

# Περιεχόμενα

<b>1 Εισαγωγή</b>	<b>1</b>
1.1 Σημασία του προβλήματος . . . . .	1
1.2 Στόχοι της εργασίας . . . . .	1
1.3 Μεθοδολογία προσέγγισης . . . . .	1
1.4 Διάρθρωση της διπλωματικής εργασίας . . . . .	1
<b>2 Διαχείριση Εργασιών</b>	<b>2</b>
2.1 Το πρόβλημα της διαχείρισης εργασιών . . . . .	2
2.1.1 Ορισμός της εργασίας . . . . .	3
2.1.2 Ορισμός της διαχείρισης εργασιών . . . . .	3
2.1.3 Πεδίο εφαρμογής διαχείρισης εργασιών . . . . .	3
2.1.4 Διαφορά με διαχείριση έργου . . . . .	4
2.2 Ιστορική αναδρομή . . . . .	4
2.2.1 Προφορικότητα και μνημονικές συσκευές . . . . .	5
2.2.2 Πρώτα ημερολόγια . . . . .	6
2.2.3 Σύγχρονη εποχή . . . . .	8
2.3 Η συνδρομή της τεχνολογίας . . . . .	9
2.3.1 Ψηφιακά εργαλεία . . . . .	9
2.3.1.1 Todoist . . . . .	10
2.3.1.2 Notion . . . . .	10
2.3.1.3 Microsoft Project . . . . .	11
2.3.1.4 Trello . . . . .	13
2.4 Μεθοδολογίες . . . . .	13
2.4.1 Διάγραμμα Γκαντ . . . . .	14
2.4.2 Program evaluation and review technique (PERT) . . . . .	15
2.4.3 Kanban . . . . .	17
2.5 Διαχείριση εργασιών στο πανεπιστήμιο . . . . .	18
2.5.1 Προβλήματα διαχείρισης που αντιμετωπίζουν οι φοιτητές . . . . .	19
2.5.2 Χαρακτηριστικά που οι φοιτητές θα επιθυμούσαν σε μια εφαρμογή	19
<b>3 Low-Code</b>	<b>21</b>
3.1 Τι είναι ο χαμηλός κώδικας . . . . .	21

3.1.1	Οπτικός προγραμματισμός (visual programming) . . . . .	22
3.1.2	Καθόλου κώδικας (no-code) . . . . .	24
3.1.3	Η έννοια του προγραμματιστή πολίτη (citizen developer) . . . . .	24
3.1.3.1	Διαφορά με τους παραδοσιακούς προγραμματιστές . . . . .	26
3.1.3.2	Χαρακτηριστικά των προγραμματιστών πολιτών . . . . .	26
3.2	Πώς φτάσαμε στον χαμηλό κώδικα . . . . .	27
3.2.1	Computer-Aided Software Engineering (CASE) . . . . .	29
3.2.2	Model-driven Architecture (MDA) . . . . .	31
3.2.3	Προγενέστερα λογισμικά οπτικού προγραμματισμού . . . . .	32
3.3	Πλατφόρμες Ανάπτυξης Εφαρμογών σε Low-Code (LCDP) . . . . .	33
3.3.1	Χαρακτηριστικά και δομή των LCDP . . . . .	35
3.3.2	Διαδικασία ανάπτυξης στα LCDP . . . . .	37
3.3.3	Παραδείγματα από (LCDP) . . . . .	38
3.3.3.1	Mendix . . . . .	38
3.3.3.2	OutSystems . . . . .	38
3.3.3.3	Power Apps . . . . .	39
<b>4</b>	<b>Mendix</b>	<b>41</b>
4.1	Τι είναι το Mendix; . . . . .	41
4.2	To Mendix Studio . . . . .	42
4.2.1	Περιβάλλον ανάπτυξης . . . . .	43
4.2.1.1	Επιλογές για deployment . . . . .	44
4.3	Δομή εφαρμογών του Mendix . . . . .	44
4.3.1	Modules . . . . .	44
4.3.1.1	To module App . . . . .	45
4.3.1.2	To module System . . . . .	48
4.3.2	Έγγραφα . . . . .	48
4.3.3	Σελίδες . . . . .	48
4.3.3.1	Layout . . . . .	49
4.3.3.2	Widgets . . . . .	50
4.3.3.3	Εμφάνιση σελίδων . . . . .	51
4.3.3.4	Παράδειγμα δομής σελίδας . . . . .	53
4.3.4	Microflows . . . . .	54
4.3.5	Εκφράσεις . . . . .	56
4.3.6	Domain Model . . . . .	56
4.3.6.1	Οντότητες . . . . .	57
4.3.7	Dashboard εφαρμογής . . . . .	60
<b>5</b>	<b>Γλοποίηση εφαρμογής</b>	<b>61</b>
5.1	Mockups και σχεδιαστική προσέγγιση . . . . .	61
5.2	Παρουσίαση της εφαρμογής . . . . .	63

5.3 Δομή της εφαρμογής . . . . .	71
5.3.1 Module Administrator . . . . .	72
5.3.1.1 Domain model του Administrator . . . . .	74
5.3.1.2 Σελίδες του Administrator . . . . .	74
5.3.1.3 Microflows του Administrator . . . . .	79
5.3.2 Module TaskManager . . . . .	83
5.3.2.1 Domain model του TaskManager . . . . .	84
5.3.2.2 Σελίδες του TaskManager . . . . .	85
5.3.2.3 Microflows του TaskManager . . . . .	95
5.3.3 Module UniTaskDesignSystem . . . . .	105
5.3.3.1 Layouts του UniTaskDesignSystem . . . . .	105
5.3.3.2 Styling του UniTaskDesignSystem . . . . .	106
<b>6 &lt;Συμπεράσματα - Προεκτάσεις&gt;</b>	<b>107</b>

# **Κεφάλαιο 1**

## **Εισαγωγή**

- 1.1 Σημασία του προβλήματος**
- 1.2 Στόχοι της εργασίας**
- 1.3 Μεθοδολογία προσέγγισης**
- 1.4 Διάρθρωση της διπλωματικής εργασίας**

## Κεφάλαιο 2

# 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.

### 2.1 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.

### 2.1.1 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. [16]

### 2.1.2 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. [16]

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.

### 2.1.3 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 εφαρμογές<sup>1</sup> (π.χ. Todoist [43], Notion [48], 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

<sup>1</sup>Θα αναφερθούμε πιο εκτεταμένα σε ψηφιακά εργαλεία στην ενότητα 2.3.1

εργαζομένους. Ένα από τα πλέον δημοφιλή εργαλεία που χρησιμοποιούνται σε ομαδικά επαγγελματικά περιβάλλοντα για τον σκοπό αυτό είναι το Trello [23], 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.

#### 2.1.4 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.

## 2.2 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.

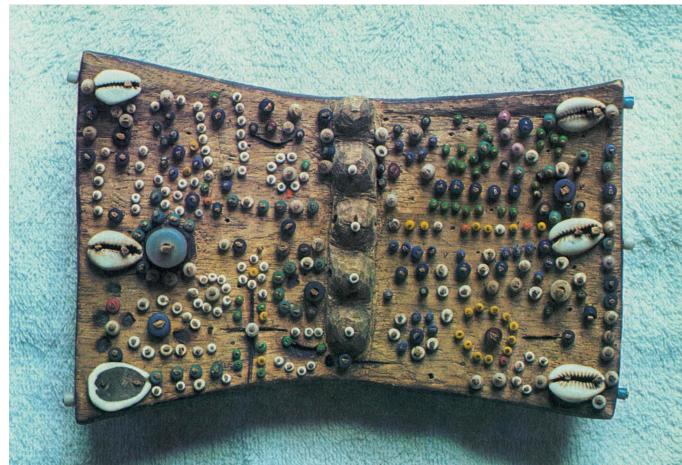
### 2.2.1 Προφορικότητα και μνημονικές συσκευές

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. [15]

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. [34] 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. [32]

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



Σχήμα 2.1: Η συσκευή λουκάσα



Σχήμα 2.2: 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.

### 2.2.2 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 2.3: 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. [35]

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.

### 2.2.3 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 2.4. 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. [17]

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 2.4.



Σχήμα 2.4: Το φράγμα Χούβερ [18]

## 2.3 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.

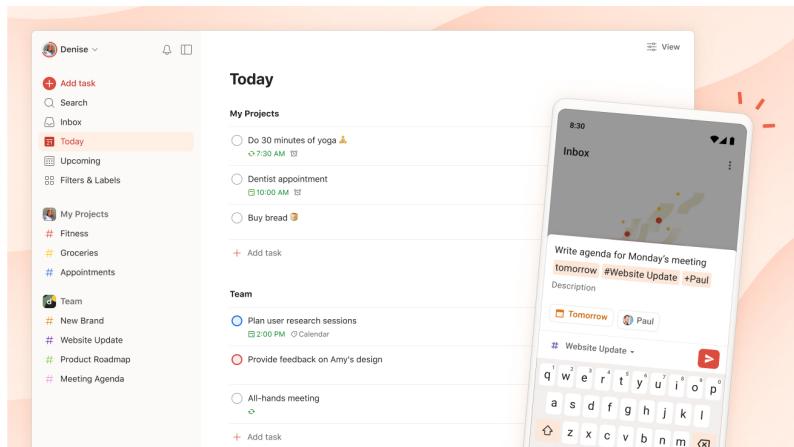
Η τεχνολογική πρόοδος έφερε νέα εργαλεία και λογισμικά που ενίσχυσαν τη συνεργασία μεταξύ ομάδων και τμημάτων, όπως το Microsoft Project και αργότερα οι πλατφόρμες συνεργασίας τύπου Trello και Asana, επέτρεψαν σε ομάδες διαφορετικών γεωγραφικών περιοχών να συνεργάζονται απρόσκοπτα, μειώνοντας τα εμπόδια επικοινωνίας, ενώ πρόσθεσε 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.

### 2.3.1 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.

### 2.3.1.1 Todoist

Αν και έχει χάσει πλέον την πρωτοκαθεδρία του, το Todoist [43] 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 2.5: Η εφαρμογή Todoist.

### 2.3.1.2 Notion

Το Notion [48] 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 εγκυρωτική.

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

The screenshot shows the Notion Projects interface. At the top, there are navigation icons and a search bar. Below that is a header with tabs: Active (selected), Timeline, Mine, and 2 more... There are also buttons for Filter, Sort, and a search icon. A 'New' button is highlighted in blue.

The main area displays two tasks in a timeline view:

- Planning (2)**: Contains a task titled "New Emojis don't render" with a progress bar at 0% from Sep 26 to Oct 23. It involves users Zoe Ludwig and David Tibbitts.
- In Progress (1)**: Contains a task titled "Dogfood new mobile experience" with a progress bar at 100% from Sep 20 to 28. It involves users Alex Hao and Penny Shen.

On the right side, there's a sidebar for "Hidden groups" with categories: Paused (1), Backlog (0), Done (4), and Canceled (0). A "New" button is located below the planning section.

Σχήμα 2.6: 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.

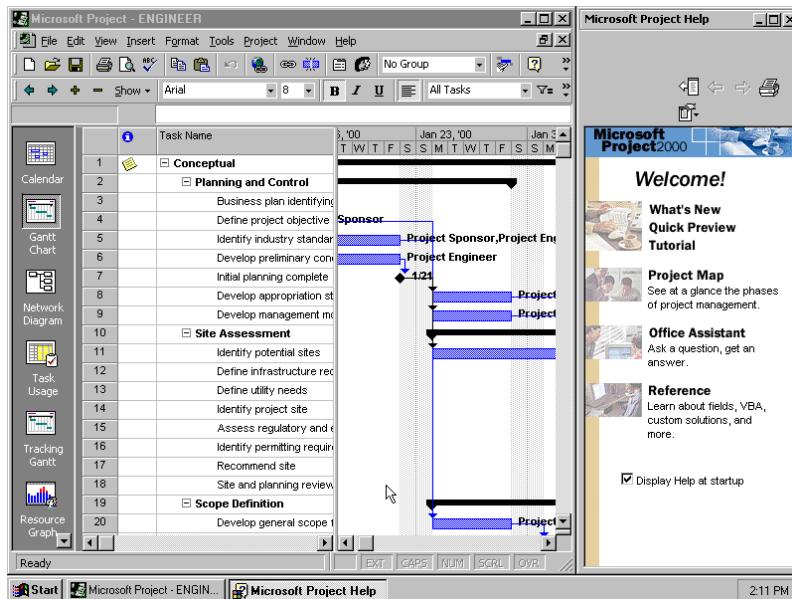
### 2.3.1.3 Microsoft Project



Σχήμα 2.7: Στιγμιότυπο από το Microsoft Project 3.0 (σε DOS) [47]

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.



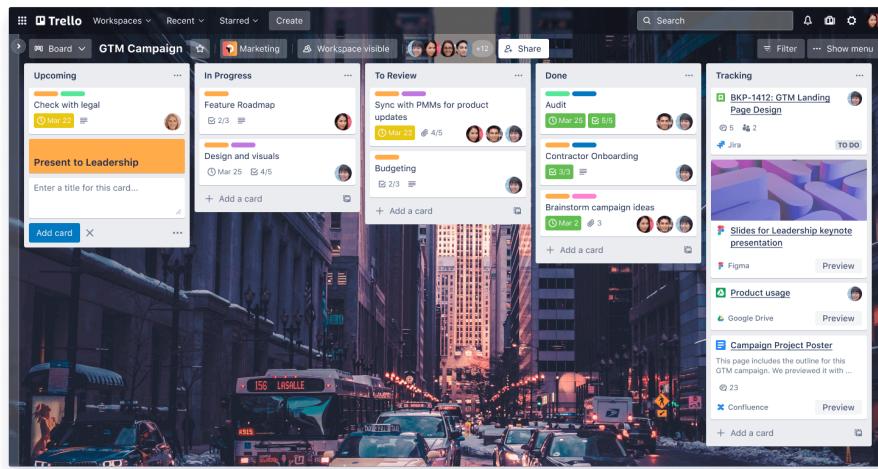
Σχήμα 2.8: Στιγμιότυπο από το Microsoft Project 2000 [47]

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.

### 2.3.1.4 Trello



Σχήμα 2.9: Στιγμιότυπο του Trello

Το Trello [23] είναι μια πλατφόρμα διαχείρισης εργασιών που βασίζεται στους πίνακες Kanban (θα αναλυθούν στην ενότητα 2.4), 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.

