποιαδήποτε καθυστέρηση σε αυτές θα παρατείνει το συνολικό χρονοδιάγραμμα του έργου. Αντίθετα, οι μη κρίσιμες δραστηριότητες έχουν ένα χρονικό περιθώριο (slack), το οποίο τους επιτρέπει κάποιες καθυστερήσεις χωρίς να επηρεαστεί το συνολικό έργο.

Σχήμα 1.12: Παράδειγμα διαγράμματος PERT
(με κόκκινο εμφανίζεται η κρίσιμη διαδρομή)

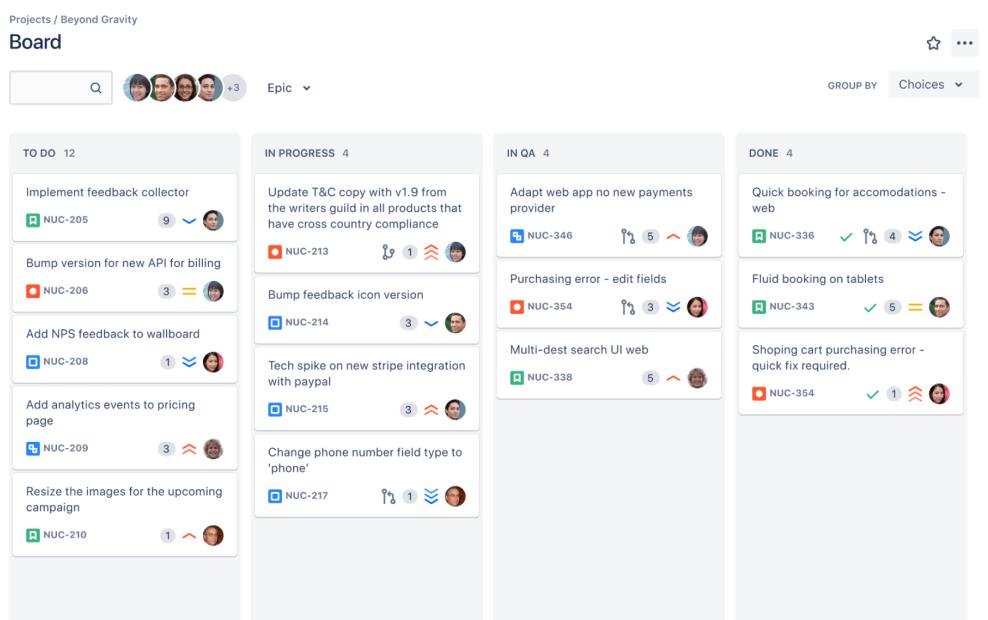
Η μεθοδολογία PERT παρέχει σημαντικά εργαλεία για την ανάλυση του χρόνου ολοκλήρωσης του έργου, όπως ο υπολογισμός του νωρίτερου και του αργότερου χρόνου για κάθε γεγονός και δραστηριότητα. Με τη βοήθεια αυτών των υπολογισμών, οι διαχειριστές έργων μπορούν να προβλέψουν την ελάχιστη και μέγιστη διάρκεια του έργου, να αναγνωρίσουν κρίσιμα σημεία και να σχεδιάσουν κατάλληλα τις ενέργειες τους για την αντιμετώπιση πιθανών καθυστερήσεων.

Το PERT, παρόλο που είναι ιδιαίτερα χρήσιμο σε έργα με πολλές αβεβαιότητες, παρουσιάζει ορισμένους περιορισμούς. Οι εκτιμήσεις χρόνου συχνά βασίζονται σε υποκειμενικά δεδομένα, γεγονός που μπορεί να οδηγήσει σε αποκλίσεις. Παρόλα αυτά, η εφαρμογή του PERT συμβάλλει σημαντικά στη διαχείριση έργων, διευκολύνοντας τη λήψη αποφάσεων και την κατανομή πόρων με τρόπο αποτελεσματικό. [19]

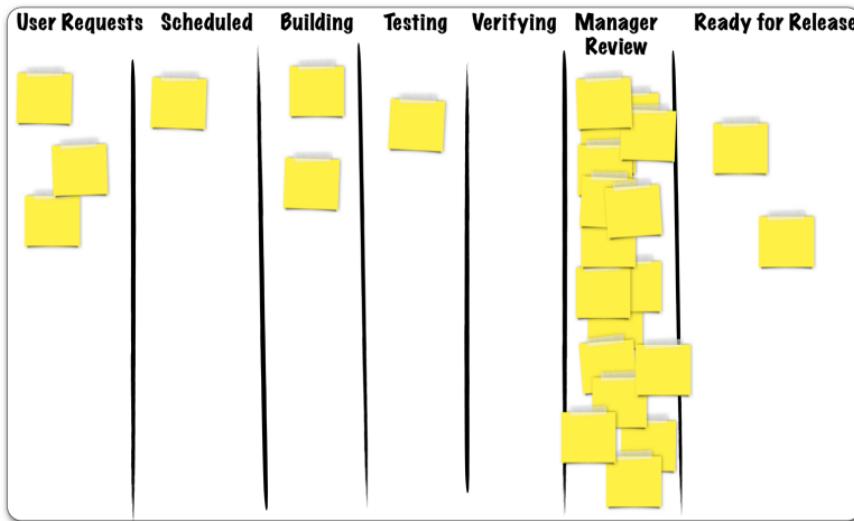
1.4.3 Kanban

Το Kanban είναι μια μέθοδος διαχείρισης εργασιών που αρχικά αναπτύχθηκε από την Toyota στον τομέα της παραγωγής τη δεκαετία του 1940, με σκοπό τη βελτίωση της ροής εργασιών και την ελαχιστοποίηση των καθυστερήσεων και στη συνέχεια εξελίχθηκε και υιοθετήθηκε ευρέως στον τομέα της ανάπτυξης λογισμικού και άλλων έργων, για να βελτιώσει την αποτελεσματικότητα των ομάδων και την παράδοση των έργων.

Η κεντρική ιδέα του Kanban είναι η οπτικοποίηση των εργασιών μέσω πίνακα (Kanban board). Στον πίνακα, οι εργασίες αναπαρίστανται ως κάρτες που μετακινούνται μεταξύ διαφορετικών σταδίων, όπως “To Do”, “In Progress” και “Done”. Αυτή η οπτική αναπαράσταση της διαδικασίας επιτρέπει στα μέλη της ομάδας να παρακολουθούν την πρόοδο των εργασιών σε πραγματικό χρόνο και να εντοπίζουν εύκολα τα τμήματα του έργου που καθυστερούν ή χρειάζονται προσοχή. Στη σύγχρονη πρακτική, τα Kanban boards συνήθως υποστηρίζονται από φημιακά εργαλεία όπως το Trello [20], το Jira [3] και το Asana [2], τα οποία προσφέρουν επιπλέον δυνατότητες όπως η αυτόματη ενημέρωση και η συνεργασία σε πραγματικό χρόνο.



Σχήμα 1.13: Ένας Kanban πίνακας στο Jira



Σχήμα 1.14: Ένα παράδειγμα συμφορήσεων [34]

Η βασική αρχή του Kanban είναι ο περιορισμός της εργασίας που βρίσκεται σε εξέλιξη (“Work In Progress, WIP”). Έτσι οι ομάδες επικεντρώνονται σε συγκεκριμένες εργασίες και τις ολοκληρώνουν προτού ξεκινήσουν μια νέα. Με αυτόν τον τρόπο, το Kanban διασφαλίζει ότι η ομάδα δεν αναλαμβάνει περισσότερες εργασίες από όσες μπορεί να διαχειριστεί, ενισχύοντας τη ροή της εργασίας και μειώνοντας τις καθυστερήσεις. Επιπλέον, μπορούν εύκολα να εντοπίζουν συμφορήσεις (bottlenecks) στη ροή εργασιών. Οι συμφορήσεις αυτές προκύπτουν όταν μια συγκεκριμένη φάση ή εργασία καθυστερεί και εμποδίζει την πρόοδο των υπόλοιπων εργασιών.

Η εφαρμογή του Kanban έχει αποδειχθεί αποτελεσματική στην αύξηση της παραγωγικότητας και της ποιότητας των έργων, καθώς ενθαρρύνει τη συνεχιζόμενη βελτίωση και την ανάλυση της ροής εργασίας. Με την εφαρμογή της μεθόδου, οι ομάδες μπορούν να παρακολουθούν τις καθυστερήσεις, να μειώσουν τις χρόνιες καθυστερήσεις και να βελτιώσουν τις επικοινωνιακές ροές, ενισχύοντας τη συνεργασία και τη διαφάνεια. [1] [34]

1.5 Διαχείριση εργασιών στο πανεπιστήμιο

Σε πολλές περιπτώσεις, η αποτελεσματικότητα των φοιτητών καθορίζεται χυρίως από την οργάνωσή τους και τον σωστό προγραμματισμό τους, τόσο στις ακαδημαϊκές όσο και στις προσωπικές τους υποχρεώσεις. Η σωστή διαχείριση χρόνου και προτεραιοτήτων είναι καθοριστική για την αποφυγή άγχους, την αύξηση της παραγωγικότητας και την επίτευξη των στόχων τους. Παρόλα αυτά, οι φοιτητές συχνά έρχονται αντιμέτωποι με προκλήσεις, όπως η αναβλητικότητα και η έλλειψη σωστής οργάνωσης για την ολοκλήρωση των καθημερινών τους καθηγόντων. Οι ερευνητι-

κές προσπάθειες στον τομέα αυτό αναδεικνύουν τα εμπόδια που αντιμετωπίζουν οι φοιτητές και προσφέρουν πολύτιμες πληροφορίες για τη βελτίωση των δεξιοτήτων διαχείρισης εργασιών.

1.5.1 Προβλήματα διαχείρισης που αντιμετωπίζουν οι φοιτητές

Σε έρευνα [11] που διεξήχθη στο Πανεπιστήμιο του Τσουκούμπα της Ιαπωνίας, η οποία διερευνούσε τη διαχείριση του προγραμματισμού των εργασιών από την πλευρά των φοιτητών, παρατηρήθηκε πως η πλειοψηφία τους αντιμετωπίζει δυσκολίες στην εκκίνηση μιας νέας εργασίας. Οι βασικοί λόγοι που καταγράφηκαν περιλαμβάνουν: α) την έλλειψη χρόνου (26,9%), β) την αγνόηση της εργασίας επειδή τη θεωρούσαν ελάσσονος σημασίας (15,7%), γ) επειδή τη ξέχασαν (12,3%), δ) λόγω κακής συνεργασίας (11,2%) και ε) επειδή ήταν κουραστική (8,9%). Οι τρεις πρώτοι λόγοι, που καλύπτουν το μεγαλύτερο ποσοστό (54,9%), αφορούν σε θέματα κακής οργάνωσης από την πλευρά των φοιτητών, υποδεικνύοντας την ανάγκη για αποτελεσματικότερα εργαλεία χρονοπρογραμματισμού.