## 2.4 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.

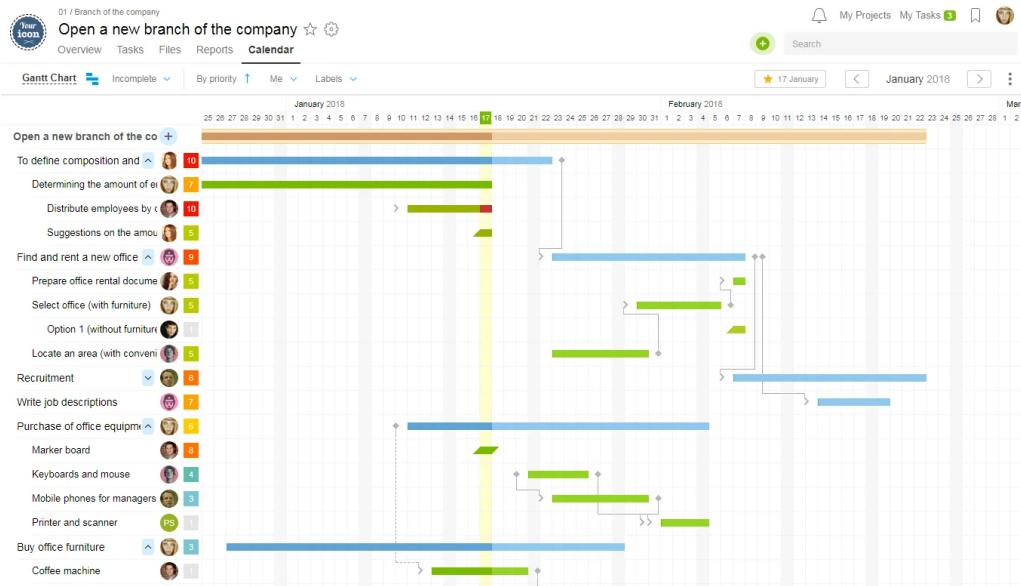
#### 2.4.1 Διάγραμμα Γκαντ

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. [49]



Σχήμα 2.10: Παράδειγμα διαγράμματος Γκαντ

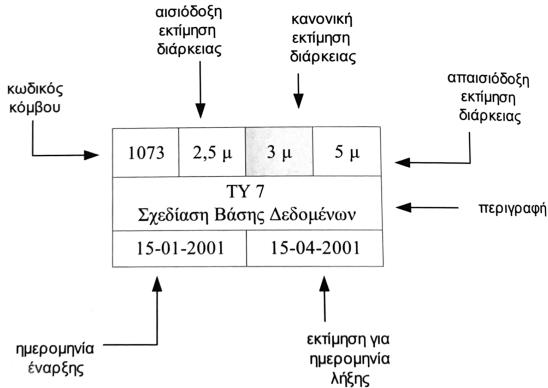
## 2.4.2 Program evaluation and review technique (PERT)

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.

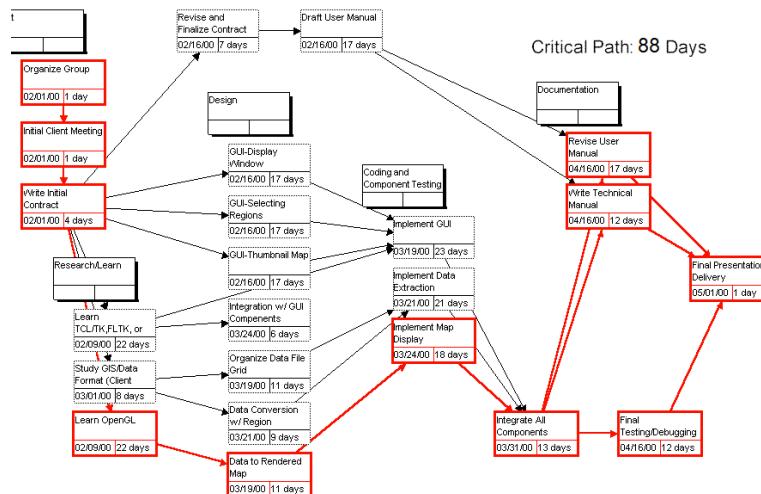
Το PERT βασίζεται στην εκτίμηση του χρόνου για κάθε δραστηριότητα χρησιμοποιώντας τρεις παραμέτρους: τον αισιόδοξο χρόνο (ο ελάχιστος χρόνος ολοκλήρωσης), τον πιο πιθανό χρόνο και τον απαισιόδοξο χρόνο (ο μέγιστος χρόνος ολοκλήρωσης). Με βάση αυτές τις εκτιμήσεις, υπολογίζεται ο αναμενόμενος χρόνος για 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 2.11: Παράδειγμα κόμβου σε διάγραμμα PERT [49]

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 2.12: 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. [22]

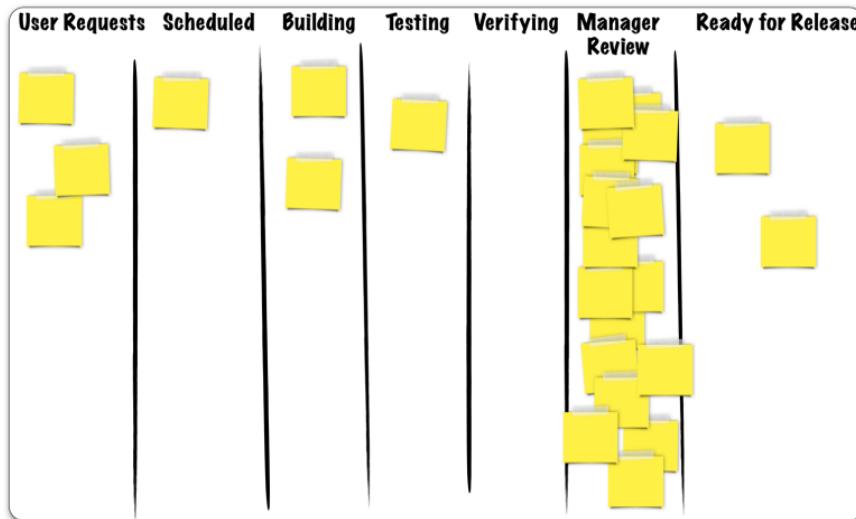
### 2.4.3 Kanban

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 [23], το Jira [3] και το Asana [2], τα οποία προσφέρουν επιπλέον δυνατότητες όπως η αυτόματη ενημέρωση και η συνεργασία σε πραγματικό χρόνο.

TO DO	IN PROGRESS	DONE
Implement feedback collector NUC-205	Update T&C copy with v1.9 from the writers guild in all products that have cross country compliance NUC-213	Quick booking for accomodations - web NUC-336
Bump version for new API for billing NUC-206	Bump feedback icon version NUC-214	Fluid booking on tablets NUC-343
Add NPS feedback to wallboard NUC-208	Tech spike on new stripe integration with paypal NUC-215	Shopping cart purchasing error - quick fix required. NUC-354
Add analytics events to pricing page NUC-209	Change phone number field type to 'phone' NUC-217	
Resize the images for the upcoming campaign NUC-210		

Σχήμα 2.13: Ένας Kanban πίνακας στο Jira



Σχήμα 2.14: Ένα παράδειγμα συμφορήσεων [41]

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] [41]

## 2.5 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.

### 2.5.1 Προβλήματα διαχείρισης που αντιμετωπίζουν οι φοιτητές

Σε έρευνα [11] που διεξήχθη στο Πανεπιστήμιο του Τσουκουμπα 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 [45] 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.

### 2.5.2 Χαρακτηριστικά που οι φοιτητές θα επιθυμούσαν σε μια εφαρμογή

Σε έρευνα [45] 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 3

## 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.

### 3.1 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.” [46]*

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 3.1.3), 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 3.3. 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. [19] [40] [42]

### 3.1.1 Οπτικός προγραμματισμός (visual programming)

Στο σχήμα 3.1 παρατίθενται δύο παραδείγματα από την παραδοσιακή ανάπτυξη εφαρμογών και την ανάπτυξη εφαρμογών σε low-code στην πλατφόρμα Mendix.

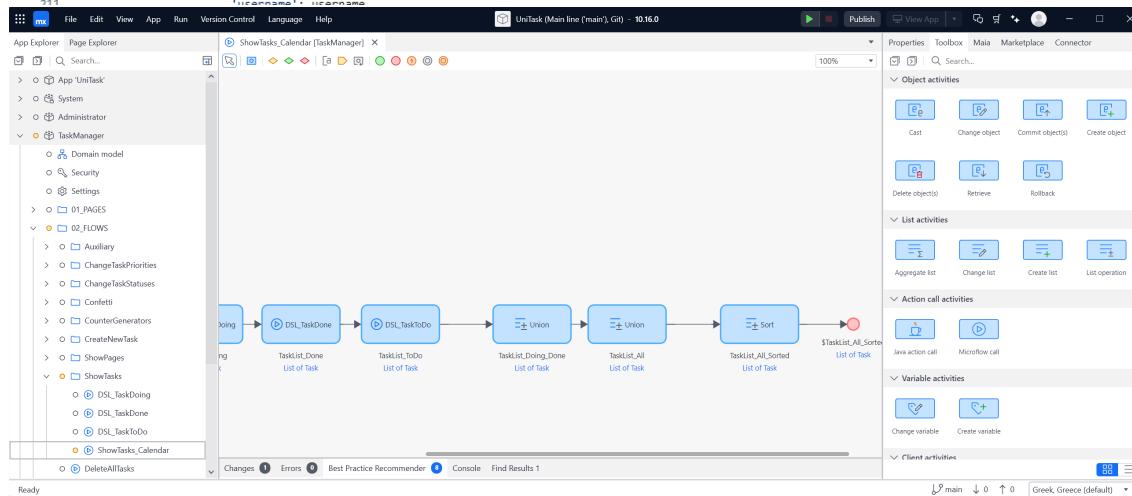
Στο γραφικό περιβάλλον, ο προγραμματισμός γίνεται σε ένα διάγραμμα ροής με drag-and-drop επαναχρησιμοποιούμενα στοιχεία. Αυτή η προσέγγιση διευκολύνει την ανάπτυξη εφαρμογών, καθώς επιτρέπει στους χρήστες να επικεντρωθούν στη λογική της εφαρμογής, χωρίς να χρειάζεται να ασχοληθούν με τη σύνταξη και τις λεπτομέρειες του κώδικα. Στο σχήμα 3.1, 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 (υψηλός κώδικας), 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 3.1: 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 [20]. 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.

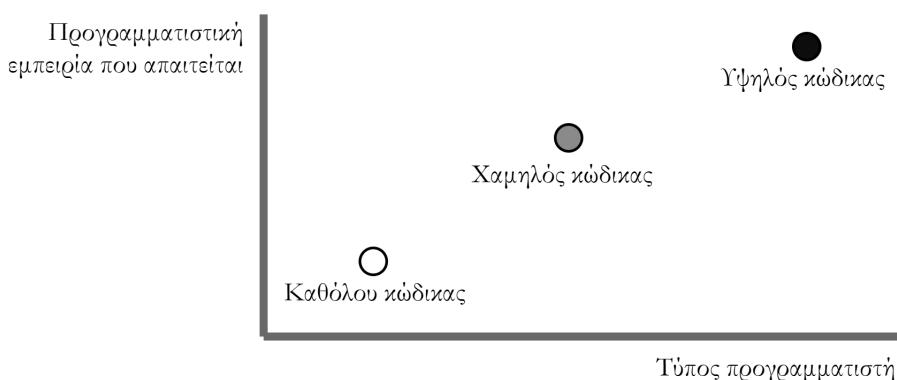
Τέλος, είναι σημαντικό να αναφερθεί πως η οπτική προσέγγιση επιτρέπει την ταχύτερη ανίχνευση σφαλμάτων και την πιο αποτελεσματική συνεργασία μεταξύ των μελών μιας ομάδας, καθώς ένα διάγραμμα ροής είναι άμεσα κατανοητό από όλους τους εμπλεκόμενους, ανεξαρτήτως τεχνικού υποβάθρου.

### 3.1.2 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. [40]



Σχήμα 3.2: Σύγκριση ανάμεσα στην προγραμματιστική εμπειρία των χρηστών και την προγραμματιστική προσέγγιση που χρησιμοποιούν [40]

### 3.1.3 Η έννοια του προγραμματιστή πολίτη (citizen developer)

Ο προγραμματιστής πολίτης είναι ένας χρήστης που αναπτύσσει εφαρμογές σε συνεργασία με τους προγραμματιστές υψηλού επιπέδου, 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 [40]:

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.

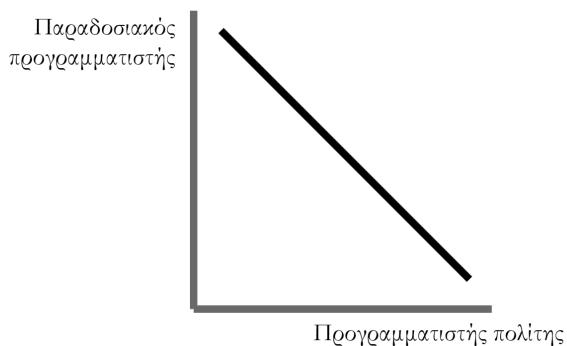
Η Gartner, μια κορυφαία διεθνής εταιρία ερευνών και συμβουλευτικών υπηρεσιών που εξειδικεύεται στους τομείς της τεχνολογίας, της επιχειρηματικής στρατηγικής και της καινοτομίας, υπογραμμίζει τη δυναμική αυτής της εξέλιξης. Σύμφωνα με έκθεσή της, προβλέπεται πως «έως το 2023, ο αριθμός των ενεργών προγραμματιστών πολιτών σε μεγάλες επιχειρήσεις θα είναι τουλάχιστον τετραπλάσιος από τον αριθμό των παραδοσιακών προγραμματιστών» [31]. Επίσης, σύμφωνα με φετινή έκθεσή της αναφέρεται πως «μέχρι το 2029, οι πλατφόρμες ανάπτυξης εφαρμογών low-code για επιχειρήσεις (Enterprise LCAPs) θα χρησιμοποιούνται για την ανάπτυξη εφαρμογών κρίσιμης σημασίας στο 80% των επιχειρήσεων παγκοσμίως, σε σύγκριση με το 15% που είναι το 2024». [13]

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» [36]. 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.

### 3.1.3.1 Διαφορά με τους παραδοσιακούς προγραμματιστές

Οι παραδοσιακοί προγραμματιστές (προγραμματιστές υψηλού κώδικα), 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 3.3.

Ποιος θα αναπτύξει την εφαρμογή;



Σχήμα 3.3: Ποιος θα φτιάξει την εφαρμογή; [40]

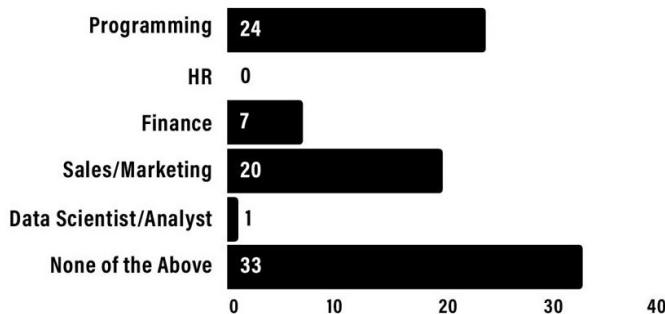
Παρότι οι προγραμματιστές πολίτες μπορούν και συνεισφέρουν στην υλοποίηση απλών εφαρμογών, παραμένει ανέφικτο για αυτούς να αναλάβουν τον σχεδιασμό και την ανάπτυξη πολύπλοκων συστημάτων που απαιτούν σχεδιαστική γνώση και εμπειρία. Αυτή η πραγματικότητα δημιουργεί έναν φυσικό διαχωρισμό ρόλων. Με ένα 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 διεκπεραιώθουν 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. [38]

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

### 3.1.3.2 Χαρακτηριστικά των προγραμματιστών πολιτών

Σε μια έρευνα που πραγματοποιήθηκε από την εταιρεία QuickBase [30], 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.