Παράλληλα, μια διαφορετική έρευνα [38] καταδεικνύει πάλι πως το κυριότερο πρόβλημα που αντιμετωπίζουν οι φοιτητές είναι η σωστή δόμηση του προγράμματός τους. Παρατηρήθηκε ότι ο τρόπος με τον οποίο οργανώνουν το διάβασμά τους καθοδηγείται κυρίως από τις καταληκτικές ημερομηνίες των εργασιών τους, με αποτέλεσμα να παραμελούν άλλες σημαντικές ακαδημαϊκές δραστηριότητες, όπως η παρακολούθηση διαλέξεων. Αυτό οδηγεί σε ανισορροπία μεταξύ των ακαδημαϊκών τους υποχρεώσεων, επηρεάζοντας την απόδοσή τους συνολικά.

Από τις παραπάνω έρευνες προκύπτουν ορισμένα σημαντικά συμπεράσματα. Πρώτον, η έλλειψη οργανωτικών δεξιοτήτων παραμένει ένας από τους κύριους παράγοντες που εμποδίζουν την αποτελεσματική διαχείριση των εργασιών από τους φοιτητές. Οι λόγοι αυτοί συνδέονται συχνά με την αναβλητικότητα και την έλλειψη εργαλείων που θα μπορούσαν να βοηθήσουν στην αποδοτικότερη οργάνωση των καθημερινών τους υποχρεώσεων. Δεύτερον, η ανάγκη για ένα δομημένο σύστημα προγραμματισμού είναι εμφανής, καθώς θα μπορούσε να παρέχει σαφείς προτεραιότητες και να συμβάλει στη μείωση του άγχους που προκαλείται από τις καταληκτικές ημερομηνίες. Συνεπώς, ένα αποτελεσματικό σύστημα διαχείρισης και προγραμματισμού των εργασιών, προσαρμοσμένο στις ανάγκες των φοιτητών, μπορεί να λειτουργήσει ως βασικό εργαλείο για την ενίσχυση της παραγωγικότητας και της επιτυχίας τους.

1.5.2 Χαρακτηριστικά που οι φοιτητές θα επιθυμούσαν σε μια εφαρμογή

Σε έρευνα [38] που πραγματοποιήθηκε στο τμήμα Πληροφορικής του Πανεπιστημίου του Εδιμβούργου, αναδείχθηκαν κάποιες σημαντικές προτιμήσεις και απαιτήσεις της ακαδημαϊκής κοινότητας σχετικά με τη λειτουργικότητα των εφαρμογών διαχείρισης εργασιών. Η πλειοψηφία των συμμετεχόντων τόνισε τη σημασία της ύπαρξης ενός ενσωματωμένου ημερολογίου, το οποίο θα παρέχει τη δυνατότητα καταγραφής

των ημερομηνιών έναρξης και λήξης για κάθε εργασία. Αυτή η λειτουργία θεωρήθηκε κρίσιμη για τη σωστή οργάνωση και τον προγραμματισμό των υποχρεώσεων, καθώς επιτρέπει στους χρήστες να έχουν σαφή εικόνα των προθεσμιών τους. Παράλληλα, υπογραμμίστηκε η αξία της χρωματικής ταξινόμησης (color-coding), η οποία διευκολύνει τη διάκριση μεταξύ διαφορετικών κατηγοριών ή τύπων εργασιών, ενισχύοντας τη διαφάνεια και την ευκολία χρήσης της εφαρμογής.

Επιπλέον, οι συμμετέχοντες επισήμαναν την ανάγκη για ειδοποιήσεις/γνωστοποιήσεις, οι οποίες θα λειτουργούν ως υπενθυμίσεις για τις επερχόμενες προθεσμίες ή τις εκκρεμότητες που απαιτούν άμεση προσοχή. Εξίσου σημαντική θεωρήθηκε η ύπαρξη to-do λιστών, οι οποίες θα διαθέτουν λειτουργίες όπως ιεράρχηση των εργασιών, ομαδοποίηση σε κατηγορίες, και δυνατότητα εμφάνισης μπάρας προοδίου. Αυτές οι δυνατότητες συμβάλλουν στην αποτελεσματικότερη παρακολούθηση της προοδίου των εργασιών και στη δημιουργία μιας αίσθησης ολοκλήρωσης και επίτευξης στόχων.

Ιδιαίτερο ενδιαφέρον παρουσίασε η πρόταση για την ενσωμάτωση ενός συστήματος ανταμοιβής, το οποίο στοχεύει στην ενθάρρυνση των φοιτητών να ολοκληρώνουν τις εργασίες τους εγκαίρως. Ένα τέτοιο σύστημα θα μπορούσε να περιλαμβάνει την εμφάνιση γραφικών στοιχείων όπως κομφετί ή εικονικά μετάλλια, ή την ανταμοιβή πόντων, συμβάλλοντας στη δημιουργία κινήτρων. Η εισαγωγή αυτού του συστήματος έχει σκοπό να ενισχύσει τη δέσμευση και την παραγωγικότητα των χρηστών, προσφέροντας έναν πιο διαδραστικό και ελκυστικό τρόπο διαχείρισης εργασιών. Με την ενσωμάτωση χαρακτηριστικών όπως τα προαναφερθέντα, τέτοιες εφαρμογές μπορούν να βελτιώσουν ουσιαστικά την παραγωγικότητα και την αποτελεσματικότητα, προσφέροντας παράλληλα μια ευχάριστη εμπειρία χρήσης.

Κεφάλαιο 2

Low-Code

“*The future of coding is no coding at all.*”

—Chris Wanstrath, Co-Founder, Github

Πριν προχωρήσουμε στην περιγραφή της υλοποίησης της εφαρμογής είναι κρίσιμο να αναφερθούμε στην έννοια του χαμηλού κώδικα (low code). Ο χαμηλός κώδικας αποτελεί απάντηση στην ανάγκη για γρηγορότερη παραγωγή ποιοτικών εφαρμογών, μάλιστα επιτρέποντας σε χρήστες με περιορισμένες ή μηδενικές γνώσεις προγραμματισμού να συμμετέχουν ενεργά στη διαδικασία ανάπτυξης. Η έννοια αυτή συνδέεται στενά με τις παλαιότερες προσπάθειες αυτοματοποίησης της ανάπτυξης λογισμικού, όπως το Computer-Aided Software Engineering (CASE) και η Model-Driven Architecture (MDA), οι οποίες αποτέλεσαν τις θεωρητικές ρίζες του χαμηλού κώδικα.

Στο κεφάλαιο αυτό, παρουσιάζονται οι βασικές αρχές, τα χαρακτηριστικά και τα πλεονεκτήματα της προσέγγισης αυτής, το ιστορικό υπόβαθρο, η έννοια του προγραμματιστή πολίτη (citizen developer), και γίνεται αναφορά σε δημοφιλείς Πλατφόρμες Ανάπτυξης Εφαρμογών σε Low-Code και στον τρόπο που αυτές έχουν αναδιαμορφώσει το τοπίο της ανάπτυξης λογισμικού, αποτελώντας τη βάση για την εφαρμογή που παρουσιάζεται στην παρούσα εργασία. Μια εξ’ αυτών, η Mendix, είναι αυτή που έχει χρησιμοποιηθεί για την ανάπτυξη της εφαρμογής και θα αναλυθεί στο επόμενο κεφάλαιο.

2.1 Τι είναι ο χαμηλός κώδικας

“*When you can visually create new business applications with minimal hand-coding –when your developers can do more of greater value, faster—that’s low-code.*” [39]

Για να κατανοήσουμε καλύτερα την έννοια του χαμηλού κώδικα, μπορούμε να εξετάσουμε την εξέλιξη των γλωσσών προγραμματισμού. Για παράδειγμα, η Python, μια γλώσσα υφηλού επιπέδου, θα μπορούσε να χαρακτηριστεί ως χαμηλός κώδικας σε σύγκριση με τη C++. Αντίστοιχα η C θα μπορούσε να χαρακτηριστεί και αυτή

χαμηλός κώδικας συγχριτικά με την Assembly, όπως και η Assembly είναι χαμηλός κώδικας αν τη συγχρίνουμε με το να αλλάζουμε χειροκίνητα μηδενικά και άσσους στους καταχωρητές. Πρακτικά, η εξέλιξη του προγραμματισμού ταυτίζεται με την εξέλιξη του χαμηλού κώδικα.

Επομένως, ο χαμηλός κώδικας, αν και ονομάστηκε πρόσφατα ως όρος, η έννοιά του δεν είναι καθόλου καινούρια, και είναι η άμεση εξέλιξη του υψηλού επιπέδου γλωσσών προγραμματισμού (4GLs) των τελευταίων δεκαετιών. Το υψηλότερο προγραμματιστικό επίπεδο με τις αφαιρέσεις που διαθέτει, επιτρέπει ευκολότερη και γρηγορότερη ανάπτυξη λογισμικού. Πέρα όμως από τους χρήστες που είναι ήδη προγραμματιστές, προσφέρει επιπλέον τη δυνατότητα σε χρήστες με λίγη ή και καθόλου προγραμματιστική εμπειρία (προγραμματιστές πολίτες – περιγράφονται αναλυτικότερα στο 2.1.3), να τροποποιήσουν εφαρμογές ή και να φτιάξουν εξ' ολοκλήρου τις δικές τους, με την ίδια λογική όπως η Python επέτρεψε σε περισσότερο κόσμο να προγραμματίσει σε σχέση με την Assembly.

Ο προγραμματισμός σε χαμηλό κώδικα πραγματοποιείται σε πλατφόρμες που ονομάζονται **Πλατφόρμες Ανάπτυξης Λογισμικού σε Low-Code** (Low-Code Development Platforms – LCDPs), οι οποίες θα αναλυθούν στην ενότητα 2.3. Οι πλατφόρμες περιλαμβάνουν γραφικό περιβάλλον με drag-and-drop και WYSIWYG (What-You-See-Is-What-You-Get) editors, επιτρέποντας την πιο γρήγορη και ενστικτώδη κατασκευή εφαρμογών. Αυτός ο οπτικός προγραμματισμός (visual programming) είναι σημαντικός παράγοντας στην προσβασιμότητα που προσφέρει ο χαμηλός κώδικας. [17] [33] [35]

2.1.1 Οπτικός προγραμματισμός (visual programming)

Στο σχήμα 2.1 παρατίθενται δύο παραδείγματα από την παραδοσιακή ανάπτυξη εφαρμογών και την ανάπτυξη εφαρμογών σε low-code στην πλατφόρμα Mendix.