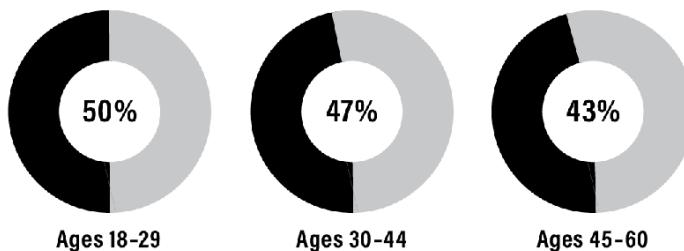
Σχήμα 3.4: Επαγγελματικό υπόβαθρο προγραμματιστών πολιτών. [40]

Το σχήμα 3.4 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 3.5 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.

### 3.2 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 3.5: Ηλικιακή κατανομή προγραμματιστών πολιτών. [44]

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)<sup>1</sup> 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)<sup>2</sup>, 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-

<sup>1</sup>Πρόκειται 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.

<sup>2</sup>Είναι 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, 37]

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). [33]

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]

### 3.2.1 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, 20, 29]

### 3.2.2 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) [24], 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, 37, 39]

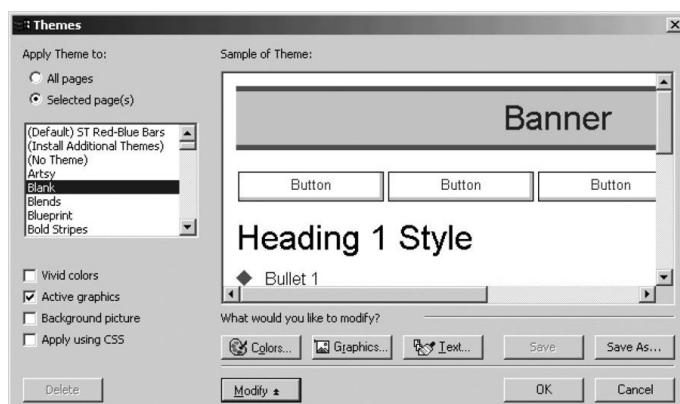
Εν τέλει, η αρχιτεκτονική που βασίζεται σε μοντέλα έχει φέρει μια αλλαγή στον τρόπο σκέψης των προγραμματιστών και μηχανικών λογισμικού. Εισήγαγε μια κουλτούρα που βασίζεται στην αφαιρετικότητα, τον σωστό διαχωρισμό των χαρακτηριστικών και την αυτοματοποίηση. Με αυτόν τον τρόπο, οι προγραμματιστές μπορούν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

### 3.2.3 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 3.6: Το γραφικό περιβάλλον του Microsoft FrontPage

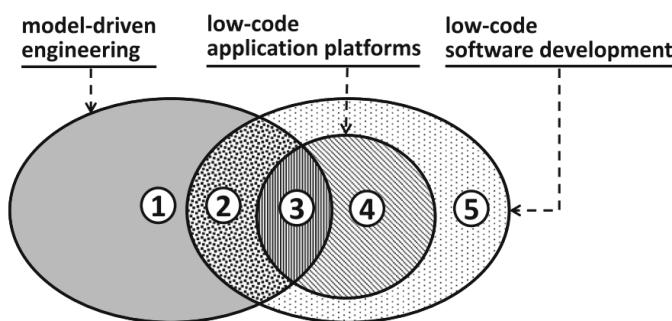
Τα LCDPs που ακολούθησαν πήραν αυτές τις αρχές και τις εξέλιξαν, ενσωματώνοντας προγραμμένα χαρακτηριστικά όπως η ευελιξία στη σύνδεση διαφορετικών

συστημάτων, η χρήση cloud τεχνολογιών ή η υποστήριξη πολλαπλών πλατφορμών, δημιουργώντας εν τέλει εργαλεία που είναι ισχυρά, αλλά ταυτόχρονα προσιτά. [40]

### 3.3 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)<sup>3</sup>, 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]

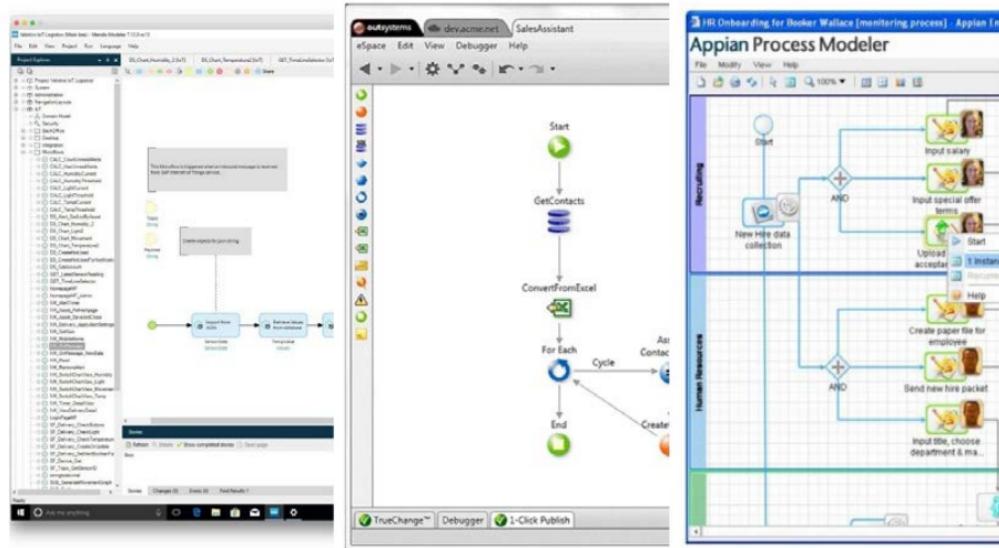


Σχήμα 3.7: Σύγκριση μεταξύ της μηχανικής που βασίζεται σε μοντέλα και των Low-Code πλατφορμών. [37]

Το σχήμα 3.7 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-

<sup>3</sup>Εναλλακτικές 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 3.8: Στιγμιότυπα από LCDP από αριστερά προς τα δεξιά: Mendix, OutSystems, Appian [19]

**Μια Πλατφόρμα Ανάπτυξης Εφαρμογών σε Low-Code (LCDP)** είναι μια πλατφόρμα ανάπτυξης λογισμικού που συνήθως λειτουργεί ως Platform-as-service (PaaS) βασιζόμενη στο cloud και έχει σχεδιαστεί για να υποστηρίζει την ταχεία ανάπτυξη, εκτέλεση και διαχείριση εφαρμογών με ελάχιστο ή καθόλου ανάγκη για παραδοσιακό κώδικα και δομημένο προγραμματισμό. Αντίθετα, παρέχεται περιβάλλον με οπτικές αφαιρέσεις (visual abstractions) και χρησιμοποιείται προγραμματισμός με υψηλό επίπεδο αφαιρέσεων χρησιμοποιώντας μοντέλα και μεταδεδομένα.

Η αποδοτική ανάπτυξη λογισμικού με χαμηλό κόστος και ελάχιστη προσπάθεια αποτελεί το βασικό στόχο των LCDP. 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 LCDP έχει θύχει θετικής αποδοχής από τη βιομηχανία με πολλές εταιρείες και οργανισμοί να έχουν ενσωματώσει αυτές τις πλατφόρμες στις στρατηγικές

τους για την ανάπτυξη λογισμικού και την υιοθέτησή τους συνεχώς να αυξάνεται. [4, 5, 38]

Οι τεχνολογίες των LCDP δεν είναι ούτε νέες, ούτε καινοτόμες. Αντίθετα, περιλαμβάνουν γνωστά εργαλεία και χαρακτηριστικά που χρησιμοποιούνται κατά κόρον εδώ και χρόνια σε κάποια μορφή. Το πλεονέκτημα όμως των LCDP είναι ότι συνδυάζουν αυτά τα εργαλεία και χαρακτηριστικά σε ένα ενιαίο περιβάλλον προσφέροντας μια ολοκληρωμένη λύση για την ανάπτυξη λογισμικού. Αυτό επιτρέπει στους χρήστες να αναπτύσσουν εφαρμογές με μεγάλη ταχύτητα, ενώ ταυτόχρονα διατηρούν την ευελιξία και την προσαρμοστικότητα που απαιτούνται από τις σύγχρονες εφαρμογές.

### 3.3.1 Χαρακτηριστικά και δομή των LCDP

Επιγραμματικά κάποια κοινά χαρακτηριστικά για τα οποία διαχρίνονται οι Πλατφόρμες Ανάπτυξης Εφαρμογών σε Low-Code είναι τα εξής:

- **Γραφικός σχεδιαστής (GUI Designer):** Ένα από τα πιο προφανή χαρακτηριστικά των LCDP είναι το γραφικό περιβάλλον χρήστη, ο οποίος παρέχει υποστήριξη για την προσαρμογή του GUI σε διαφορετικά περιβάλλοντα (υπολογιστές, κινητά κ.α.). Περιλαμβάνονται drag-and-drop εργαλεία, μηχανές αποφάσεων για τη μοντελοποίηση πολύπλοκης λογικής, κατασκευαστές φορμών (form builders) και άλλα.
- **Μοντελοποίηση δομών δεδομένων:** Σχεδόν πάντα εφαρμόζεται μια παραλλαγή του τυπικού μοντέλου οντοτήτων-συσχετίσεων (ER model). Σε ορισμένες περιπτώσεις οι δομές δεδομένων ορίζονται μόνο μέσω διαλόγων ή λιστών από το περιβάλλον διεπαφής του χρήστη. Συχνά υπάρχει πρόσβαση σε εξωτερικές πηγές δεδομένων χρησιμοποιώντας API και χρήση προτύπων όπως το JDBC<sup>4</sup> και συνδέσμους (connectors) για άλλους τύπους αρχείων και συστημάτων. Τα δεδομένα σχεδιάζονται ώστε να αποθηκεύονται είτε στο εσωτερικό σύστημα βάσεων δεδομένων της πλατφόρμας είτε σε εξωτερικές πηγές.
- **Ροή προγράμματος:** Χρήση απλών γλωσσικών εκφράσεων για κανόνες απόφασης και καθορισμού συνθηκών ροής προγράμματος. Περιλαμβάνονται βιβλιοθήκες με λειτουργίες γενικού τύπου (όπως μαθηματικές συναρτήσεις) ή την ενσωμάτωση εσωτερικών λειτουργιών (για παράδειγμα REST υπηρεσίες).
- **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

<sup>4</sup>Java Database Connectivity: Πρότυπο για πρόσβαση σε βάσεις δεδομένων μέσω της Java.

τις εφαρμογές πιο ευέλικτες και προσαρμόσιμες στις ανάγκες των προγραμματιστών ή των τελικών χρηστών.

- Εύκολη διάθεση και έλεγχος έκδοσης: Ορισμένες πλατφόρμες η εφαρμογή να γίνεται deploy απομακρυσμένα σε κάποιον διακομιστή, ενώ άλλες το επιτρέπουν στο εκάστοτε μηχάνημα. Κάθε LCDP ενσωματώνει χαρακτηριστικά Dev Ops όπως το version control μέσω Git, διευκολύνοντας την παρακολούθηση αλλαγών και την αναίρεση ανεπιθύμητων ενεργειών. [4]

Περνώντας σε παραπάνω λεπτομέρειες, αρχιτεκτονικά οι Πλατφόρμες Ανάπτυξης Λογισμικού σε Low-Code διαχωρίζονται σε τέσσερα επίπεδα: Το υψηλότερο επίπεδο, το *Επίπεδο Εφαρμογής* (Application Layer) περιλαμβάνει το γραφικό περιβάλλον με το οποίο αλληλεπιδρούν οι χρήστες. Εκεί βρίσκονται τα εργαλεία και τα widgets που χρησιμοποιούνται για τη δημιουργία των σελίδων της εφαρμογής. Είναι το μέρος που οι χρήστες μπορούν να καθορίσουν τη συμπεριφορά της εφαρμογής, να συνδέουν δεδομένα από εξωτερικές πηγές και να τα επεξεργαστούν και περιλαμβάνονται μηχανισμοί επαλήθευσης ταυτότητας και εξουσιοδότησης. Για τη σύνδεση των εξωτερικών πηγών μέσω των μηχανισμών επαλήθευσης ταυτότητας και των API (για παράδειγμα DropBox ή Git) χρησιμοποιείται το *Επίπεδο Ενσωμάτωσης Υπηρεσιών* (Service Integration Layer). Αμέσως χαμηλότερα, το *Επίπεδο Ενσωμάτωσης Δεδομένων* (Data Integration Layer) επιτρέπει την ομοιόμορφη διαχείρισης, επεξεργασία και ενοποίηση δεδομένων, ακόμα αν προέρχονται από ετερογενείς πηγές. Το χαμηλότερο επίπεδο είναι το *Επίπεδο Διανομής* (Deployment Layer) στο οποίο η εκάστοτε εφαρμογή πακετάρεται και διανέμεται στο cloud. [38]



Σχήμα 3.9: Αρχιτεκτονικός σχεδιασμός των LCDP [38]

Επεκτείνοντας την αρχιτεκτονική των τεσσάρων προαναφερθέντων επιπέδων, μπορούμε να ομαδοποιήσουμε τα κύρια στοιχεία που συνθέτουν οποιαδήποτε Πλατφόρμα Ανάπτυξης Λογισμικού σε Low-Code (LCDP) σε τρία μέρη: Το πρώτο μέρος περιλαμβάνει τον μοντελοποιητή εφαρμογών (application modeler), το βασικό μηχανισμό που επιτρέπει στους χρήστες να δημιουργούν και να διαμορφώνουν εφαρμογές μέσω μιας γραφικής διεπαφής, το δεύτερο μέρος αφορά την πλευρά του διακομιστή (server-side)

(περιλαμβάνει τον κεντρικό μηχανισμό επεξεργασίας που χειρίζεται και επεξεργάζεται τα δεδομένα και διαχειρίζεται την ασφάλεια και επικοινωνία μεταξύ των εξωτερικών πηγών δεδομένων), ενώ το τρίτο μέρος επικεντρώνεται στις εξωτερικές υπηρεσίες (external services) που ενσωματώνονται στην πλατφόρμα.

Υπάρχει υποστήριξη και για SQL και για NoSQL βάσεις δεδομένων<sup>5</sup>. Όποιο και να είναι το είδος των δεδομένων, οι προγραμματιστές των εφαρμογών δεν χρειάζεται να ασχολούνται με τη διαχείριση των δεδομένων σε χαμηλό επίπεδο καθώς όλες οι διαδικασίες δημιουργίας, συντονισμού και διαχείρισης των δεδομένων πραγματοποιούνται στο back-end της πλατφόρμας. Επίσης, συνήθως παρέχονται αποθετήρια (repositories) για τη διαχείριση των εκδόσεων της εφαρμογής (version control), διευκολύνοντας τη συνεργασία μεταξύ των ομάδων ανάπτυξης και επιτρέποντας την παρακολούθηση και την ανάκτηση προηγούμενων εκδόσεων της εφαρμογής. Για να υποστηριχθεί η συνεργατική ανάπτυξη, οι LCDP ενσωματώνουν πλήθος εργαλείων και λειτουργιών που διευκολύνουν τη χρήση μεθοδολογιών ανάπτυξης λογισμικού, όπως το Agile, το Kanban και το Scrum. Αυτές οι μεθοδολογίες επιτρέπουν στις ομάδες ανάπτυξης να εργάζονται σε μικρότερα, διαχειρίσιμα κομμάτια του έργου, ενώ παράλληλα διευκολύνουν την επικοινωνία και τη συνεργασία. Μέσω αυτών των λειτουργιών, οι πλατφόρμες προσφέρουν ένα περιβάλλον που υποστηρίζει τη συνεχιζόμενη παράδοση (continuous delivery) και τη γρήγορη προσαρμογή στις αλλαγές των απαιτήσεων, κάτι που είναι καίριο για την ανάπτυξη σύγχρονων επιχειρηματικών εφαρμογών.