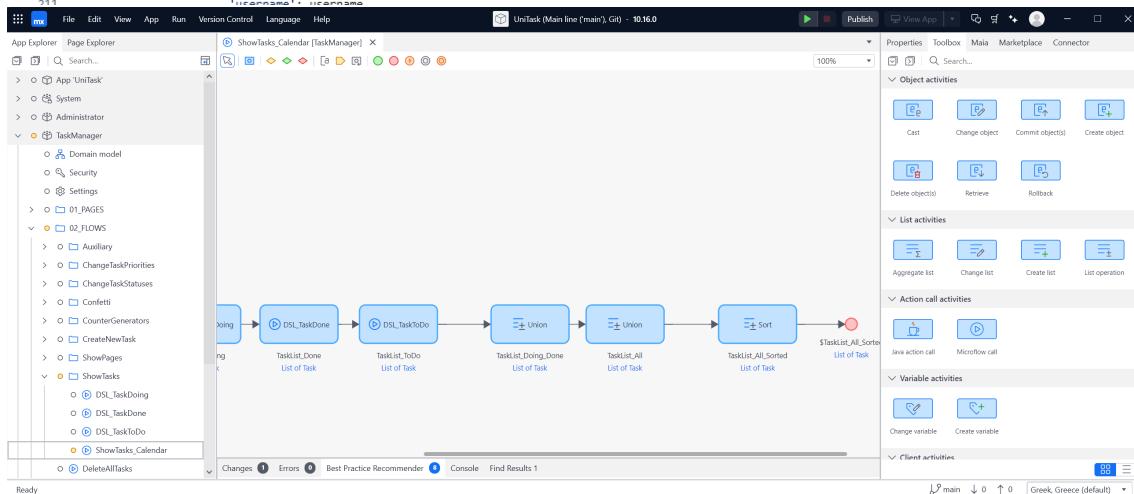
Στο γραφικό περιβάλλον, ο προγραμματισμός γίνεται σε ένα διάγραμμα ροής με drag-and-drop επαναχρησιμοποιούμενα στοιχεία. Αυτή η προσέγγιση διευκολύνει την ανάπτυξη εφαρμογών, καθώς επιτρέπει στους χρήστες να επικεντρωθούν στη λογική της εφαρμογής, χωρίς να χρειάζεται να ασχοληθούν με τη σύνταξη και τις λεπτομέρειες του κώδικα. Στο σχήμα 2.1, παρατηρούμε ένα χαρακτηριστικό παράδειγμα επαναχρησιμοποιούμενων στοιχείων· τα μπλε τετράγωνα στο διάγραμμα ροής αποτελούν βασικά δομικά συστατικά που μπορούν να τοποθετηθούν και να συνδεθούν εύκολα. Στη δεξιά πλευρά της οθόνης εμφανίζεται μια βιβλιοθήκη εργαλείων (toolbox), μέσω της οποίας οι χρήστες μπορούν να επιλέξουν και να προσθέσουν τα απαραίτητα στοιχεία στο διάγραμμά τους. Η διαδικασία αυτή, σε αντίθεση με την παραδοσιακή γραμμή-γραμμή σύνταξη κώδικα, καθιστά την ανάπτυξη εφαρμογών εξαιρετικά γρήγορη, μειώνοντας τον χρόνο εκμάθησης και διευρύνοντας το φάσμα των χρηστών που μπορούν να συμμετάσχουν σε αυτήν.

Η οπτικοποίηση του προγραμματισμού αποτελεί μια εκ θεμελίων επαναστατι-

```

187         }
188     });
189
190     // ===== CREATE RESCUER ACCOUNT =====
191     // Create new Rescuer
192     document.getElementById('createRescuerForm').addEventListener('submit', function(e) {
193         e.preventDefault();
194
195         // Function to create rescuer account
196         function createRescuerAccount() {
197             return new Promise((resolve, reject) => {
198                 // Get form data
199                 var name = document.getElementById('rescuerName').value;
200                 var username = document.getElementById('rescuerUsername').value;
201                 var email = document.getElementById('rescuerEmail').value;
202                 var password = document.getElementById('rescuerPassword').value;
203                 var phone = document.getElementById('rescuerPhone').value;
204
205                 // Make an AJAX POST request to create the rescuer account
206                 $.ajax({
207                     url: './php/create_rescuer.php',
208                     type: 'POST',
209                     data: {
210                         'name': name,
211                         'username': username
212                     }
213                 })
214             })
215         }
216     });

```



Σχήμα 2.1: Παραδοσιακός κώδικας και το γραφικό περιβάλλον του Mendix.

κή αλλαγή, καθώς αναδεικνύει μια βαθιά ανθρώπινη ανάγκη: τη χρήση της οπτικής επικοινωνίας για την κατανόηση και τη μεταφορά ιδεών. Εξάλλου οι πρώτες πετρογραφίες και οι ζωγραφιές στους τοίχους των σπηλαίων αποδεικνύουν ότι η οπτική παράσταση ήταν ανέκαθεν ένα ισχυρό εργαλείο επικοινωνίας [18]. Ο οπτικός προγραμματισμός στηρίζεται σε αυτή τη λογική και τη μεταφέρει στον σύγχρονο κόσμο της τεχνολογίας. Η χρήση του ποντικιού, το οποίο θεωρείται πιο προσιτό από το πληκτρολόγιο για τους περισσότερους χρήστες, διευκολύνει τη διαδικασία εισαγωγής, επεξεργασίας και διασύνδεσης στοιχείων. Το αποτέλεσμα είναι ένα περιβάλλον που συνδυάζει λειτουργικότητα και ευκολία, εξαλείφοντας την ανάγκη για εξειδικευμένες τεχνικές γνώσεις, ενώ ταυτόχρονα αυξάνει την αποδοτικότητα της ανάπτυξης.

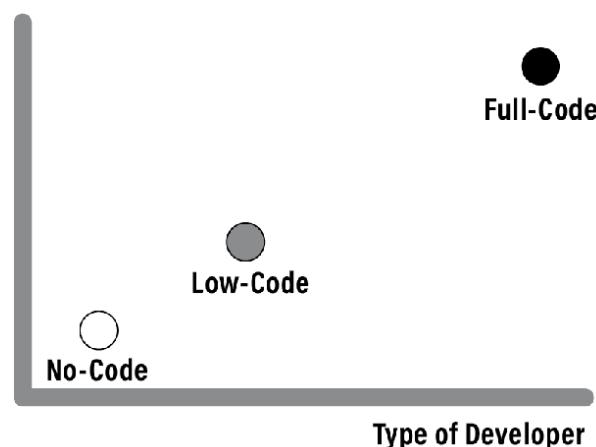
Τέλος, είναι σημαντικό να αναφερθεί πως η οπτική προσέγγιση επιτρέπει την ταχύτερη ανίχνευση σφαλμάτων και την πιο αποτελεσματική συνεργασία μεταξύ των μελών μιας ομάδας, καθώς ένα διάγραμμα ροής είναι άμεσα κατανοητό από όλους τους εμπλεκόμενους, ανεξαρτήτως τεχνικού υποβάθρου.

2.1.2 Καθόλου κώδικας (no-code)

Ένας εύληπτος τρόπος για να κατανοήσουμε τη διαφορά ανάμεσα στον χαμηλό κώδικα και τον καθόλου κώδικα είναι η προσέγγιση που ακολουθείται στην αλληλεπίδραση με το λογισμικό. Στα no-code περιβάλλοντα, οι χρήστες δε χρειάζεται να χρησιμοποιούν καθόλου το πληκτρολόγιο· όλες οι ενέργειες πραγματοποιούνται μέσω ποντικιού, αξιοποιώντας ένα πλήρως γραφικό περιβάλλον. Η διαδικασία αυτή εξαλείφει κάθε ανάγκη γραφικής κώδικα, καθιστώντας τα no-code περιβάλλοντα ιδιαίτερα ελκυστικά για χρήστες που δε διαθέτουν τεχνικές γνώσεις.

Ένα βασικό μειονέκτημα αυτών των συστημάτων είναι η έλλειψη ευελιξίας. Οι χρήστες δεν μπορούν να προσαρμόσουν πλήρως τις εφαρμογές στις ιδιαίτερες ανάγκες τους, καθώς αφού δεν υπάρχει η δυνατότητα για εξατομίκευση με custom κώδικα, οι χρήστες περιορίζονται αποκλειστικά από τις δυνατότητες που έχουν προκαθοριστεί από την πλατφόρμα. Από την άλλη, ο περιοριστικός χαρακτήρας αυτών των εργαλείων ελαχιστοποιεί την πιθανότητα εμφάνισης σφαλμάτων και έτσι αυξάνεται η απλότητα και η ακρίβεια.

Αντίθετα, ο χαμηλός κώδικας (low-code) συνδυάζει τα καλύτερα στοιχεία και των δύο κόσμων: της παραδοσιακής προγραμματιστικής διαδικασίας και των no-code εργαλείων. Οι χρήστες μπορούν να αξιοποιήσουν την απλότητα και την ευκολία του γραφικού περιβάλλοντος, αλλά παράλληλα έχουν τη δυνατότητα να ενσωματώσουν τον δικό τους κώδικα για πλήρη παραμετροποίηση και ανάπτυξη. Αυτή η προσέγγιση καθιστά τα low-code εργαλεία ιδανικά για πιο σύνθετα έργα, όπου απαιτούνται πιο προσαρμοσμένες λύσεις. [33]



Σχήμα 2.2: Σύγκριση ανάμεσα στην προγραμματιστική εμπειρία των χρηστών και την προγραμματιστική προσέγγιση που χρησιμοποιούν [33]

2.1.3 Η έννοια του προγραμματιστή πολίτη (citizen developer)

Ο προγραμματιστής πολίτης είναι ένας χρήστης που αναπτύσσει εφαρμογές σε συνεργασία με τους προγραμματιστές, χωρίς να είναι απαραίτητο να διαθέτει προγραμματιστικές γνώσεις. Η έλλειψη προηγούμενων γνώσεων και εμπειρίας στον προγραμματισμό δεν τον εμποδίζει από το να συνεισφέρει στην ανάπτυξη με ισότιμο τρόπο όπως οι παραδοσιακοί προγραμματιστές.

Μια περίληψη των βασικών διαφορών που παρατηρούνται ανάμεσα στους προγραμματιστές πολίτες και τους μηχανικούς λογισμικού συνοφίζεται στον παρακάτω πίνακα [33]:

Χαρακτηριστικό	Παραδοσιακός μηχανικός λογισμικού	Προγραμματιστής πολίτης
Γνωστικό υπόβαθρο	Μηχανική λογισμικού ή Επιστήμη των Υπολογιστών	Ποικίλει
Θέση στην επιχείρηση	Τεχνολογία πληροφοριών (IT), DevOps	Μάρκετινγκ, πωλήσεις, HR, λογιστικά
Γνώσεις που αφορούν την επιχείρηση	Λίγες	Πολλές

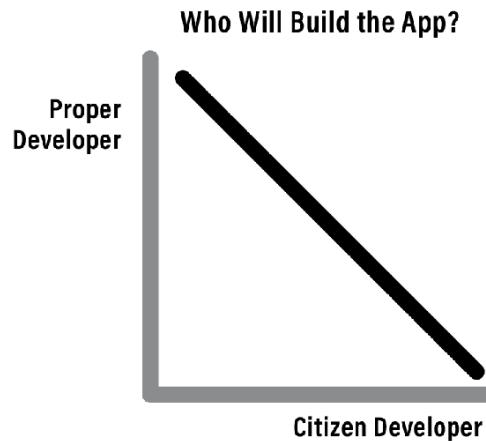
Ο ρόλος των προγραμματιστών πολιτών αναμένεται να γνωρίσει σημαντική άνοδο τα επόμενα χρόνια. Δεν πρόκειται τόσο για μια νέα εξειδικευμένη θέση εργασίας όσο για ένα πρόσθετο χαρακτηριστικό που θα ενσωματωθεί σε ήδη υπάρχουσες θέσεις. Οι εργαζόμενοι που αναλαμβάνουν διοικητικές, επιχειρηματικές ή άλλες λειτουργικές αρμοδιότητες θα αποκτούν δεξιότητες ανάπτυξης λογισμικού, επιτρέποντας τους να αναπτύσσουν λύσεις προσαρμοσμένες στις ανάγκες τους, χωρίς να εξαρτώνται αποκλειστικά από τις ομάδες προγραμματιστών.

Η Gartner, μια κορυφαία διεθνής εταιρία ερευνών και συμβουλευτικών υπηρεσιών που εξειδικεύεται στους τομείς της τεχνολογίας, της επιχειρηματικής στρατηγικής και της καινοτομίας, υπογραμμίζει τη δυναμική αυτής της εξέλιξης. Σύμφωνα με έκθεσή της, προβλέπεται πως «έως το 2023, ο αριθμός των ενεργών προγραμματιστών πολιτών σε μεγάλες επιχειρήσεις θα είναι τουλάχιστον τετραπλάσιος από τον αριθμό των παραδοσιακών προγραμματιστών» [24].

Επιπλέον, η Microsoft ενισχύει αυτήν την πρόβλεψη, επισημαίνοντας ότι «μέχρι το 2025, οι προγραμματιστές πολίτες θα έχουν δημιουργήσει 450 εκατομμύρια εφαρμογές χρησιμοποιώντας πλατφόρμες χαμηλού ή καθόλου κώδικα» [29]. Αυτή η εντυπωσιακή αριθμητική αύξηση σηματοδοτεί μια νέα εποχή στον τρόπο με τον οποίο προσεγγίζεται η ανάπτυξη λογισμικού, δίνοντας έμφαση στη συμμετοχή ευρύτερων ομάδων εργαζομένων, ενώ παράλληλα αναδεικνύουν τον εκδημοκρατισμό της τεχνολογίας.

2.1.3.1 Διαφορά με τους παραδοσιακούς προγραμματιστές

Οι παραδοσιακοί προγραμματιστές, σε αντίθεση με την αρχική εντύπωση που μπορεί να δημιουργείται, δεν αντιμετωπίζουν τους προγραμματιστές πολίτες με καχυποψία ή φόβο για απώλεια του ρόλου τους. Αντίθετα, δείχνουν ενθουσιασμό για την παρουσία και την προσφορά τους. Ο λόγος πίσω από αυτή τη θετική στάση είναι απλός και συνοψίζεται στο σχήμα 2.3.



Σχήμα 2.3: Ποιος θα φτιάξει την εφαρμογή; [33]

Παρότι οι προγραμματιστές πολίτες μπορούν και συνεισφέρουν στην υλοποίηση απλών εφαρμογών, παραμένει ανέφικτο για αυτούς να αναλάβουν τον σχεδιασμό και την ανάπτυξη πολύπλοκων συστημάτων που απαιτούν σχεδιαστική γνώση και εμπειρία. Αυτή η πραγματικότητα δημιουργεί έναν φυσικό διαχωρισμό ρόλων. Με ένα σημαντικό μέρος της εργασίας που σχετίζεται με καθημερινές, επαναλαμβανόμενες διαδικασίες να μεταφέρεται στους προγραμματιστές πολίτες, οι επαγγελματίες προγραμματιστές απελευθερώνονται από τα βάρη των «αδιάφορων» λεπτομερειών και μπορούν να αφιερώσουν περισσότερη ενέργεια στη σχεδίαση, την αρχιτεκτονική των εφαρμογών και σε πιο στρατηγικούς στόχους. Αυτό σημαίνει ότι αντί να ασχολούνται με λεπτομέρειες που μπορεί να διεκπεραιωθούν με τη βοήθεια εργαλείων χαμηλού ή καθόλου κώδικα, στρέφουν την προσοχή τους σε σύνθετα ζητήματα όπως η κλιμάκωση των συστημάτων, η διασφάλιση της ασφάλειας, η βελτιστοποίηση των επιδόσεων και η δημιουργία καινοτόμων λύσεων.

Οι προγραμματιστές πολίτες και οι παραδοσιακοί προγραμματιστές μαζί συνθέτουν μια δυναμική ομάδα που καλύπτει ένα ευρύ φάσμα αναγκών, προόγνοντας την αποτελεσματικότητα και την καινοτομία στην ανάπτυξη λογισμικού.

2.1.3.2 Χαρακτηριστικά των προγραμματιστών πολιτών

Σε μια έρευνα που πραγματοποιήθηκε από την εταιρεία QuickBase [23], στην οποία συμμετείχαν 148 αυτοαποκαλούμενοι προγραμματιστές πολίτες, αναδεικνύονται

σημαντικά χαρακτηριστικά αυτής της ομάδας. Σύμφωνα με τα αποτελέσματα της έρευνας, το 97% των συμμετεχόντων κατείχε βασικές δεξιότητες στη χρήση εργαλείων επεξεργασίας κειμένου και υπολογιστικών φύλλων, ενώ το 36% των ερωτηθέντων διέθετε γνώσεις front-end web development, όπως HTML, CSS και JavaScript, επιβεβαιώνοντας ότι ορισμένοι προγραμματιστές πολίτες διαθέτουν τεχνικό υπόβαθρο πέρα από τα βασικά. Επιπλέον, ένα μικρότερο αλλά εξίσου σημαντικό ποσοστό, το 8%, είχε επαφή με πιο προχωρημένες γλώσσες προγραμματισμού, όπως Java, .NET, Python, Ruby και PHP, δείχνοντας ότι οι δεξιότητες ορισμένων προγραμματιστών πολιτών επεκτείνονται και σε πιο παραδοσιακές τεχνολογίες ανάπτυξης λογισμικού.



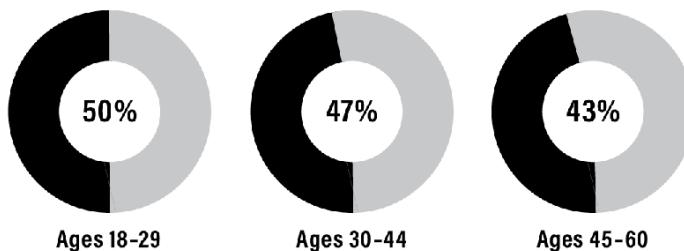
Σχήμα 2.4: Επαγγελματικό υπόβαθρο προγραμματιστών πολιτών. [33]

Το σχήμα 2.4 παρουσιάζει έναν επιπλέον ενδιαφέρων διαχωρισμό: το 72% των προγραμματιστών πολιτών δεν είναι επαγγελματίες προγραμματιστές, αλλά προέρχονται από διαφορετικά επαγγελματικά αντικείμενα. Από την άλλη πλευρά, το 28% των συμμετεχόντων που είναι ήδη επαγγελματίες προγραμματιστές δείχνει μια τάση αποδοχής των εργαλείων χαμηλού και καθόλου κώδικα ακόμα και από αυτούς που έχουν την ικανότητα να γράφουν παραδοσιακό κώδικα. Τέλος, το σχήμα 2.5 φανερώνει πως οι νεότερες γενιές εμφανίζουν αυξημένες πιθανότητες να ασχολούνται ως προγραμματιστές πολίτες. Αυτή η παρατήρηση μπορεί να αποδοθεί στη μεγαλύτερη εξοικείωση των νεότερων ηλικιακά με την τεχνολογία και τα ψηφιακά εργαλεία, καθώς και στη διάθεση για καινοτομία και πειραματισμό που χαρακτηρίζει τις γενιές αυτές. Επιπλέον, η αυξημένη διείσδυση εργαλείων χαμηλού κώδικα στην εκπαίδευση και στο επαγγελματικό περιβάλλον φαίνεται να ενθαρρύνει τη συμμετοχή νεότερων ατόμων, δημιουργώντας τις προϋποθέσεις για περαιτέρω διάδοση του φαινομένου.

2.2 Πώς φτάσαμε στον χαμηλό κώδικα

Πριν προχωρήσουμε στις Πλατφόρμες Ανάπτυξης Λογισμικού σε Low-Code, αξίζει να αναφερθούμε πρώτα στις τεχνολογικές εξελίξεις που μας οδήγησαν σήμερα στην ύπαρξη αυτών των πλατφορμών, ξεκινώντας με τον ορισμό της μηχανικής λογισμικού.