### 3.3.2 Διαδικασία ανάπτυξης στα LCDP

Για την ανάπτυξη μιας εφαρμογής μέσω πλατφορμών χαμηλού κώδικα (Low-Code Development Platforms – LCDPs), συνήθως ακολουθείται μια σειρά βημάτων που εξασφαλίζουν τη σωστή δομή και λειτουργικότητα της εφαρμογής. Το πρώτο βήμα περιλαμβάνει τη μοντελοποίηση δεδομένων. Σε αυτή τη φάση, οι χρήστες χρησιμοποιούν το γραφικό περιβάλλον της πλατφόρμας για να διαμορφώσουν τις οντότητες της εφαρμογής, τις σχέσεις μεταξύ των οντοτήτων και να ορίσουν περιορισμούς και εξαρτήσεις. Αφού ολοκληρωθεί η μοντελοποίηση, ακολουθεί ο σχεδιασμός της διεπαφής χρήστη. Σε αυτό το στάδιο, οι χρήστες διαμορφώνουν φόρμες και σελίδες που θα χρησιμοποιηθούν για την παρουσίαση της εφαρμογής. Παράλληλα, ορίζονται οι ρόλοι χρηστών και οι μηχανισμοί ασφαλείας που θα ισχύουν για τις οντότητες, τις φόρμες και τις σελίδες. Στη συνέχεια, γίνεται ο καθορισμός των κανόνων λογικής και των ροών εργασίας. Αυτό περιλαμβάνει τη διαχείριση ροών μεταξύ φορητών ή σελίδων που απαιτούν διαφορετικές λειτουργίες, οι οποίες υλοποιούνται μέσω οπτικών ροών εργασίας. Ακολουθεί η ενσωμάτωση εξωτερικών υπηρεσιών μέσω APIs τρίτων. Στο

<sup>5</sup>Οι SQL βάσεις δεδομένων οργανώνουν τα δεδομένα σε πίνακες με γραμμές και στήλες όπου κάθε γραμμή αντιπροσωπεύει μια εγγραφή και κάθε στήλη ένα πεδίο δεδομένων. Οι NoSQL βάσεις δε χρησιμοποιούν αυτή την παραδοσιακή σχέση πίνακα-γραμμής-στήλης αλλά υποστηρίζουν διάφορους τύπους αποθήκευσης δεδομένων όπως έγγραφα, κλειδιά-τιμές ή γραφήματα και είναι πιο ευέλικτες στην επεξεργασία μη δομημένων δεδομένων.

τελευταίο στάδιο γίνεται η διάθεση της εφαρμογής, κάτι που μπορεί να γίνει στις περισσότερες πλατφόρμες με λίγα μόνο κλικ.

### 3.3.3 Παραδείγματα από (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. Κάποιες από τις πιο δημοφιλείς πλατφόρμες ανάπτυξης εφαρμογών σε Low-Code είναι η πλατφόρμα Mendix, η OutSystems και η σουίτα Power Apps της Microsoft.

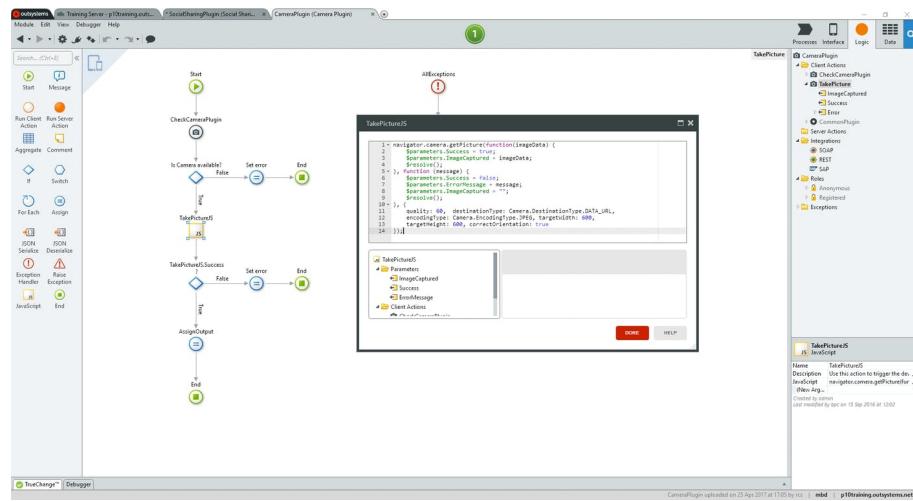
#### 3.3.3.1 Mendix

Το Mendix αποτελεί μια από τις κορυφαίες πλατφόρμες ανάπτυξης λογισμικού χαμηλού κώδικα, με drag-and-drop δυνατότητες και δυνατότητα για συνεργασία σε πραγματικό χρόνο με συναδέλφους με ενσωματωμένη χρήση Agile μεθόδων. Περιλαμβάνει γραφικό περιβάλλον που βοηθάει στην επαναχρησιμοποίηση διαφόρων στοιχείων κάτι που επιταχύνει τη διαδικασία ανάπτυξης, από τη ρύθμιση του μοντέλου δεδομένων ως τον ορισμό των διεπαφών χρήστη. Η πλατφόρμα υποστηρίζει ανάπτυξη εφαρμογών για web, mobile και PWA (Progressive Web Apps) και παρέχεται cloud υποδομή για άμεση διάθεση της εφαρμογής. Θα αναφερθούμε αναλυτικά στο Mendix στο κεφάλαιο 4.

#### 3.3.3.2 OutSystems

Η OutSystems είναι μια άλλη δημοφιλής PaaS βασισμένη στο cloud πλατφόρμα ανάπτυξης εφαρμογών χαμηλού κώδικα που είναι σχεδιασμένη με γνώμονα την απόδοση, την επεκτασιμότητα και την υψηλή διαθεσιμότητα. Αποτελεί ένα από τα κορυφαία ονόματα ειδικά στην ανάπτυξη mobile εφαρμογών, μαζί εταιρείες όπως η IBM, και η Gartner την χαρακτήρισε ως «օραματιστής» ανάμεσα σε 36 διαφορετικές πλατφόρμες ανάπτυξης mobile εφαρμογών. Τα πλεονεκτήματά της περιλαμβάνουν την προσέγγισή του που βασίζεται σε μοντέλα (model-driven approach) και τη συμπερίληψη μιας ποικιλίας APIs. Παρόλα αυτά, επισημάνθηκαν και κάποιες προειδοποιήσεις, όπως η έλλειψη ελέγχου στον παραγόμενο κώδικα, η ασυμβατότητα με τις παραδοσιακές μεθόδους ανάπτυξης και η απουσία ορισμένων επιλογών ανάπτυξης, τα οποία ενδέχεται να αποθαρρύνουν πιθανούς πελάτες από τη χρήση της πλατφόρμας. [12]

Η πλατφόρμα αποτελείται από δύο μέρη: έναν διακομιστή και την αντίστοιχη εφαρμογή για υπολογιστές. Η εγκατάσταση του διακομιστή μπορεί να επεκταθεί με επιπλέον υπηρεσίες και αποθετήρια (repositories), αν αυτά είναι απαραίτητα για την ανάπτυξη των εφαρμογών. Για την ανάπτυξη των εφαρμογών μπορεί να χρησιμοποιηθούν περιβάλλοντα όπως το Microsoft .NET Stack ή το WebLogic της Oracle, και ο πηγαίος κώδικας των εφαρμογών παράγεται είτε σε C# είτε σε Java, επιτρέποντας τη



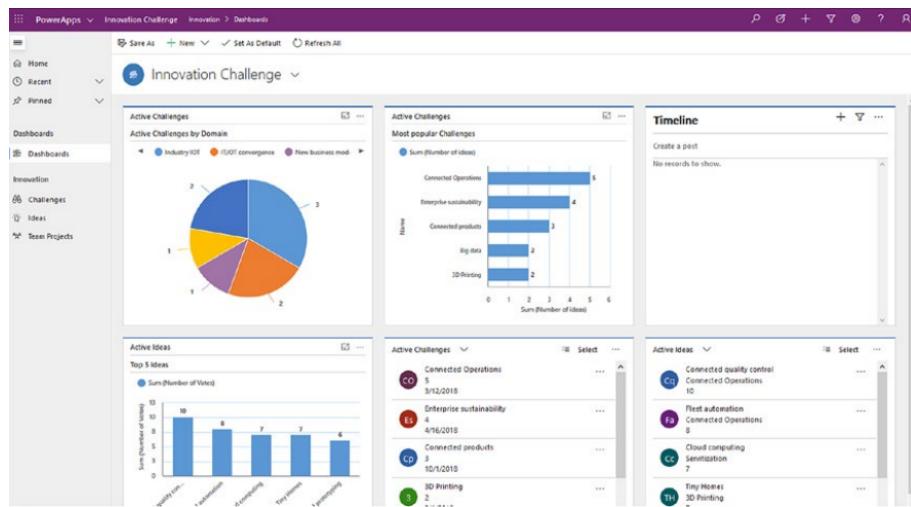
Σχήμα 3.10: Στιγμιότυπο από τη πλατφόρμα OutSystems

χρήση μηχανών όπως Common Language Runtime ή Java Virtual Machine. Η εφαρμογή της OutSystems, το Service Studio, δίνει εύκολη πρόσβαση σε όλες τις λειτουργίες της πλατφόρμας, όπως η μοντελοποίηση της διεπαφής χρήστη, τις βάσεις δεδομένων, SOAP/REST υπηρεσιών, προγραμματιζόμενων εργασιών κ.α. Πέρα από το Service Studio, διατίθεται και το Integration Studio, μια εφαρμογή επέκτασης της πλατφόρμας με πρόσθετη λειτουργικότητα, όπου οι προγραμματιστές μπορούν να ενοποιήσουν εξωτερικές βιβλιοθήκες, υπηρεσίες ή βάσεις δεδομένων με το OutSystems. Οι εφαρμογές Service Studio και το Integration Studio δεν εκτελούν τον κώδικα της πλατφόρμας τοπικά. Συνδέονται απομακρυσμένα στην OutSystems, επομένως δεν υπάρχουν μεγάλες απαιτήσεις σε υπολογιστική ισχύ ή σε διαθέσιμο αποθηκευτικό χώρο, παρά μόνο καλή σύνδεση στο διαδίκτυο, και επιπλέον είναι διαθέσιμες και σε browser μορφή, κάτι που εξαλείφει εντελώς την ανάγκη για εγκατάστασή τους. [14] [28]

### 3.3.3.3 Power Apps

Το **Power Apps** είναι μια άλλη PaaS πλατφόρμα για την ανάπτυξη εφαρμογών σε χαμηλό κώδικα. Σε αντίθεση με πλατφόρμες όπως την OutSystems που επικεντρώνεται στην ανάπτυξη εφαρμογών για καταναλωτές, όπως παιχνίδια ή mobile εφαρμογές ευρείας χρήσης, το Power Apps εστιάζει στην παροχή εργαλείων για τη δημιουργία προσαρμοσμένων λύσεων που ανταποκρίνονται στις εσωτερικές ανάγκες των επιχειρήσεων.

Η Microsoft παρουσίασε το **Power Apps** στα τέλη του 2016, προσφέροντας ένα γραφικό περιβάλλον χρήστη που θύμιζε έντονα εκείνο του Microsoft Excel, διευκολύνοντας έτσι τους χρήστες με βασικές γνώσεις του οικοσυστήματος της εταιρείας να προσαρμοστούν γρήγορα. Από την αρχή, το Power Apps υποστήριζε την ενσωμάτωση δεδομένων από διάφορες πηγές, όπως το SharePoint, το Dynamics 365, οι διακο-



Σχήμα 3.11: Παράδειγμα portal app στο Power Apps

μιστές SQL και εκαποντάδες ακόμη συνδέσεις, ενισχύοντας τη χρησιμότητά του για οργανισμούς με διαφορετικές ανάγκες δεδομένων. Στα επόμενα χρόνια, η πλατφόρμα εξελίχθηκε περαιτέρω με την εισαγωγή ενός νέου τρόπου ανάπτυξης εφαρμογών βασισμένου σε μοντέλα (model-driven development) ο οποίος επέτρεψε τη δημιουργία λειτουργικών διεπαφών χρήστη με ελάχιστη ή και καθόλου παρέμβαση από προγραμματιστές. Το 2019, η Microsoft εισήγαγε τα Power Apps Portals, μια υπηρεσία που επέτρεπε τη δημιουργία ιστοσελίδων. Αυτή η προσθήκη ενίσχυσε σημαντικά τις δυνατότητες του Power Apps, καθώς πλέον οι επιχειρήσεις μπορούσαν να δημιουργήσουν ιστοσελίδες που επιτρέπουν σε χρήστες εκτός της επιχείρησης να έχουν πρόσβαση στις επιχειρηματικές εφαρμογές τους. [21]

## Κεφάλαιο 4

# Mendix

Έχοντας πλέον μια καλή εικόνα για τον ορισμό του χαμηλού κώδικα και των Πλατφόρμων Ανάπτυξης Λογισμικού σε Low-Code, στο παρόν κεφάλαιο, θα επικεντρωθούμε σε μια από αυτές τις πλατφόρμες, το Mendix, η οποία αποτέλεσε το βασικό εργαλείο για την υλοποίηση της εφαρμογής που αναπτύχθηκε στο πλαίσιο της παρούσας διπλωματικής εργασίας.

Σε αυτό το κεφάλαιο, θα περιγραφεί το γραφικό περιβάλλον του Mendix Studio Pro, και θα αναλυθεί η δομή μιας εφαρμογής που αναπτύσσεται στο Mendix. Θα περιγραφούν έννοιες όπως τα modules, τα έγγραφα, τα widgets, οι σελίδες, τα microflows, τα domain models και άλλα. Ο στόχος είναι να κατανοηθεί πλήρως η δομή μιας εφαρμογής στο Mendix, πριν προχωρήσουμε στη διαδικασία υλοποίησης στο επόμενο κεφάλαιο.

### 4.1 Τι είναι το Mendix;

Το Mendix αποτελεί μία από τις πιο διαδεδομένες πλατφόρμες ανάπτυξης λογισμικού που βασίζεται σε χαμηλό κώδικα. 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 αποδοτικά με χαμηλό κόστος. Το Mendix περιλαμβάνει όλα τα οφέλη και τα χαρακτηριστικά των LCDP που έχουν περιγραφεί στην ενότητα 3.3, συμπεριλαμβάνοντας γραφικό περιβάλλον με WYSIWYG GUI σχεδιαστή, drag-and-drop εργαλεία και έτοιμες βιβλιοθήκες, τη χρήση domain models, το εύκολο deployment της εφαρμογής στο cloud, version control μέσω Git, συνεργασία χρησιμοποιώντας Agile μεθοδολογία και άλλα.