Η μηχανική λογισμικού είναι ο τομέας που ασχολείται με τη συστηματική ανάπτυξη και διαχείριση υπολογιστικών συστημάτων. Ορίζεται ως η πειθαρχημένη και αυστηρή



Σχήμα 2.5: Ηλικιακή κατανομή προγραμματιστών πολιτών. [37]

εφαρμογή μεθόδων, διαδικασιών και εργαλείων στη διαχείριση και ανάπτυξη υπολογιστικών συστημάτων. Ιστορικά, έχει περάσει πολλά στάδια μέχρι να αρχίσουμε να αναφερόμαστε και σε χρήση χαμηλού κώδικα. Μέχρι τη δεκαετία του 1970, η ανάπτυξη πληροφοριακών συστημάτων έμοιαζε περισσότερο με τέχνη παρά με επιστήμη, καθώς δεν υπήρχε τυποποιημένη διαδικασία ή καμία συγκεκριμένη δομή για την ανάπτυξη των προγραμμάτων. Οι προγραμματιστές λαμβάνανε απλώς τις απαιτήσεις από τους χρήστες και, έπειτα από κάποιο χρονικό διάστημα, παρέδιδαν ένα σύστημα που συνήθως δεν κάλυπτε όλες τις ανάγκες του χρήστη, αλλά παρέμενε χρήσιμο σε σχέση με την προηγούμενη κατάσταση. Αυτή η μέθοδος, γνωστή και ως “χλασική μέθοδος”, χαρακτηρίζεται από την έλλειψη τυποποίησης και αναφορών (documentation), καθώς οι διαδικασίες ήταν αδόμητες και συχνά ανεπίσημες.

Η ανάγκη για αύξηση της παραγωγικότητας στον κύκλο ζωής έκδοσης λογισμικού (software release life cycle)¹ οδήγησε στην εμφάνιση “επίσημων μεθόδων” στα τέλη της δεκαετίας του 1970. Ο στόχος αυτών των μεθόδων ήταν η τυποποίηση του σχεδιασμού για τη βελτίωση της ποιότητας του λογισμικού. Ένα από τα πιο χαρακτηριστικά παραδείγματα αυτών των μεθόδων είναι η χρήση δομημένης ανάλυσης (structured analysis), η οποία προσέφερε μια περισσότερο οργανωμένη και συστηματική προσέγγιση στην ανάπτυξη λογισμικού. Οι μηχανικοί μπορούσαν να χρησιμοποιούν εργαλεία όπως τα διαγράμματα ροής δεδομένων (data flow diagrams)², επιτρέποντας τη σαφέστερη κατανόηση και ανάλυση των απαιτήσεων του συστήματος, ή τα μοντέλα οντοτήτων-συσχετίσεων (ER models), τα οποία περιγράφουν ένα σύνολο αντικειμένων (οντότητες) και τις σχέσεις μεταξύ αυτών.

Με την περαιτέρω ανάπτυξη των προσωπικών υπολογιστών στα μέσα της δεκαετίας του 1980, η χρήση επίσημων μεθόδων για την ανάπτυξη λογισμικού έγινε μονόδορος. Αυτή η ανάγκη για αυστηρές και οργανωμένες διαδικασίες ενίσχυσε την τάση για αυτοματοποίηση του προγραμματισμού και την ελαχιστοποίηση της χειροκίνητης γρα-

¹Πρόκειται μια έννοια που αναφέρεται στις φάσεις ανάπτυξης και ύπαρξης ενός λογισμικού. Εξεινάσιει από τη σύλληψη της ιδέας, τη μελέτη για τις απαιτήσεις του, την υλοποίηση του, τη διάθεση του προϊόντος στο πελάτη, τη υποστήριξή του με ενημερώσεις και τέλος την απόσυρσή του.

²Είναι μια οπτική αναπαράσταση της ροής των δεδομένων σε ένα σύστημα. Αναγράφονται οι διεργασίες (κύκλοι), οι είσοδοι και έξοδοι (τετράγωνα) και η αποθήκευση των δεδομένων (παράλληλες γραμμές).

φής κώδικα, κάτι που αποτέλεσε θεμελιώδη αλλαγή στον τρόπο που αναπτύσσονταν οι υπολογιστικές εφαρμογές. [7, 8, 30]

Αυτή η τάση οδήγησε σε σημαντικές εξελίξεις στον τομέα του προγραμματισμού, με κυριότερες τις εξής: α) τη δεκαετία του 1980, οι γλώσσες προγραμματισμού τέταρτης γενιάς, β) τα CASE περιβάλλοντα, γ) η Ταχεία Ανάπτυξη Εφαρμογών τη δεκαετία του 1990, δ) η ανάπτυξη του Προγραμματισμού Τελικού Χρήστη τη δεκαετία του 2000 και ε) οι αρχιτεκτονικές που βασίζονται σε μοντέλα τις τελευταίες δύο δεκαετίες. Στις επόμενες υποενότητες θα αναφερθούμε πιο εκτεταμένα στα CASE και MDA περιβάλλοντα, τα οποία υπήρξαν και τα κύρια πρωταίτια των Low-Code περιβάλλοντων, αλλά προς το παρόν ας κάνουμε μια αναφορά στις υπόλοιπες:

Οι γλώσσες προγραμματισμού τέταρτης γενιάς (fourth-Generation Programming Languages – 4GLs) σχεδιάστηκαν για να διευκολύνουν την ανάπτυξη λογισμικού και να την κάνουν πιο κατανοητή, προσεγγίζοντας τη φυσική ανθρώπινη γλώσσα. Σε αντίθεση με τις γλώσσες προγραμματισμού τρίτης γενιάς, όπως η C και η Java, οι οποίες απαιτούν πιο εξειδικευμένες γνώσεις και είναι πιο κοντά στο μηχάνημα, οι 4GLs όπως η Python, SQL και Ruby παρέχουν υψηλότερου επιπέδου αφαιρέσεις, καθιστώντας τον προγραμματισμό πιο προσβάσιμο και λιγότερο χρονοβόρο. [10]

Η Ταχεία Ανάπτυξη Εφαρμογών (Rapid Application Development – RAD) που εμφανίστηκε τη δεκαετία του 1990, επικεντρώθηκε στη δημιουργία πρωτοτύπων και χρησιμοποίησε εργαλεία που επέτρεπαν στους προγραμματιστές να αναπτύσσουν γρήγορα λογισμικά χωρίς να χρειάζεται να ξεκινούν από το μηδέν και να ξοδεύουν πολύτιμο χρόνο για να περιγράψουν λεπτομερώς όλες τις προδιαγραφές του. Παραδείγματα RAD εργαλείων είναι οι GUI builders· πρόκειται για WYSIWYG (What-You-See-Is-What-You-Get) επεξεργαστές για τη γρήγορη ανάπτυξη λογισμικών με διεπαφή χρήστη (user interface). [26]

Ο Προγραμματισμός Τελικού Χρήστη (End-User Development – EUD), που αφορά τη δυνατότητα των απλών χρηστών να προγραμματίζουν και να δημιουργούν εφαρμογές χωρίς να απαιτούνται γνώσεις κώδικα. Παραδείγματα αυτών των εργαλείων περιλαμβάνουν λογιστικά φύλλα όπως το Microsoft Excel και εκπαιδευτικά εργαλεία όπως το Scratch, τα οποία επιτρέπουν στους χρήστες να δημιουργούν απλές εφαρμογές και αυτοματισμούς. [9]

2.2.1 Computer-Aided Software Engineering (CASE)

Τα Computer-Aided Software Engineering (CASE – Μηχανική Λογισμικού Υποβοηθούμενη από Υπολογιστή) περιβάλλοντα αποτέλεσαν μια σημαντική εξέλιξη στον τομέα της ανάπτυξης λογισμικού, προσφέροντας στους μηχανικούς τη δυνατότητα να καταγράφουν και να μοντελοποιούν με συστηματικό και οργανωμένο τρόπο κάθε στάδιο του πληροφοριακού συστήματος, από τις πρώτες περιγραφές των απαιτήσεων του χρήστη μέχρι τη σχεδίαση, την υλοποίηση και τη δοκιμή του τελικού προϊόντος. Αυτά τα εργαλεία δεν περιορίζονται μόνο στη δημιουργία λογισμικού αλλά έδιναν έμφαση

και στη διασφάλιση της συνοχής και της ποιότητάς του, ενισχύοντας τη συστηματικότητα και μειώνοντας τον ανθρώπινο παράγοντα λάθους.

Προτού τα CASE εργαλεία εισαχθούν στην καθημερινότητα των προγραμματιστών, τα περισσότερα εργαλεία ανάπτυξης λογισμικού επικεντρώνονται χυρίως στην επεξεργασία και τη συγγραφή πηγαίου κώδικα καθώς και στην αποσφαλμάτωσή του. Αυτή η προσέγγιση είχε ως αποτέλεσμα οι μηχανικοί λογισμικού να δαπανούν υπερβολικό χρόνο σε χειρωνακτικές εργασίες, χωρίς να έχουν στη διάθεσή τους αυτοματοποιημένα μέσα για την ανάλυση ή τη σχεδίαση του συστήματος. Αντίθετα, τα CASE περιβάλλοντα εισήγαγαν έναν εντελώς νέο τρόπο εργασίας, εστιάζοντας στη μεθοδολογία της ανάπτυξης λογισμικού. Περιελάμβαναν εργαλεία για την ανάλυση απαιτήσεων, για το λογικό σχεδιασμό, τον έλεγχο εγκυρότητας των συστημάτων, την επαναχρησιμοποίηση κώδικα και τη μείωση ή και εξάλειψη των πλεονασμών, βελτιώνοντας σημαντικά τη ροή εργασιών και τον τελικό σχεδιασμό.

Με λίγα λόγια, τα CASE εργαλεία αποτελούν το δεξί χέρι των μηχανικών λογισμικού, καθώς τους επέτρεπε να αυτοματοποιήσουν δύσκολες ή χρονοβόρες διαδικασίες, αυξάνοντας όχι μόνο την παραγωγικότητα αλλά και την ποιότητα της δουλειάς τους. Τα εργαλεία που προσέφεραν τα CASE περιβάλλοντα ποικίλουν· από τη δυνατότητα δημιουργίας διαγραμμάτων για την αναπαράσταση της ροής δεδομένων ή των σχέσεων οντοτήτων (ER diagrams) μέχρι και πλήρη αυτοματοποίηση όλων των σταδίων του κύκλου ζωής του λογισμικού. Ένα ολοκληρωμένο CASE περιβάλλον περιλαμβάνει:

- ένα διαδραστικό και φιλικό για το χρήστη γραφικό περιβάλλον διαχείρισης
- ένα σύνολο από εργαλεία ανάπτυξης (επεξεργαστές κειμένου, λεξικά, αναλυτές σχεδιασμού κ.α.)
- ένα σύνολο από εργαλεία για τον έλεγχο της διαδικασίας (για τον χρονοπρογραμματισμό, τη διασφάλιση ποιότητας κ.α.)
- ένα περιβάλλον βοήθειας με το documentation των εργαλείων
- ένα σύστημα διαχείρισης βάσεων δεδομένων

Τα CASE εργαλεία συνέβαλαν επίσης στη θέσπιση νέων προτύπων στον τομέα της ανάπτυξης λογισμικού. Ο οπτικός προγραμματισμός (visual programming), ο οποίος βασίζεται στη χρήση διαγραμμάτων και γραφικών για την απλοποίηση της σχεδίασης και του προγραμματισμού, και ο προγραμματισμός που βασίζεται σε μοντέλα (model-driven programming), ο οποίος προωθεί τη χρήση αφηρημένων μοντέλων για την αυτοματοποίηση της ανάπτυξης λογισμικού, είναι μόνο δύο από τις τεχνολογίες που επηρεάστηκαν βαθύτατα από τα CASE εργαλεία.