Το 2018, το Mendix εξαγοράστηκε από τη 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 των λογισμικών σχεδιασμένων για επιχειρήσεις. Έτσι, το Mendix αποτελεί μία από τις πιο ισχυρές και ευέλικτες λύσεις στην αγορά του low-code προγραμμα-

τισμού, 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. [19]

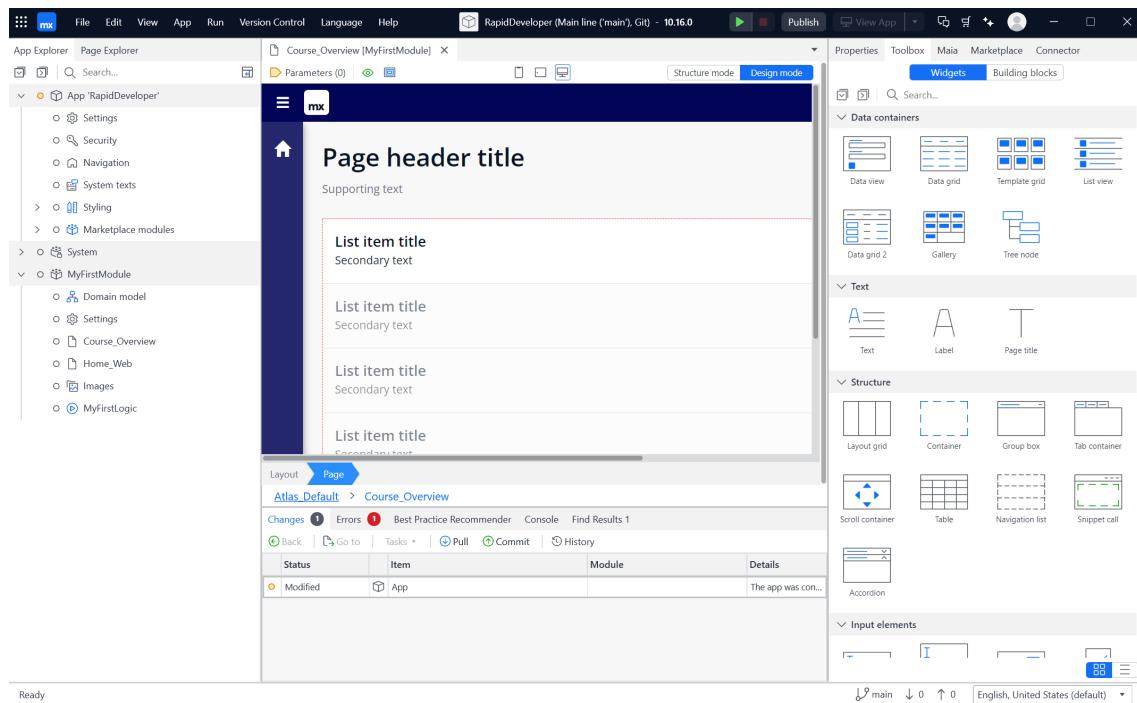
Η Gartner, μία από τις μεγαλύτερες εταιρείες έρευνας και συμβουλευτικής στον κλάδο της τεχνολογίας, χαρακτηρίζει το Mendix ως ηγέτη στην αγορά των πλατφορμών ανάπτυξης λογισμικού για 8 συνεχόμενα χρόνια. (βλ. Σχήμα 4.1). Η κατάταξη αυτή αποδεικνύει την ικανότητα του Mendix να παρέχει λύσεις υψηλής ποιότητας και αξίας στους πελάτες του, καθώς και την ικανότητά του να προσαρμόζεται στις ανάγκες της αγοράς και να προσφέρει συνεχώς καινοτόμες λύσεις. [13] Για αυτούς τους λόγους έχει προτιμηθεί για την υλοποίηση της εφαρμογής που θα παρουσιαστεί στο επόμενο κεφάλαιο.



Σχήμα 4.1: Τεταρτημόριο της Gartner με πλατφόρμες ανάπτυξης λογισμικού [13]

## 4.2 To Mendix Studio

Το Mendix Studio Pro αποτελεί το βασικό εργαλείο ανάπτυξης εφαρμογών της πλατφόρμας Mendix, προσφέροντας στους χρήστες τη δυνατότητα να δημιουργήσουν, να προσαρμόσουν, να δοκιμάσουν και να αναπτύξουν εφαρμογές.



Σχήμα 4.2: Στιγμιότυπο του Mendix Studio Pro.

#### 4.2.1 Περιβάλλον ανάπτυξης

Στο 4.2 παρουσιάζεται το γραφικό περιβάλλον του Mendix Studio Pro με ανοιχτή την εφαρμογή RapidDeveloper<sup>1</sup>.

Μπορούμε να διαχωρίσουμε το γραφικό περιβάλλον σε τέσσερα μέρη. Η μαύρη μπάρα στο πάνω μέρος περιλαμβάνει το βασικό μενού της πλατφόρμας, το όνομα της εφαρμογής που αναπτύσσουμε, κουμπιά τα οποία επιτρέπουν είτε την τοπική εκτέλεση της εφαρμογής μέσω localhost ή τη διάθεσή της στο cloud (βλ. ενότητα 4.2.1.1) και σύνδεσμοι που οδηγούν στο Mendix προφίλ του χρήστη, στο Marketplace κ.α.

Στο κεντρικό τμήμα της οθόνης βρίσκεται το *Working Area*, ένας WYSIWYG σχεδιαστής όπου μπορούμε να προβάλλουμε και να επεξεργαστούμε τις σελίδες της εφαρμογής μας. Μια σελίδα μπορεί να εμφανιστεί στο Working Area με διάφορους τρόπους, οι οποίοι θα αναλυθούν στην ενότητα 4.3.3.3.

Στην αριστερή πλευρά, εντοπίζουμε το *App Explorer*, ο οποίος περιλαμβάνει τη δομή φακέλων και αρχείων της εφαρμογής, καθώς και τον *Page Explorer*, που καταγράφει όλα τα στοιχεία που έχουν χρησιμοποιηθεί στη σελίδα της εφαρμογής που είναι ανοιχτή. Στη δεξιά πλευρά, βρίσκεται το *Properties Panel*, όπου εμφανίζονται όλες οι

<sup>1</sup>Η εφαρμογή RapidDeveloper δημιουργήθηκε ως αποτέλεσμα των μαθημάτων (crash courses) που προσφέρονται μέσω του Mendix Academy. Αυτά τα μαθήματα έχουν σχεδιαστεί για να παρέχουν στους χρήστες μια γρήγορη και πρακτική εισαγωγή στις βασικές δυνατότητες της πλατφόρμας Mendix, επιτρέποντάς τους να εξοικειωθούν με τη διαδικασία ανάπτυξης εφαρμογών σε περιβάλλον low-code.

ρυθμίσεις και οι παράμετροι του στοιχείου ή της σελίδας που έχουμε επιλεγμένη, και το *Toolbox*, το οποίο περιέχει ένα σύνολο από προκατασκευασμένα στοιχεία που μπορούν να εισαχθούν στη σελίδα. Στο κάτω μέρος εμφανίζονται panels με τις αλλαγές που πραγματοποιούμε ανά commit, τα σφάλματα αν τυχόν υπάρχουν, logs, κονσόλα και άλλα. Μπορούμε σε οποιαδήποτε πεδίο να προσθέσουμε ή να αφαιρέσουμε Panels από το Μενού → View. [26]

#### 4.2.1.1 Επιλογές για deployment

Το Mendix παρέχει διάφορες επιλογές για το deployment των εφαρμογών που αναπτύσσονται στην πλατφόρμα.

Με το Mendix Free μπορούμε να κάνουμε deploy την εφαρμογή στο διαδίκτυο σε ένα URL της μορφής <όνομα εφαρμογής>-sandbox.mxapps.io, και είναι αυτό που έχει χρησιμοποιηθεί για την υλοποίηση της εφαρμογής μας στο κεφάλαιο 5. Το Mendix Free είναι κατάλληλο για την ανάπτυξη μικρών εφαρμογών και την εξοικείωση με την πλατφόρμα, αλλά δεν προσφέρει την απαιτούμενη ασφάλεια και ευελιξία για την ανάπτυξη μεγάλων επιχειρησιακών εφαρμογών καθώς οι εφαρμογές παύουν να λειτουργούν μετά από λίγες ώρες αδράνειας (sleep mode), δεν μπορούν να κλιμακωθούν, υπάρχει όριο στην υπολογιστική ισχύ και το μέγεθος της βάσης δεδομένων, δεν τρέχουν προγραμματιζόμενα γεγονότα, δεν υποστηρίζονται custom domains κ.α. Παρόλα αυτά είναι μια πολύ καλή επιλογή που εξυπηρετεί τις ανάγκες των αρχάριων χρηστών και μικρών εφαρμογών. Για χρήστες ή επιχειρήσεις με αυξανόμενες ανάγκες, το Mendix προσφέρει λύσεις επί πληρωμή που άρουν τους περιορισμούς που αναφέρθηκαν νωρίτερα. [25]

Το Mendix προφίλ κάθε χρήστη περιλαμβάνει dashboards για κάθε εφαρμογή που αναπτύσσει όπου περιλαμβάνονται ρυθμίσεις και πληροφορίες όσον αφορά το deployment.

### 4.3 Δομή εφαρμογών του Mendix

Μια εφαρμογή στο Mendix απαρτίζεται από διαφορετικά έγγραφα και modules. Τα modules επιτρέπουν τον διαχωρισμό της εφαρμογής σε αυτόνομα λειτουργικά κομμάτια. Ο τρόπος με τον οποίο πραγματοποιείται ο διαχωρισμός εξαρτάται από την κρίση και τη σχεδιαστική προσέγγιση του μηχανικού λογισμικού.

#### 4.3.1 Modules

Μια εφαρμογή αποτελείται από ένα App module, ένα System module, από modules που δημιουργούνται από τους χρήστες, από modules από το marketplace του Mendix ή από modules που καθορίζουν την εμφάνιση της εφαρμογής (UI resources modules). Τα modules του marketplace προσφέρουν έτοιμες λειτουργικότητες κατασκευασμένες

The screenshot shows the Mendix Cloud Environments page for the "UniTask" app. At the top, there's a message about being on a free version and a link to learn more. Below that, tabs for "Deploy" and "Permissions" are visible, with "Deploy" being active. A central box displays deployment status: "Your app 'UniTask' is deployed in Mendix Cloud EU: AWS Ireland (Free). Currently running version 1.0.0.811cf6c1". Below this are three buttons: "View Live Log", "Show Latest Build Output", and "Show Debugger Information". Under the heading "Activity", a table lists five deployment events with columns for "Activity" and "Date". The table includes a header row with "Activity" and "Date" and a footer showing "Showing 1 to 5 of 56 Activities". Navigation arrows and page numbers (1, 2, 3, 4, 5, ..., 12, >) are at the bottom of the activity list.

Activity	Date
Deployed model version 'main-1.0.0.811cf6c1.mda' to your sandbox by Alexandros Xiarchos.	Mon, 06 Jan 2025 22:00:12 GMT+2
Deployed model version 'main-1.0.0.a72a0c7b.mda' to your sandbox by Alexandros Xiarchos.	Sun, 29 Dec 2024 17:07:04 GMT+2
Deployed model version 'main-1.0.0.4265c558.mda' to your sandbox by Alexandros Xiarchos.	Mon, 23 Dec 2024 16:48:04 GMT+2
Deployed model version 'main-1.0.0.8abbe264.mda' to your sandbox by Alexandros Xiarchos.	Mon, 23 Dec 2024 15:44:55 GMT+2
Deployed model version 'main-1.0.0.39856037.mda' to your sandbox by Alexandros Xiarchos.	Thu, 28 Nov 2024 20:17:00 GMT+2

Σχήμα 4.3: Σελίδα Environments της εφαρμογής UniTask (κεφ. 5)

από τρίτους ενώ τα UI resources modules εμφανίζονται με πράσινο χρώμα και περιλαμβάνουν πρότυπα σελίδων (page templates) και δομικά στοιχεία (building blocks).

Για παράδειγμα, στο App Explorer της εφαρμογής RapidDeveloper (σχήμα 4.2) παρατηρούμε πως εμφανίζονται τρία διαφορετικά modules: το module App, το module System και το module MyFirstModule. Τα δύο πρώτα είναι modules που δημιουργούνται αυτόματα κατά τη δημιουργία μιας εφαρμογής, ενώ το τρίτο είναι ένα module που δημιουργήθηκε από τον χρήστη.

Κάθε module περιλαμβάνει ένα domain model που ορίζει τη δομή των δεδομένων του. Θα αναφερθούμε αναλυτικά στο domain model στην ενότητα 4.3.6. Πέρα από το domain models, κάθε module περιλαμβάνει παράθυρα για τις Ρυθμίσεις (Settings) του module και την Ασφάλεια (Security).

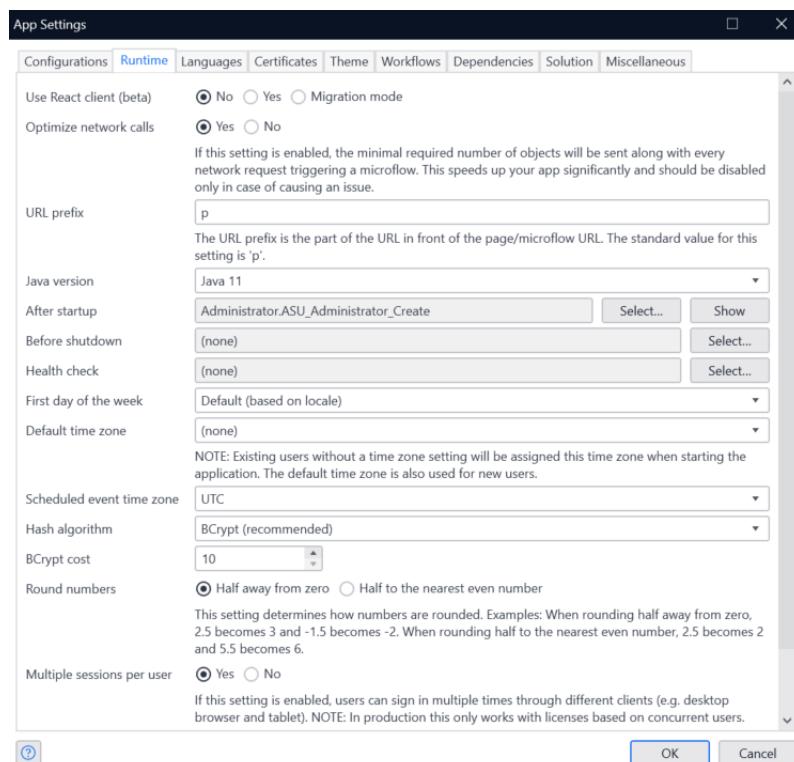
Στις Ρυθμίσεις επιτρέπεται η εξαγωγή του Module ως App Module με όλο το πηγαίο κώδικα του ή ως Add-on Module με σκοπό να χρησιμοποιηθεί από άλλους χρήστες, και επίσης εκεί μπορεί να γίνει προσθήκη Java Dependancies στο module. Στην Ασφάλεια μπορεί να καθοριστεί η πρόσβαση όλων των ρόλων χρηστών για κάθε σελίδα, οντότητα ή microflow που υπάρχει στο συγκεκριμένο module.