Τελικά, τα CASE περιβάλλοντα αντιπροσωπεύουν την εφαρμογή της πειθαρχημένης μεθοδολογίας της μηχανικής λογισμικού στην πράξη. Δεν ήταν απλώς εργαλεία υποβοήθησης· αποτελούσαν ολοκληρωμένες λύσεις που ενίσχυαν τη συνεργασία, τη διαφάνεια και τη δομημένη σκέψη σε όλα τα στάδια της ανάπτυξης λογισμικού. Παράλληλα, έθεσαν τις βάσεις για την εξέλιξη των σύγχρονων πλατφορμών ανάπτυξης

λογισμικού, όπως οι λύσεις low-code και no-code, που βασίζονται στις ίδιες αρχές απλοποίησης και αυτοματοποίησης. [8, 7, 18, 22]

2.2.2 Model-driven Architecture (MDA)

Τα μοντέλα προσφέρουν τη δυνατότητα αφαίρεσης σε ένα σύστημα, επιτρέποντας στους μηχανικούς να επικεντρώνονται αποκλειστικά στο πρόβλημα που καλούνται να λύσουν και αγνοούν τις περιττές λεπτομέρειες που δε σχετίζονται με αυτό. Αυτή η προσέγγιση είναι ιδιαίτερα χρήσιμη για την κατανόηση και την επεξεργασία πολύπλοκων συστημάτων, όπου οι λεπτομέρειες μπορεί να είναι τόσο πολλές που να καθιστούν δύσκολη τη συνολική θεώρηση του συστήματος. Για παράδειγμα, η μοντέλοποίηση διαφορετικών επιπέδων εμφάνισης ενός συστήματος, όπως το δομικό επίπεδο, το επίπεδο συμπεριφοράς ή το επίπεδο φυσικής υλοποίησης, βοηθά τους μηχανικούς να κατανοούν το σύστημα από διάφορες οπτικές γωνίες.

Η ιδέα της χρήσης μοντέλων αποτέλεσε τη βάση για μια νέα προσέγγιση ανάπτυξης λογισμικού, γνωστή ως μηχανική που βασίζεται σε μοντέλα (model-driven engineering – MDE). Στη MDE, τα μοντέλα δεν είναι απλώς εργαλεία για την αναπαράσταση ενός συστήματος· γίνονται βασικά στοιχεία της διαδικασίας ανάπτυξης λογισμικού. Με τη βοήθεια σαφών κανόνων και τυποποιημένων διαδικασιών, τα μοντέλα μπορούν να περιγράφονται, να μετασχηματίζονται και να αξιοποιούνται για την ανάπτυξη λογισμικού. Για παράδειγμα, μπορούν να χρησιμοποιηθούν κανόνες για τη μετατροπή ενός μοντέλου υψηλού επιπέδου σε ένα πιο λεπτομερές, ή για την παρακολούθηση της συνέπειας μεταξύ στοιχείων διαφορετικών μοντέλων. Αυτή η διαδικασία συνιστά τη βάση για την αρχιτεκτονική που βασίζεται σε μοντέλα (model-driven architecture – MDA).

Η MDA, η οποία υποστηρίζεται από την Object Management Group (OMG) [21], βασίζεται σε ένα σύνολο προτύπων που αφορούν τον ορισμό μοντέλων, συμβολισμών και κανόνων μετασχηματισμού. Τα πρότυπα αυτά περιλαμβάνουν το UML (Unified Modeling Language), το οποίο παρέχει μια κοινή γλώσσα για την αναπαράσταση συστημάτων. Τα μοντέλα χρησιμοποιούνται για τον προσδιορισμό, την προσομοίωση, την επαλήθευση, τον εκσυγχρονισμό, τη συντήρηση, την κατανόηση και τη δημιουργία κώδικα. Στόχος παραμένει η αυτοματοποίηση διαφόρων βημάτων στην ανάπτυξη λογισμικού, κάτι που αυξάνει την παραγωγικότητα και την ποιότητα του παραγόμενου λογισμικού.

Ένα άλλο σημαντικό χαρακτηριστικό της MDA είναι ο διαχωρισμός της λειτουργικότητας μιας εφαρμογής από την υλοποίησή της σε μια συγκεκριμένη πλατφόρμα. Αυτός ο διαχωρισμός επιτρέπει στους προγραμματιστές να επικεντρώνονται περισσότερο στον σχεδιασμό του συστήματος και λιγότερο σε ζητήματα που σχετίζονται με τις τεχνικές λεπτομέρειες της πλατφόρμας υλοποίησης. Ως αποτέλεσμα, τα συστήματα που αναπτύσσονται με βάση τη MDA είναι πιο φορητά και πιο ευέλικτα, ενώ μπορούν να προσαρμόζονται εύκολα σε αλλαγές στις τεχνολογικές απαιτήσεις ή στις

πλατφόρμες. [5, 30, 32]

Εν τέλει, η αρχιτεκτονική που βασίζεται σε μοντέλα έχει φέρει μια αλλαγή στον τρόπο σκέψης των προγραμματιστών και μηχανικών λογισμικού. Εισήγαγε μια κουλτούρα που βασίζεται στην αφαιρετικότητα, τον σωστό διαχωρισμό των χαρακτηριστικών και την αυτοματοποίηση. Με αυτόν τον τρόπο, οι προγραμματιστές μπορούν να αναπτύσσουν λογισμικό που είναι όχι μόνο πιο αποδοτικό και ευέλικτο αλλά και πιο ανθεκτικό σε μελλοντικές τεχνολογικές αλλαγές.

2.2.3 Προγενέστερα λογισμικά οπτικού προγραμματισμού

Στο πεδίο του οπτικού προγραμματισμού που εισήγαγαν τα CASE εργαλεία, οι Πλατφόρμες Ανάπτυξης Εφαρμογών σε Low-Code δεν ήταν οι πρώτες που αξιοποίησαν γραφικό περιβάλλον εργασίας για την ανάπτυξη εφαρμογών. Υπήρχαν ήδη αρκετά προγενέστερα λογισμικά που εισήγαγαν χρήσιμες ιδέες και θέτουν τη βάση για την κατανόηση της εξέλιξης αυτών των τεχνολογιών.

Ένα από τα πρώτα και πιο σημαντικά λογισμικά με γραφικό περιβάλλον ήταν το Microsoft Access, το οποίο έδωσε τη δυνατότητα στους χρήστες να δημιουργούν βάσεις δεδομένων, φόρμες και αναφορές, χρησιμοποιώντας ένα drag-and-drop γραφικό περιβάλλον εργασίας. Αυτό απλοποίησε σε μεγάλο βαθμό τη διαχείριση δεδομένων, καθώς οι χρήστες μπορούσαν να κατασκευάζουν βάσεις δεδομένων, φόρμες χωρίς να απαιτείται εξειδικευμένη γνώση της SQL.

Παρόμοια, το Microsoft FrontPage υπήρξε ένας από τους πρώτους WYSIWYG (What-You-See-Is-What-You-Get) επεξεργαστές για τη δημιουργία και διαχείριση ιστοσελίδων. Το FrontPage επέτρεπε στους χρήστες να δημιουργούν ιστοσελίδες μέσω ενός απλού γραφικού περιβάλλοντος, παρακάμπτοντας την ανάγκη να γράφουν HTML ή CSS από το μηδέν.



Σχήμα 2.6: Το γραφικό περιβάλλον του Microsoft FrontPage

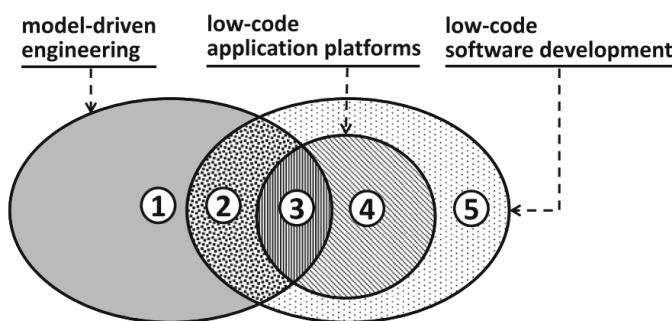
Τα LCDPs που ακολούθησαν πήραν αυτές τις αρχές και τις εξέλιξαν, ενσωματώνοντας προγραμμένα χαρακτηριστικά όπως η ευελιξία στη σύνδεση διαφορετικών

συστημάτων, η χρήση cloud τεχνολογιών ή η υποστήριξη πολλαπλών πλατφορμών, δημιουργώντας εν τέλει εργαλεία που είναι ισχυρά, αλλά ταυτόχρονα προσιτά. [33]

2.3 Πλατφόρμες Ανάπτυξης Εφαρμογών σε Low-Code (LCDP)

Όσο και αν η έννοια των μοντέλων άλλαξε τη μηχανική λογισμικού και έθεσε νέες προδιαγραφές στον σχεδιασμό του, η εξάρτησή της από δύσχρηστα πρότυπα όπως το UML περιόριζε την ευρεία υιοθέτησή της στη βιομηχανία. Ανταποκρινόμενες σε αυτή την πρόκληση, τα τελευταία χρόνια εμφανίστηκαν πλατφόρμες που αξιοποιούν τις αρχές της αφαίρεσης των μοντέλων, αλλά με μια πιο πρακτική και προσβάσιμη προσέγγιση. Αυτές οι πλατφόρμες εστιάζουν στην απλοποίηση της διαδικασίας ανάπτυξης λογισμικού, επιτρέποντας στους χρήστες να δημιουργούν εφαρμογές με ελάχιστη ή καθόλου γραφή κώδικα.