#### 4.3.1.1 To module App

Κάτα τη δημιουργία μιας νέας εφαρμογής, το Mendix διαθέτει ένα σύνολο από προεγκατεστημένες σελίδες με έτοιμο σχεδιασμό και λειτουργικότητα. Τα αρχεία αυτών των σελίδων βρίσκονται στο module App. Το module App πέρα από τα παρά-

θυρα των Ρυθμίσεων και Ασφάλειας (τα οποία έτσι και αλλιώς υπάρχουν σε όλα τα modules) επιπλέον περιλαμβάνει την Πλοήγηση (Navigation) και τα Κείμενα Συστήματος (System Texts)<sup>2</sup>. Επιπλέον, περιλαμβάνει ένα φάκελο Styling με .js και .css αρχεία τα οποία χρησιμοποιούνται για το styling της εφαρμογής<sup>3</sup> και έναν φάκελο Marketplace modules το οποίο περιλαμβάνει εξωτερικά modules που μπορούν να προστεθούν μέσω του Mendix Marketplace.

Το παράθυρο **Ρυθμίσεων** του App (βλ. σχήμα 4.4) διαφέρει σε σχέση με τα υπόλοιπα modules, καθώς αυτό περιλαμβάνει παραμετροποιήσεις για το runtime περιβάλλον της εφαρμογής, το theme που χρησιμοποιείται, την επιλογή συγκεκριμένων ενεργειών πριν αρχικοποιηθεί η εφαρμογή κατά την εκκίνησή της, επιλογή συγκεκριμένου αλγόριθμου κρυπτογράφησης για το Hashed String τύπο δεδομένων, καθορισμός γλώσσας και άλλα.

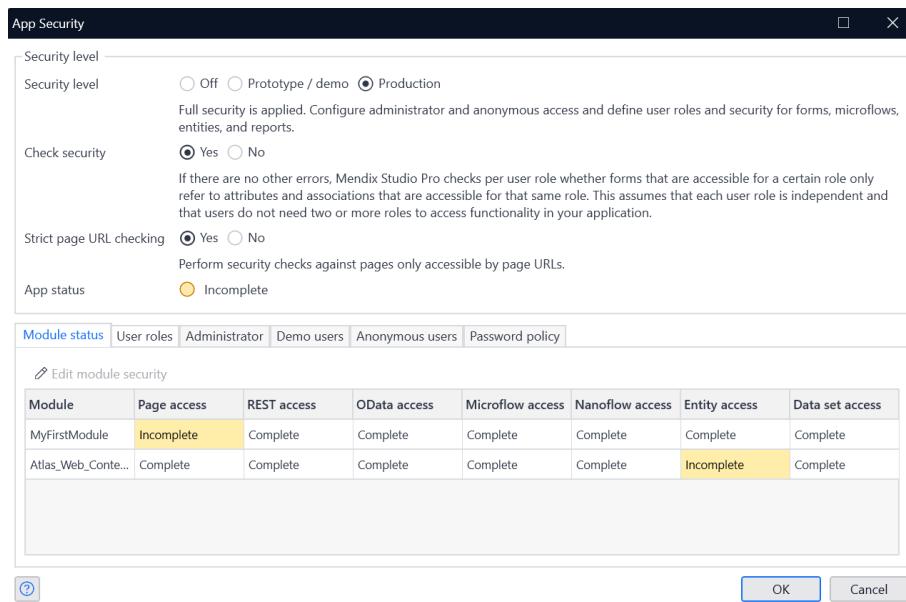


Σχήμα 4.4: Παράθυρο Settings του App

Το παράθυρο **Ασφάλεια** του App (βλ. σχήμα 4.5) το οποίο και αυτό διαφέρει

<sup>2</sup>Στο έγγραφο με τα Κείμενα Συστήματος μπορεί να γίνει μετάφραση των μηνυμάτων που παράγονται από τον διαχομιστή κατά την εκτέλεση μιας εφαρμογής (για παράδειγμα “Password too short”).

<sup>3</sup>Στα συγκεκριμένα αρχεία μπορούν να επεξεργαστούν μεταβλητές που αφορούν τη σχεδίαση του UI resources module **Atlas**, το οποίο χρησιμοποιείται ως κεντρικό theme για τις προεγκατεστημένες σελίδες του Mendix. Το **Atlas** για παράδειγμα περιλαμβάνει έτοιμα color schemes (`primary`, `success`, `warning`, `danger`, `info`) τα οποία χρησιμοποιούνται σε όλα τα widgets των σελιδών. Όπως επίσης γραμματοσειρές, spacings κτλ. Τα αρχεία λοιπόν έναν από τους τρόπους προσαρμογής της προεπιλεγμένης σχεδίασης του **Atlas**. Εναλλακτικός τρόπος είναι η δημιουργία ενός custom UI resources module.



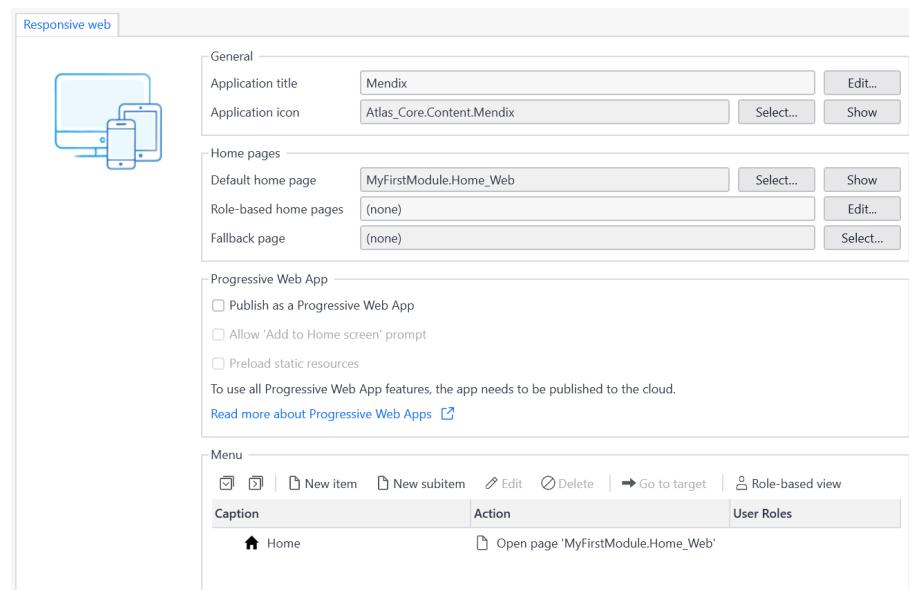
Σχήμα 4.5: Παράθυρο Security του App

σε σχέση με τα αντίστοιχα παράθυρα των υπόλοιπων modules, χρησιμοποιείται για να τροποποιηθεί το επίπεδο ασφάλειας (Security level) της εφαρμογής. Συγκεκριμένα μπορεί να επιλεγεί ώστε οι χρήστες να μη χρειάζεται να συνδεθούν για να έχουν πρόσβαση στην εφαρμογή (Security level = Off), να χρειάζεται να συνδεθούν για να ωστε να μπορέσουν να χρησιμοποιήσουν φόρμες, microflows κτλ (Security level = Prototype/demo), ή να χρειάζεται να συνδεθούν ώστε να έχουν πρόσβαση στην εφαρμογή καθολικά (Security level = Production).

Επίσης, μπορούν να οριστούν ρόλοι χρηστών όπως για παράδειγμα Administrator, User ή Guest. Κατ' αυτόν τον τρόπο καθίσταται δυνατό να διαχωριστούν τα δικαιώματα πρόσβασης των modules ανάλογα με το ποιος είναι ο ρόλος του χρήστη. Για παράδειγμα, ένας Administrator χρήστης μπορεί να έχει πλήρη πρόσβαση στα modules και στις σελίδες της εφαρμογής, ενώ ένας Guest μπορεί να έχει περιορισμένη προβολή.

Επιπλέον, στην Ασφάλεια μπορούν να οριστούν τα στοιχεία σύνδεσης του διαχειριστή (Administrator) της εφαρμογής (ώστε να μπορέσει να γίνει η πρώτη σύνδεση κατά το deployment), να οριστούν demo χρήστες (ώστε να δοκιμαστούν οι ρόλοι χρηστών) ή ανώνυμοι χρήστες (οι οποίοι έχουν πρόσβαση στην εφαρμογή χωρίς να συνδεθούν) και να ρυθμιστούν κανόνες για τους κωδικούς πρόσβασης.

Η Πλοήγηση του module App (βλ. σχήμα 4.6) εμφανίζει το σύνολο των σελίδων του κεντρικού μενού της εφαρμογής. Εκεί μπορεί να γίνει η επεξεργασία του μενού με την προσθήκη στοιχείων ή και υποστοιχείων σε στοιχεία. Επίσης, περιλαμβάνονται διαφορετικές προβολές ανάλογα με τον εκάστοτε ρόλο χρήστη, στις οποίες εμφανίζονται μόνο οι σελίδες που είναι προσβάσιμες σε κάθε ρόλο. Στην Πλοήγηση επίσης μπορεί να διαμορφωθεί το όνομα και το εικονίδιο της εφαρμογής όπως επίσης και να



Σχήμα 4.6: Το έγγραφο Navigation.

οριστεί και η αρχική σελίδα (home page) της, η οποία μάλιστα μπορεί να διαμορφωθεί ώστε να είναι διαφορετική για εκάστοτε ρόλο χρήστη. [26]

#### 4.3.1.2 To module System

Το module System είναι ένα προεγκατεστημένο module που περιλαμβάνει τη βασική λειτουργικότητα η οποία χρησιμοποιείται στις ήδη κατασκευασμένες σελίδες και microflows του App. Το συγκεκριμένο module δεν μπορεί να επεξεργαστεί από τον χρήστη, αλλά μπορούν με βάσει αυτό να προστεθούν συσχετίσεις (associations) ή γενικεύσεις (generalizations) τα modules τα οποία κατασκευάζονται από τους χρήστες. Για παράδειγμα, μπορεί να δημιουργηθεί μια οντότητα Πελάτης η οποία θα συσχετίζεται με την οντότητα User του module System και έτσι θα κληρονομεί τις ρυθμίσεις ασφαλείας του. [27]

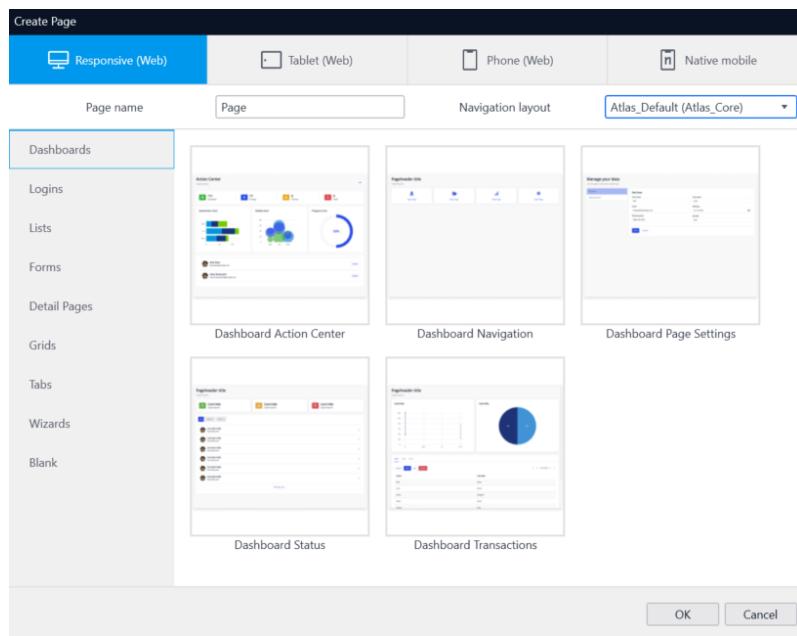
#### 4.3.2 Έγγραφα

Η Πλοιόγγηση είναι ένα έγγραφο της εφαρμογής. Άλλα παραδείγματα εγγράφων είναι οι Σελίδες (Pages) (βλ. ενότητα 4.3.3), τα Microflows (βλ. ενότητα 4.3.4) και τα Enumerations<sup>4</sup>.

<sup>4</sup>Τα enumerations (κατάλογος, απαρίθμηση) καθορίζουν μια λίστα από προκαθορισμένες τιμές. Για παράδειγμα, ένα enumeration μπορεί να χρησιμοποιηθεί για τον καθορισμό της κατάστασης μιας εργασίας ως To do, Done ή Doing.

### 4.3.3 Σελίδες

Η σελίδα είναι ο κεντρικός τρόπος αλληλεπίδρασης του χρήστη με την εφαρμογή. Η δημιουργία μιας σελίδας μπορεί να βασιστεί σε πόρους που προέρχονται από έγγραφα, όπως οι Εικόνες (Images), τα Layouts τα οποία καθορίζουν τη διάταξη της σελίδας, τα Μενού (Menus) που διαμορφώνουν την πλοήγηση, ή τα Snippets, τα οποία αποτελούν επαναχρησιμοποιήσιμα τμήματα διεπαφής.<sup>5</sup>



Σχήμα 4.7: Το παράθυρο δημιουργίας μιας νέας σελίδας.

#### 4.3.3.1 Layout

Κάθε σελίδα στο Mendix βασίζεται σε ένα προκαθορισμένο layout, το οποίο καθορίζει βασικές ιδιότητες της σελίδας, όπως το μήκος, το πλάτος ή για παράδειγμα αν πρόκειται για αναδυόμενη (dropdown) σελίδα. Επιπλέον, τα layouts επιτρέπουν τον ορισμό στατικών τμημάτων, όπως ένα header ή ένα μενού, που παραμένουν σταθερά σε όλες τις σελίδες που τα χρησιμοποιούν. Για παράδειγμα, θα μπορούσαν να δημιουργηθούν δύο layouts όπου το ένα θα έχει το μενού πλοήγησης ως μπάρα στο πάνω μέρος της οθόνης και το άλλο κάθετη στα αριστερά. Το Mendix παρέχει επίσης έτοιμα πρότυπα σελίδων (Page Templates), τα οποία διευκολύνουν τη γρήγορη και απλή δημιουργία σελίδων με προκαθορισμένη δομή και σχεδίαση.

<sup>5</sup>Το Mendix είναι μια Εφαρμογή Μίας Σελίδας (Single-page application – SPA) που σημαίνει ότι όλη η αλληλεπίδραση πραγματοποιείται σε μία μόνο καρτέλα ή παράθυρο του προγράμματος περιήγησης που φορτώνεται μία φορά και στη συνέχεια ενημερώνεται δυναμικά χωρίς να χρειάζεται να φορτωθεί ξανά. Ως αποτέλεσμα, δεν είναι δυνατή η ανοίγματος νέων σελίδων σε διαφορετική καρτέλα ή παράθυρο.

Το σχήμα 4.7 απεικονίζει το παράθυρο δημιουργίας νέας σελίδας, όπου ο χρήστης μπορεί να επιλέξει είτε από τα έτοιμα πρότυπα είτε να δημιουργήσει μια κενή σελίδα. Εδώ δύνεται επίσης η δυνατότητα επιλογής του layout (Navigation layout) της σελίδας, το οποίο μπορεί να είναι είτε ένα από τα προεγκατεστημένα layouts του Mendix είτε ένα προσαρμοσμένο layout που έχει δημιουργηθεί από τον χρήστη.