Οι συγκεκριμένες πλατφόρμες, γνωστές ως **Πλατφόρμες Ανάπτυξης Εφαρμογών σε Low-Code** (Low-Code Development Platforms – LCDPs)³, ενσωματώνουν γραφικά περιβάλλοντα εργασίας, προκαθορισμένα πρότυπα και αυτοματοποιημένα εργαλεία, με στόχο να μειώσουν την εξάρτηση από πολύπλοκα τεχνικά μέσα και να επιταχύνουν τον κύκλο ανάπτυξης λογισμικού.[4]

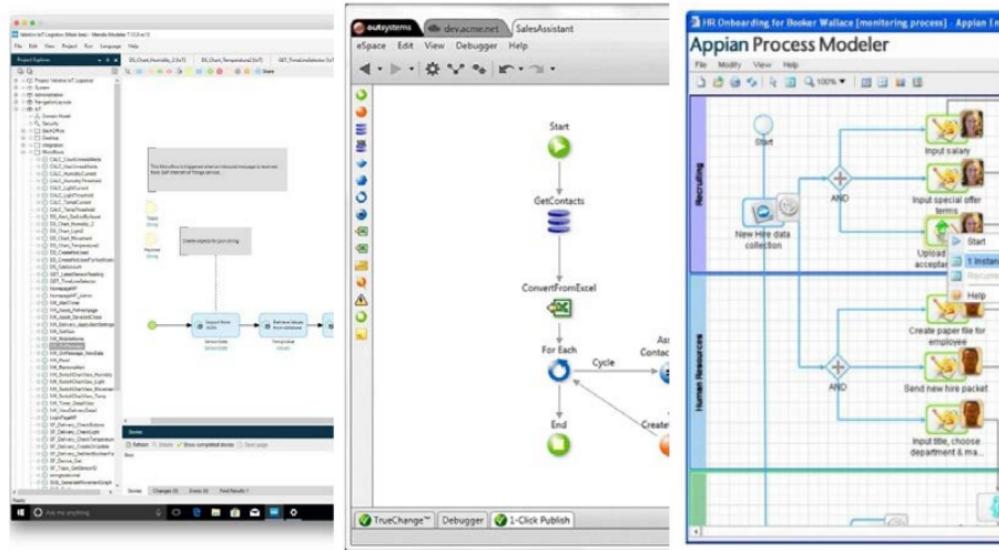


Σχήμα 2.7: Σύγκριση μεταξύ της μηχανικής που βασίζεται σε μοντέλα και των Low-Code πλατφορμών. [30]

Το σχήμα 2.7 αποτυπώνει την αλληλεπίδραση και τις διαφορές μεταξύ της μηχανικής που βασίζεται σε μοντέλα (model-driven engineering), των πλατφορμών ανάπτυξης εφαρμογών σε low-code (low-code application platforms) και της ανάπτυξης λογισμικού σε low-code (low-code software development). Στην περιοχή 1, τα μοντέλα χρησιμοποιούνται χωρίς ιδιαίτερη έμφαση στη μείωση του κώδικα, αντικατοπτρίζοντας μια παραδοσιακή προσέγγιση στη μηχανική λογισμικού. Αντίθετα, στις περιοχές 2 και 3, οι προσπάθειες εστιάζουν στη μείωση του κώδικα, κάτι που αποτελεί κεντρι-

³Εναλλακτικές ονομασίες είναι Low-Code Platforms (LCP), Low-Code Development Platforms (LCDP), ενώ η διαδικασία ανάπτυξης αναφέρεται ως Low-Code Software Development (LCSD). Η έννοια του Low-Code συχνά αναφέρεται και ως Χαμηλός Κώδικας.

κό στόχο των low-code πλατφορμών. Από την άλλη γίνεται να υπάρχουν προσπάθειες μείωσης κώδικα χωρίς να είναι απαραίτητη η χρήση μοντέλων (περιοχές 4 και 5), όπου οι πλατφόρμες συχνά βασίζονται σε άλλου είδους δομές, όπως δεδομένα από σχεσιακές βάσεις ή XML αρχεία. Μια αρίστιμη διαφοροποίηση μεταξύ των low-code application platforms και του low-code software development είναι ότι οι πλατφόρμες προσφέρουν πρόσθετα χαρακτηριστικά, όπως τη δυνατότητα διάθεσης του λογισμικού στο ευρύ κοινό και τη διαχείρισή του καθ' όλη τη διάρκεια του κύκλου ζωής του.



Σχήμα 2.8: Στιγμιότυπα από LCAP από αριστερά προς τα δεξιά: Mendix, OutSystems, Appian [17]

Μια Πλατφόρμα Ανάπτυξης Εφαρμογών σε Low-Code (LCAP) είναι μια πλατφόρμα ανάπτυξης λογισμικού που έχει σχεδιαστεί για να υποστηρίζει την ταχεία ανάπτυξη, ανάπτυξη και διαχείριση εφαρμογών με ελάχιστο ή καθόλου ανάγκη για παραδοσιακό κώδικα. Οι LCAP συνήθως λειτουργούν σύμφωνα με το μοντέλο Platform-as-a-Service (PaaS) βασιζόμενες στο cloud, χρησιμοποιείται ελάχιστος ή και μηδενικός δομημένος προγραμματισμός (structured programming) αλλά παρέχεται γραφικό περιβάλλον με οπτικές αφαιρέσεις (visual abstractions) για τους προγραμματιστές ή τους προγραμματιστές πολίτες.

Η αποδοτική ανάπτυξη λογισμικού με χαμηλό κόστος και ελάχιστη προσπάθεια αποτελεί το βασικό στόχο των LCAP. Αυτές οι πλατφόρμες διευκολύνουν την γρήγορη προσαρμογή των εφαρμογών στις συνεχώς μεταβαλλόμενες τεχνολογικές ανάγκες και συνθήκες των σύγχρονων λειτουργικών συστημάτων. Με την εξαιρετική ευχέρεια προσαρμογής τους, οι οργανισμοί μπορούν να αντιδράσουν άμεσα στις απαιτήσεις της αγοράς, αναπτύσσοντας εφαρμογές που είναι συμβατές με νέες τεχνολογίες και νέες ανάγκες των χρηστών χωρίς να απαιτείται συνεχής αναθεώρηση του κώδικα.

Η χρήση των LCAP έχει τύχει θετικής αποδοχής από τη βιομηχανία με πολλές εται-

ρείς και οργανισμοί να έχουν ενσωματώσει αυτές τις πλατφόρμες στις στρατηγικές τους για την ανάπτυξη λογισμικού και την υιοθέτησή τους συνεχώς να αυξάνεται. [4, 5, 31]

2.3.1 Χαρακτηριστικά των LCDP

Ακολουθούν κάποια από τα βασικά χαρακτηριστικά που διακρίνουν τις πλατφόρμες ανάπτυξης σε Low-Code:

- **Γραφικό περιβάλλον χρήστη:** Ένα από τα πιο προφανή χαρακτηριστικά των LCDP είναι το γραφικό περιβάλλον χρήστη (GUI). Περιλαμβάνονται drag-and-drop εργαλεία, μηχανές αποφάσεων για τη μοντελοποίηση πολύπλοκης λογικής, κατασκευαστές φορμών (form builders) και άλλα.
- **Φορητότητα και συνεργασία:** Οι LCDP διευκολύνουν τη συνεργασία, επιτρέποντας στους χρήστες να εργάζονται από οποιαδήποτε τοποθεσία και συσκευή, ανεξαρτήτως λειτουργικού συστήματος.
- **Επεκτασιμότητα:** Η επαναχρησιμοποίηση προκατασκευασμένων στοιχείων μειώνει δραστικά τον χρόνο ανάπτυξης, ενώ η ύπαρξη marketplace επιτρέπει στους χρήστες να προσθέτουν νέα modules ή λειτουργίες στις εφαρμογές τους. Έτσι προσφέρεται η δυνατότητα εύκολης ενσωμάτωσης τρίτων modules, καθιστώντας τις εφαρμογές πιο ευέλικτες και προσαρμόσιμες στις ανάγκες των προγραμματιστών ή των τελικών χρηστών.
- **Έλεγχος έκδοσης:** Ο έλεγχος έκδοσης είναι μια κρίσιμη πτυχή για οποιοδήποτε σύστημα ανάπτυξης λογισμικού. Οι LCDP ενσωματώνουν χαρακτηριστικά DevOps, όπως το version control μέσω Git, διευκολύνοντας την παρακολούθηση αλλαγών και την αναίρεση ανεπιθύμητων ενεργειών. Επίσης, η δυνατότητα αυμεσού deploy της εφαρμογής για το κοινό σημαίνει ότι οι ομάδες μπορούν να παραδίδουν γρήγορα νέες λειτουργίες ή διορθώσεις.

2.3.2 Δημοφιλή LCDP

Όπως έχει αναφερθεί, οι πλατφόρμες ανάπτυξης εφαρμογών σε Low-Code έχουν γίνει δημοφιλείς στη βιομηχανία λογισμικού, με πολλές εταιρείες να επιλέγουν να χρησιμοποιούν αυτές τις πλατφόρμες για την ανάπτυξη των εφαρμογών τους. Ακολουθούν μερικές από τις πιο δημοφιλείς πλατφόρμες ανάπτυξης εφαρμογών σε Low-Code:

2.3.2.1 Mendix

2.3.2.2 OutSystems

2.3.2.3 Microsoft Power Apps

Κεφάλαιο 3

Mendix

3.1 Τι είναι το Mendix;

Το Mendix αποτελεί μία από τις πιο διαδεδομένες και καινοτόμες πλατφόρμες ανάπτυξης λογισμικού που βασίζεται στην προσέγγιση low-code. Ανήκει στις πλατφόρμες που επιτρέπουν την ταχύτερη και πιο αποδοτική ανάπτυξη εφαρμογών χωρίς να απαιτείται εκτενής γνώση προγραμματιστικών γλωσσών, γεγονός που καθιστά τη διαδικασία προσβάσιμη σε χρήστες με περιορισμένη ή καθόλου τεχνική κατάρτιση. Ιδρύθηκε το 2005 στο Ρότερνταμ της Ολλανδίας με στόχο να παρέχει στους επιχειρηματίες και τους οργανισμούς τη δυνατότητα να αναπτύσσουν, να προσαρμόζουν και να διαχειρίζονται εφαρμογές γρήγορα, με ελάχιστο κόστος και χωρίς την ανάγκη περίπλοκων κωδικοποιητικών διαδικασιών.