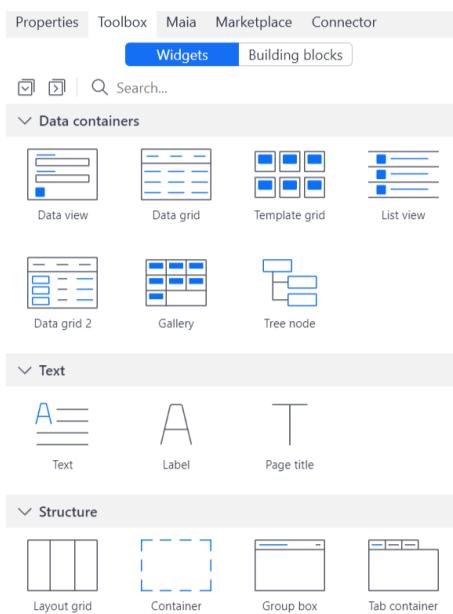
#### 4.3.3.2 Widgets

Τα Widgets είναι προκατασκευασμένα στοιχεία, έτοιμες λειτουργικές μονάδες που μπορούν να προστεθούν απευθείας σε κάθε σελίδα της εφαρμογής. Πρόκειται για εργαλεία που ενσωματώνονται εύκολα μέσω του Toolbox, όπως παρουσιάστηκε στο σχήμα 4.2. Ενδεικτικά παραδείγματα περιλαμβάνουν:

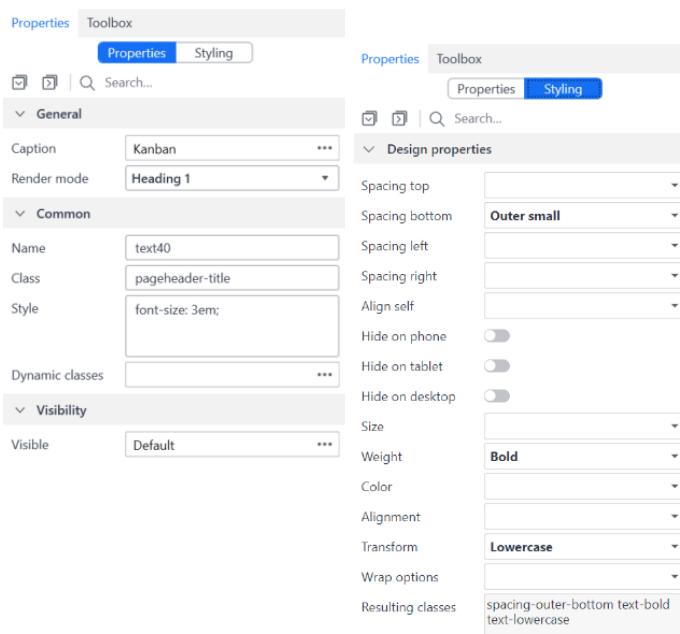
- **Data containers** – δομές δεδομένων που περιέχουν δεδομένα από τη βάση δεδομένων. Παραδείγματα είναι Data view, Data grid, List view κ.α.
- **Text widgets** – περιέχουν κείμενο. Παραδείγματα είναι Text, Label, Page Title κ.α.
- **Structure widgets** – δομικά στοιχεία που χρησιμοποιούνται για την οργάνωση των widgets στη σελίδα. Παραδείγματα είναι Layout grid, Container, Tab container, Snippet call, Table κ.α.
- **Input widgets** – πεδία εισόδου δεδομένων. Παραδείγματα είναι Text box, Text area, Check box, Radio button, Drop-down, Date picker, File uploader κ.α.
- **Images, Videos ή Files** – widgets που περιέχουν πολυμέσα.
- **Buttons** – widgets που εκτελούν ενέργειες. Το Mendix παρέχει αρκετά buttons με προκαθορισμένες ενέργειες για την εκτέλεση ενεργειών όπως Save, Cancel, Delete, New, Edit, Crate, Call microflow κ.α.

Το Mendix προσφέρει προκαθορισμένα σύνολα από widgets, γνωστά ως *Building blocks*, τα οποία δημιουργούν στοιχεία όπως επικεφαλίδες (headers), φόρμες και ειδοποιήσεις (notifications). Αυτά τα Building blocks διευκολύνουν και επιταχύνουν τη διαδικασία ανάπτυξης, παρέχοντας στους χρήστες έτοιμες λύσεις που μπορούν να ενσωματωθούν απευθείας στις εφαρμογές.

Τέλος, για κάθε widget (όπως και για κάθε σελίδα) μπορούν να επεξεργαστούν οι ιδιότητές του. Το panel Properties (σχήμα) είναι ένα δυναμικό panel στο οποίο αλλάζει το περιεχόμενό του ανάλογα με το στοιχείο που έχουμε επιλεγμένο. Στο Properties μπορούμε να ορίσουμε συνθήκες όπου θα επιτρέπεται η εμφάνιση ενός στοιχείου, να επιλέξουμε διαφορετικά render styles από τα προκατασκευασμένα που παρέχει το Mendix, να ορίσουμε events που θα συμβαίνουν όταν ο χρήστης αλληλεπιδρά με το στοιχείο, όπως επίσης και να αλλάξουμε CSS κλάσεις είτε μέσω των παρεχόμενων επιλογών του Mendix είτε μέσω της προσθήκης δικού μας custom CSS κώδικα.



Σχήμα 4.8: Τα Widgets περιλαμβάνονται στο Toolbox panel.

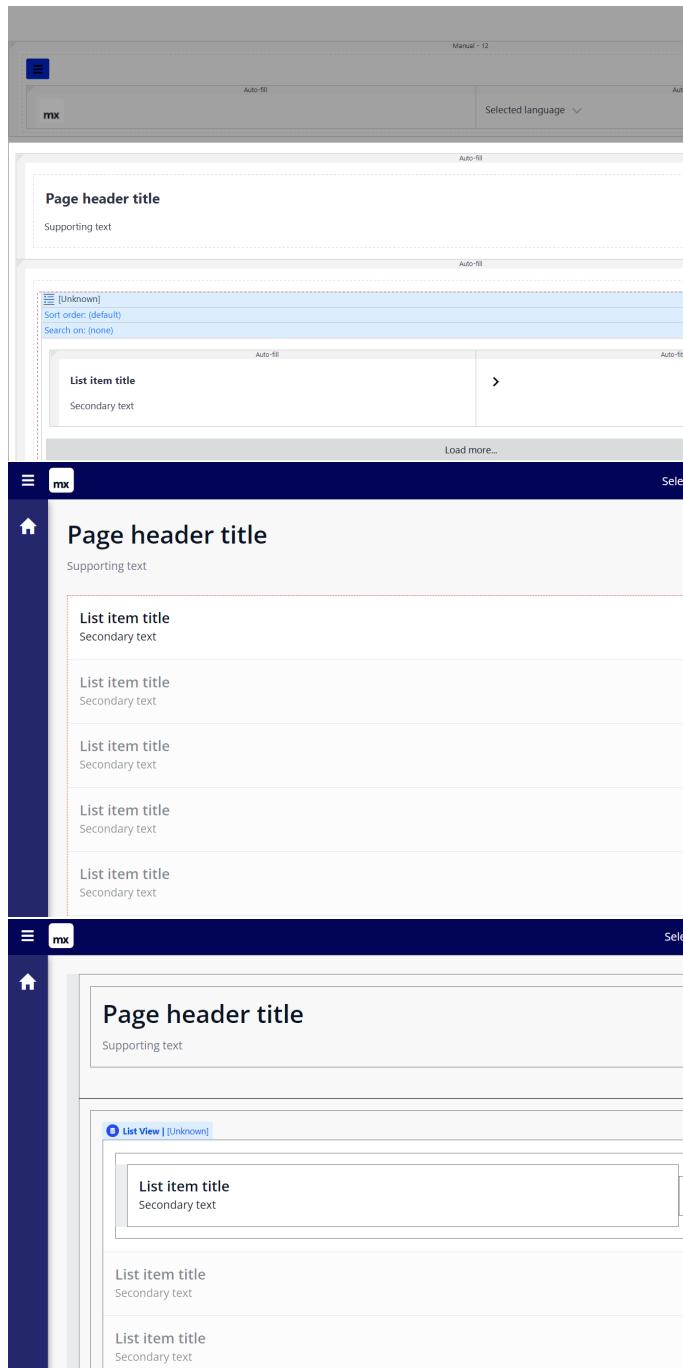


Σχήμα 4.9: Το panel Properties.

#### 4.3.3.3 Εμφάνιση σελίδων

Στο Mendix Studio Pro, μια σελίδα μπορεί να προβληθεί είτε σε Structure Mode είτε σε Design Mode, προσφέροντας στους χρήστες διαφορετικές οπτικές για τη διαχείριση και τον σχεδιασμό των σελίδων.

Στο **Structure Mode**, παρουσιάζονται με σαφήνεια όλα τα δομικά στοιχεία που



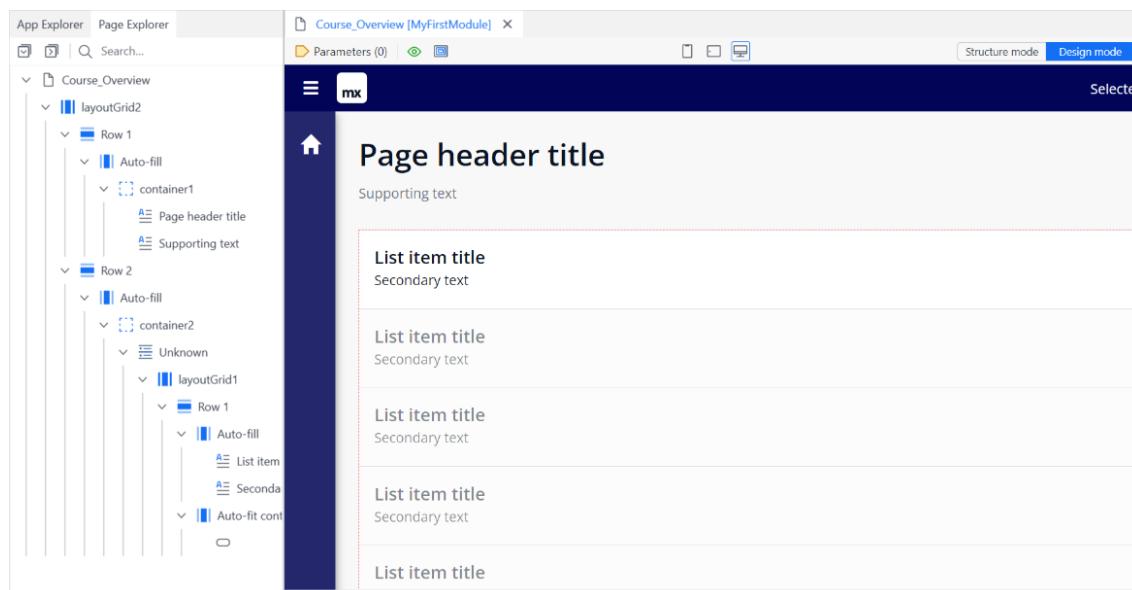
Σχήμα 4.10: Διαφορετικές εμφανίσεις της ίδιας σελίδας από αριστερά προς τα δεξιά: Structure Mode, Design Mode, X-Ray Mode.

συνθέτουν τη σελίδα, δίνοντας έμφαση στη δομή της και επιτρέποντας την εύκολη προσαρμογή και οργάνωση των περιεχομένων. Αυτή η λειτουργία είναι ιδανική για την ανάλυση της λογικής της σελίδας, τη διαχείριση των στοιχείων της και τη διόρθωση πιθανών προβλημάτων διάταξης. Είναι επίσης ο μοναδικός τρόπος προβολής των εφαρμογών κινητών συσκευών (Native mobile).

Αντίθετα, στο **Design Mode**, η σελίδα απεικονίζεται όπως ακριβώς θα εμφανίζεται στον τελικό χρήστη. Αυτό προσφέρει μια πιο ρεαλιστική απεικόνιση της εμπειρίας χρήστη. Επιπλέον, το Design Mode περιλαμβάνει τη λειτουργία **X-Ray Mode**, η οποία συνδυάζει στοιχεία από το Structure Mode και το Design Mode. Με το X-Ray Mode, οι χρήστες μπορούν να δουν τόσο την αισθητική εμφάνιση όσο και τη δομή της σελίδας ταυτόχρονα. Αυτή η λειτουργία επιτρέπει την ακριβή τοποθέτηση και διαχείριση στοιχείων, ενώ ταυτόχρονα προσφέρει τη δυνατότητα άμεσων προσαρμογών σε επίπεδο σχεδιασμού και λογικής.

Οι διαφορετικές προβολές μπορούν να επιλεγόνται από το πάνω μέρος του Working Area, το οποίο επιπλέον περιλαμβάνει και δυνατότητα αλλαγών στο μέγεθος του και μάζα της σελίδας ώστε να ελεγχθεί το πώς εμφανίζεται η εφαρμογή σε κινητά και τάμπλετ.

#### 4.3.3.4 Παράδειγμα δομής σελίδας



Σχήμα 4.11: Page Explorer και Working area της σελίδας Course\_Overview.

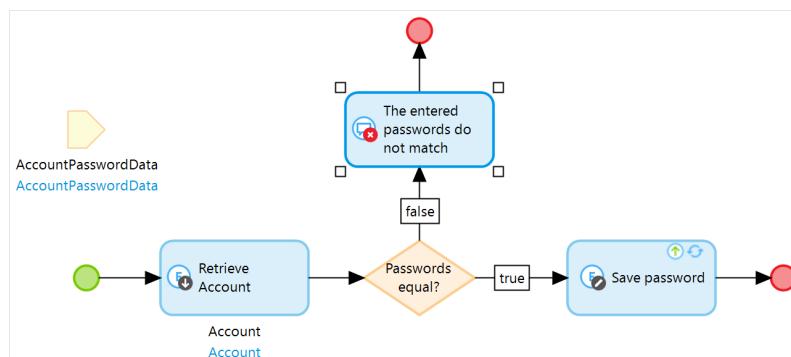
Στο σχήμα 4.11 εμφανίζεται ο Page Explorer και το Working area της σελίδας Course\_Overview της εφαρμογής RapidDeveloper. Παρατηρούμε πως όλη η σελίδα είναι δομημένη γύρω από ένα Layout Grid, το οποίο αποτελείται από δύο Rows (γραμμές). Το Row 1 δημιουργεί το Header της σελίδας μαζί με το Supporting text τα οποία είναι τοποθετημένα σε ένα container<sup>6</sup>, και το Row 2 δημιουργεί ένα List View το οποίο επαναλαμβάνεται. Το List View αποτελείται και αυτό από ένα Layout Grid

<sup>6</sup>Τα containers χρησιμοποιούνται καθώς είναι ένας βολικός τρόπος ομαδοποίησης κοινών στοιχείων της σελίδας. Έτσι μπορούν να δωθούν εύκολα κανόνες για αυτά τα στοιχεία όπως για παράδειγμα το visibility, συγκεκριμένα events ή και custom CSS κλάσεις.