Η καινοτομία του Mendix έγκειται στην εξαιρετική ευχρηστία του περιβάλλοντος ανάπτυξης, το οποίο ενσωματώνει εργαλεία drag-and-drop και visual modeling, επιτρέποντας στους χρήστες να δημιουργούν ολοκληρωμένες εφαρμογές χωρίς να χρειάζεται να κατανοούν σε βάθος τον προγραμματισμό. Αυτό το χαρακτηριστικό έχει συμβάλλει σημαντικά στην εξάπλωσή του, καθώς δίνει τη δυνατότητα σε επαγγελματίες από διάφορους τομείς, όπως μάρκετινγκ, οικονομικά και ανθρώπινο δυναμικό, να συμμετέχουν ενεργά στη διαδικασία ανάπτυξης εφαρμογών.

Το 2018, το Mendix εξαγοράστηκε από την Siemens, τη μεγαλύτερη βιομηχανική κατασκευαστική εταιρεία στην Ευρώπη, γεγονός που επέφερε σημαντικές εξελίξεις στην πλατφόρμα. Η συγχώνευση αυτή επέτρεψε την ενσωμάτωση προηγμένων βιομηχανικών και IoT (Internet of Things) λύσεων, ενισχύοντας τη θέση του Mendix στην αγορά του λογισμικού επιχειρηματικής εφαρμογής. Μέσω αυτής της συνεργασίας, η Siemens κατάφερε να επωφεληθεί από τις δυνατότητες της low-code πλατφόρμας για να επιταχύνει τη δημιουργία βιομηχανικών εφαρμογών και να προωθήσει τον φημιακό μετασχηματισμό στον τομέα της βιομηχανίας.

Η πλατφόρμα Mendix παρέχει ολοκληρωμένες δυνατότητες, όχι μόνο για την ανάπτυξη αλλά και για την ανάπτυξη cloud-based εφαρμογών, υποστηρίζοντας το DevOps μοντέλο για καλύτερη συνεργασία μεταξύ των ομάδων ανάπτυξης και παραγωγής. Η

δυνατότητα ενσωμάτωσης με άλλες τεχνολογίες και συστήματα, καθώς και η ευκολία στην προσαρμογή της πλατφόρμας στις ανάγκες του εκάστοτε χρήστη, ενδυναμώνει τη δημοτικότητά της σε παγκόσμιο επίπεδο.

Έποι, το Mendix αποτελεί μία από τις πιο ισχυρές και ευέλικτες λύσεις στην αγορά του low-code προγραμματισμού, προσφέροντας αποτελεσματικότητα, ταχύτητα και καινοτομία στην ανάπτυξη λογισμικού, ενώ παράλληλα ενσωματώνει τις πιο σύγχρονες τεχνολογίες για να καλύψει τις ανάγκες επιχειρήσεων που επιθυμούν να παραμείνουν ανταγωνιστικές στην φημιακή εποχή.

[17]



Σχήμα 3.1: Τεταρτημόριο της Gartner με πλατφόρμες ανάπτυξης λογισμικού [12]

Βιβλιογραφία

- [1] David J. Anderson. *Kanban: Successful evolutionary change for your technology business.* Blue Hole Press, 2010.
- [2] Asana. *Manage your team's work, projects, & tasks online • Asana • Asana — asana.com.* <https://asana.com/>. [Accessed 28-12-2024].
- [3] Atlassian. *Jira | Issue & Project Tracking Software | Atlassian — atlassian.com.* <https://www.atlassian.com/software/jira>. [Accessed 28-12-2024].
- [4] Alexander C. Bock και Ulrich Frank. «Low-Code Platform». Στο: *Business and Information Systems Engineering* 63 (6 Δεκ. 2021), σσ. 733–740. ISSN: 18670202. doi: 10.1007/s12599-021-00726-8.
- [5] Alessio Bucaioni, Antonio Cicchetti και Federico Ciccozzi. «Modelling in low-code development: a multi-vocal systematic review». Στο: *Software and Systems Modeling* 21 (5 Οκτ. 2022), σσ. 1959–1981. ISSN: 16191374. doi: 10.1007/s10270-021-00964-0.
- [6] *BUS402: History of Project Management | Saylor Academy — learn.saylor.org.* <https://learn.saylor.org/mod/page/view.php?id=65663>. [Accessed 26-10-2024].
- [7] Albert E Case. *Computer-aided software engineering (case): technology for improving software development productivity.* 1985.
- [8] E. J. Chikofsky. *Software Development — Computer-Aided Software Engineering (CASE).*
- [9] *End-user development - Wikipedia — en.wikipedia.org.* https://en.wikipedia.org/wiki/End-user_development. [Accessed 29-12-2024].
- [10] *Fourth-generation programming language - Wikipedia — en.wikipedia.org.* https://en.wikipedia.org/wiki/Fourth-generation_programming_language. [Accessed 29-12-2024].
- [11] Ryoko Fukuzawa, Hideo Joho και Tetsuya Maeshiro. «Practice and experience of task management of university students: Case of University of Tsukuba, Japan». Στο: *Education for Information* 31 (3 Ιούλ. 2015), σσ. 109–124. ISSN: 01678329. doi: 10.3233/EFI-150953.
- [12] *Gartner® Magic Quadrant™ for Enterprise Low-Code Application Platforms — mendix.com.* <https://www.mendix.com/resources/gartner-magic-quadrant-for-low-code-application-platforms/>. [Accessed 26-11-2024].

- [13] Jack Goody. «Memory in Oral Tradition». Στο: Cambridge University Press, Μαρ. 2013, σσ. 73–94. doi: 10.1017/cbo9781139171137.005.
- [14] *Guide to the Project Management Body of Knowledge*. Project Management Institute, 2021. isbn: 1628256648.
- [15] *Hoover Dam – the Greatest Project in Times of the Great Depression. What Can Be Done to Achieve Success? - Strefa PMI — strefapmi.pl*. <https://strefapmi.pl/strefa-studenta/hoover-dam-the-greatest-project-in-times-of-the-great-depression/>. [Accessed 30-10-2024].
- [16] *Hoover Dam | Description, Location, Construction, Facts, History, & Pictures | Britannica — britannica.com*. <https://www.britannica.com/topic/Hoover-Dam>. [Accessed 25-12-2024].
- [17] Bryan Kasam, Imran McMullen και Micah Kenneweg. *Building Low-Code Applications with Mendix enterprise web and mobile app development made... easy with mendix and the power of no-code development*. Packt Publishing Limited, 2021. isbn: 9781800201422.
- [18] D. L. Kuhn. *Selecting and Effectively Using a Computer-Aided Software Engineering Tool*. 1989.
- [19] Benne Lientz και Kathryn Rea. *Project Management for the 21st Century*. 2007.
- [20] *Manage Your Team's Projects From Anywhere | Trello — trello.com*. <https://trello.com/>. [Accessed 16-10-2024].
- [21] *MDA FAQ | Object Management Group — omg.org*. https://www.omg.org/mda/faq_mda.htm. [Accessed 08-11-2024].
- [22] G. Premkumar και Michael Potter. *Adoption of Computer Aided Software Engineering (CASE) Technology: An Innovation Adoption Perspective*.
- [23] QuickBase. *The State Of Citizen Development Report – September 2015*. https://cdn2.hubspot.net/hubfs/172645/QuickBase_Citizen_Developer_Report.pdf. [Accessed 25-11-2024].
- [24] Quickbase. *Gartner® Report: Future of Work Trends — quickbase.com*. <https://www.quickbase.com/gartner-future-of-work>. [Accessed 25-11-2024].
- [25] *Quipu - Wikipedia — en.wikipedia.org*. <https://en.wikipedia.org/wiki/Quipu>. [Accessed 22-10-2024].
- [26] *Rapid application development - Wikipedia — en.wikipedia.org*. https://en.wikipedia.org/wiki/Rapid_application_development. [Accessed 29-12-2024].
- [27] Thomas Q Reece. *Lukasa: A Luba Memory Device*. doi: doi:10.2307/3335144.
- [28] E. G. Richards. *Mapping time: The calendar and its history*. Oxford University Press, 2000.

- [29] Eric Rosenbaum. *Next frontier in Microsoft, Google, Amazon cloud battle is over a world without code* — *cnbc.com*. <https://www.cnbc.com/2020/04/01/new-microsoft-google-amazon-cloud-battle-over-world-without-code.html>. [Accessed 25-11-2024].
- [30] Davide Di Ruscio κ.ά. «Low-code development and model-driven engineering: Two sides of the same coin?» Στο: *Software and Systems Modeling* 21 (2 Απρ. 2022), σσ. 437–446. issn: 16191374. doi: 10.1007/s10270-021-00970-2.
- [31] Apurvanand Sahay κ.ά. «Supporting the understanding and comparison of low-code development platforms». Στο: *Proceedings - 46th Euromicro Conference on Software Engineering and Advanced Applications, SEAA 2020*. Institute of Electrical και Electronics Engineers Inc., Αύγ. 2020, σσ. 171–178. isbn: 9781728195322. doi: 10.1109/SEAA51224.2020.00036.
- [32] Matthias Book Sami Beydeda και Volker Gruhn. *Model-Driven Software Development*.
- [33] Phil Simon. *Low-Code/No-Code: Citizen Developers and the Surprising Future of Business Applications*. 2022.
- [34] Andrew Stellman. *Learning agile: Understanding scrum, XP, lean, and Kanban*. Findaway World, 2023.
- [35] O'Reilly Editorial Team. «Low-Code and the Democratization of Programming». Στο: *O'Reilly Media* (2021).
- [36] Todoist | A To-Do List to Organize Your Work & Life — *todoist.com*. <https://todoist.com/>. [Accessed 16-10-2024].
- [37] TrackVia. *The next generation worker: The Citizen Developer – Insights on the behaviors and characteristics of an emerging class of technology users within the enterprise*. https://lumenmarketing.com/wp-content/uploads/2017/11/TV_Citizen_Dev.pdf. [Accessed 25-11-2024]. 2014.
- [38] Julia Castillo Trujillo. *Designing A Time Management App For And With Informatics Students*. 2020.
- [39] What Is Low-Code? | IBM — *ibm.com*. <https://www.ibm.com/topics/low-code>. [Accessed 11-10-2024].
- [40] WinWorld: Welcome — *winworldpc.com*. <https://winworldpc.com/home>. [Accessed 31-10-2024].
- [41] Your connected workspace for wiki, docs & projects | Notion — *notion.so*. <https://www.notion.so/>. [Accessed 16-10-2024].
- [42] Μιχαήλ Ξένος. *Ποιότητα Λογισμικού*. GOTSID, 2021. isbn: 9786185560102.