το οποίο δημιουργεί τα δύο texts τα οποία βλέπουμε στη λίστα όπως επίσης και ένα κουμπί (το οποίο εμφανίζεται εκτός οθόνης). Οι μπλε οριζόντια μπάρα και η μπλε κάθετη μπάρα στα αριστερά αποτελούν κομμάτι του Layout της σελίδας, το οποίο είναι το `Atlas_Default`.

#### 4.3.4 Microflows

Τα Microflows (όπως και τα Nanoflows και τα Workflows)<sup>7</sup> αποτελούν τον εγκέφαλο των εφαρμογών του Mendix καθώς αναπαριστούν της λογικής λειτουργίας της εφαρμογής με έναν οπτικό τρόπο χαμηλού κώδικα. Αυτά τα διαγράμματα ροής λειτουργούν ως οδικοί χάρτες για την εκτέλεση διάφορων εντολών και αλληλουχιών ενεργειών, επιτρέποντας την κατασκευή σύνθετης λειτουργικότητας χωρίς την ανάγκη γραφής παραδοσιακού κώδικα. Χρησιμοποιούνται ευρέως για ενέργειες όπως η δημιουργία, η ενημέρωση και η διαγραφή δεδομένων, η εμφάνιση σελίδων, το φίλτρορισμα δεδομένων, η εκτέλεση ελέγχων, η είσοδος δεδομένων από εξωτερικές πηγές και άλλα



Σχήμα 4.12: Παράδειγμα Microflow. Τα πράσινα και κόκκινα κυκλώματα αναπαριστούν τα events, τα μπλε ορθογώνια τα activities και ο πορτοκαλί ρόμβος το decision. Ως είσοδο έχουμε το parameter `AccountPasswordData`. [26].

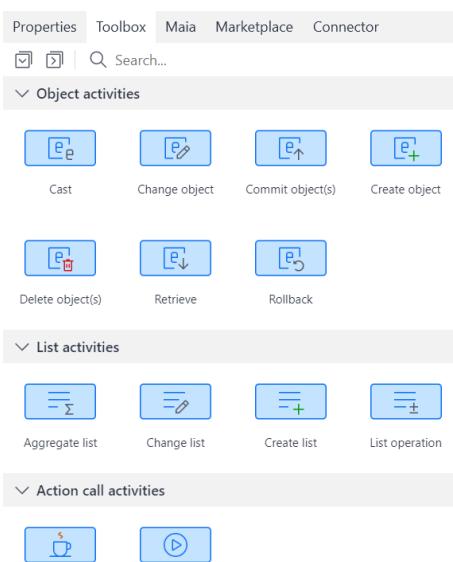
Τα Microflows αποτελείται από τα παρακάτω στοιχεία:

<sup>7</sup>Η βασική διαφορά μεταξύ Microflows και Nanoflows έγκειται στη λειτουργικότητα και τον τρόπο εκτέλεσής τους. Τα Microflows βασίζονται σε βιβλιοθήκες της Java, εκτελούνται στον διαχομιστή (runtime server) και, ως εκ τούτου, δεν είναι διαθέσιμα για εφαρμογές που λειτουργούν εκτός σύνδεσης. Από την άλλη πλευρά, τα Nanoflows χρησιμοποιούν βιβλιοθήκες της JavaScript, εκτελούνται στην πλευρά του client, γεγονός που τα καθιστά εν δυνάμει γρηγορότερα.

Τα Microflows είναι ιδανικά για την προσκόμιση και την επεξεργασία δεδομένων από τη βάση δεδομένων ή από εξωτερικές πηγές, εξασφαλίζοντας υψηλή αξιοπιστία και συνέπεια. Αντίθετα, τα Nanoflows χρησιμοποιούνται κυρίως για ενέργειες που σχετίζονται με την εμπειρία του χρήστη, όπως η εμφάνιση αναδυόμενων (pop-up) μηνυμάτων, η προβολή progress bars ή η αντολλαγή cookies.

Τέλος, τα Workflows ενδείκνυνται για τη διαχείριση σταθερών και επαναλαμβανόμενων διαδικασιών, επιτρέποντας την αυτοματοποίηση και την απλοποίηση της εκτέλεσής τους.

- **Events** – λειτουργούν ως σημεία εκκίνησης και τερματισμού του Microflow. Χρησιμοποιώντας το τελικό event μπορούμε να ορίσουμε την τιμή και το τύπο δεδομένων που επιστρέφει το Microflow, με παρόμοιο τρόπο όπως στις συναρτήσεις ή μεθόδους του υψηλού κώδικα.
- **Decisions** – επιτρέπουν την εισαγωγή λογικών συνθηκών. Για παράδειγμα, ένα decision μπορεί να ελέγχει αν μια μεταβλητή έχει τιμή και, ανάλογα με την απάντηση, να κατευθύνει τη ροή σε διαφορετικές ενέργειες. Οι συνθήκες ορίζονται με τη χρήση εκφράσεων (βλ. ενότητα 4.3.5).
- **Activities** – αποτελούν τις κύριες ενέργειες που εκτελούνται στη ροή. Παραδείγματα τέτοιων ενεργειών είναι η δημιουργία (ή ενημέρωση ή διαγραφή) αντικειμένων μέσω του activity Create (ή Change ή Delete) object, η εμφάνιση μιας σελίδας στον χρήστη μέσω του Show page, η κλήση ενός άλλου Microflow μέσω του Microflow call, το φιλτράρισμα λιστών (List operation), η σύνδεση με εξωτερικές υπηρεσίες μέσω REST APIs κ.α.
- **Loops** – επιτρέπουν την εκτέλεση επαναλαμβανόμενων ενεργειών.
- **Parameter** – πρόκειται για τα δεδομένα εισόδου του Microflow, με αντίστοιχη λογική όπως οι συναρτήσεις υψηλού κώδικα.



Σχήμα 4.13: Όταν επεξεργαζόμαστε ένα Microflow, το Toolbox panel περιλαμβάνει τα διαθέσιμα activities.

Ένα Microflow μπορεί να εκτελεστεί από διαφορετικά μέρη όπως το Navigation menu, ένα κουμπί, ένα link ή ακόμα και να καλεστεί από ένα άλλο Microflow. Επιπλέον, τα Microflows μπορούν να εκτελεστούν αυτόματα μετά από μια συγκεκριμένη ενέργεια (Event Handlers), όπως η αποθήκευση ενός αντικειμένου ή η ενημέρωση ενός πεδίου.

Τέλος, κάθε Microflow αποθηκεύεται ως ένα Java αρχείο στον πηγαίο κώδικα της εφαρμογής. Για εξειδικευμένες λειτουργίες, το Mendix παρέχει τη δυνατότητα ενσωμάτωσης προσαρμοσμένου κώδικα Java μέσω του App → Deploy to Eclipse.

#### 4.3.5 Εκφράσεις

Οι εκφράσεις (expressions) του Mendix είναι ένας τρόπος ενσωμάτωσης λειτουργικότητας στην εφαρμογή μας. Οι εκφράσεις μπορούν να περιλαμβάνουν σταθερές τιμές, μεταβλητές, συναρτήσεις, λογικές πράξεις, συγκρίσεις, επιλογές κ.α. Για παράδειγμα, μπορεί να οριστεί η εμφάνιση ενός συγκεκριμένου widget μόνο αν ισχύει μια συγκεκριμένη συνθήκη. Οι εκφράσεις μπορούν να χρησιμοποιηθούν σε πολλά σημεία της εφαρμογής, όπως στα Microflows, στις ιδιότητες των widgets κ.α.

Για παραδειγμα, η έκφραση `if $package/weight < 1.00 then 0.00 else 5.00` ελέγχει το γνώρισμα `weight` της οντότητας `package` και επιστρέφει 0.00 αν το βάρος είναι μικρότερο από 1.00, αλλιώς επιστρέφει 5.00. Θα δούμε περισσότερες τέτοιες εκφράσεις στην πράξη στο κεφάλαιο 5.

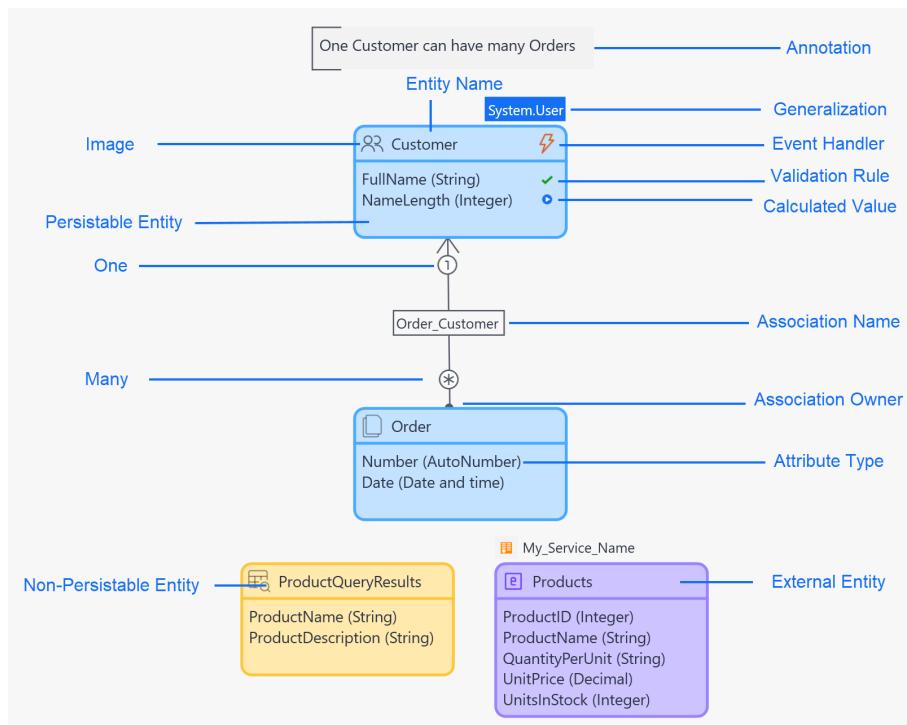
#### 4.3.6 Domain Model

Το domain model αναπαριστά τη δομή των δεδομένων κάποιου module στην πλατφόρμα Mendix. Τα δεδομένα που περιγράφονται από το domain model αποθηκεύονται στη συνέχεια σε ένα σχεσιακό σύστημα βάσεων δεδομένων του Mendix.

Το domain model αποτελεί κεντρικό πυλώνα της αρχιτεκτονικής κάθε εφαρμογής. Κάθε module έχει το δικό του domain model, και όλα τα modules μπορούν να χρησιμοποιούν δεδομένα από όλα υπόλοιπα domain modules μέσω συσχετίσεων.

Το σχήμα 4.14 είναι ένα παράδειγμα ενός domain model που αναπαριστά πελάτες και παραγγελίες. Οι πελάτες και οι παραγγελίες αποτελούν οντότητες (entities) του domain model. Οι οντότητες συσχετίζονται μεταξύ τους με μια συσχέτιση (association) πολλών-προς-ένα. Κάθε παραγγελία ανήκει σε έναν μόνο πελάτη, ενώ ένας πελάτης μπορεί να έχει πολλές παραγγελίες. Φυσικά, το Mendix περιλαμβάνει και άλλες πληθικότητες, όπως συσχετίσεις ένα-προς-ένα όπως επίσης και πολλά-προς-πολλά. Επιπλέον, αν διαγραφτεί κάποια οντότητα μπορεί να ρυθμιστεί τι θα συμβεί με τις συσχετίσεις της (π.χ. να διαγραφούν και αυτές).

Μέσα στα ορθογώνια που αναπαριστούν τις οντότητες βρίσκονται τα γνωρίσματα, οι ιδιότητες (attributes) των οντοτήτων. Στην παρένθεση κάθε γνωρίσματος καταγράφεται ο τύπος δεδομένων του. Παρατηρούμε πως υπάρχουν ορθογώνια με διαφορετικά χρώματα, κάτι που αντιστοιχεί σε διαφορετικού είδους οντότητες. Τα μπλε ορθογώνια αναπαριστούν οντότητες που αποθηκεύονται στη βάση δεδομένων, με κίτρινο μη-διατηρήσιμες οντότητες (non-persistent entities), δηλαδή οντότητες που δεν αποθηκεύονται στη βάση δεδομένων αλλά αποθηκεύονται προσωρινά στη μνήμη, και τέλος με μωβ οντότητες από εξωτερικές πηγές δεδομένων. Η μπλε ετικέτα `System.User` που συνοδεύει την οντότητα `Customer` δηλώνει πως η οντότητα αυτή βασίζεται σε μια



Σχήμα 4.14: Παράδειγμα από domain model. [26]

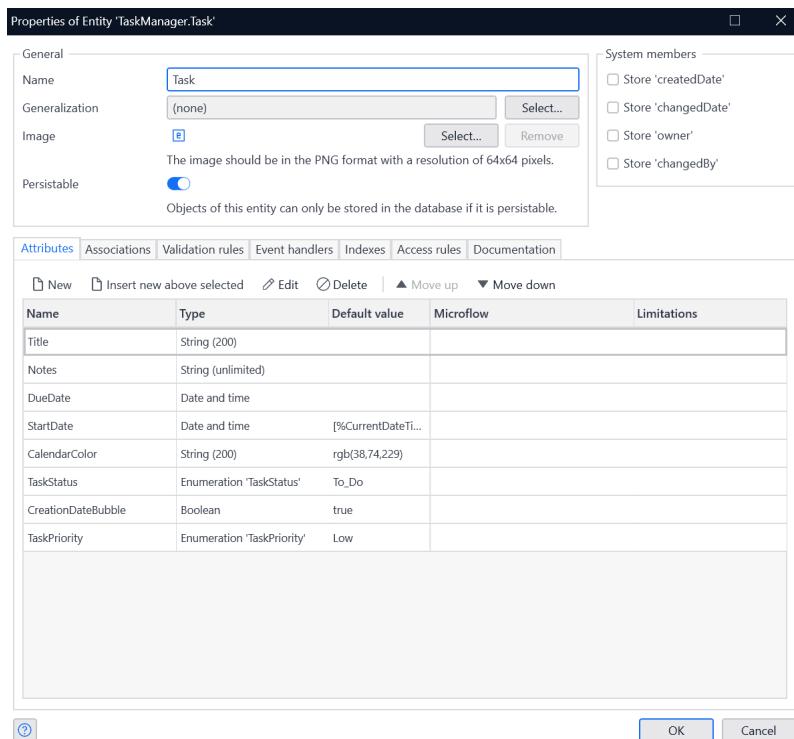
άλλη οντότητα, την οντότητα User του module system (Γενίκευση – Generalization)<sup>8</sup>. Τέλος, ανάλογα αν στην οντότητα έχει οριστεί κάποιος Event handler ή αν σε κάποιο γνώρισμα υπάρχει κάποιο validation rule, το Mendix το αναγνωρίζει και το αναπαριστά με το αντίστοιχο σύμβολο.

#### 4.3.6.1 Οντότητες

Στο σχήμα 4.15 παρουσιάζεται το παράθυρο που εμφανίζεται όταν δημιουργούμε ή επεξεργαζόμαστε μια οντότητα. Στο πάνω μέρος ορίζεται αν η οντότητα έχει κάποια γενίκευση και το αν είναι διατηρήσιμη στη βάση δεδομένων.

Στην καρτέλα **Attributes** (Γνωρίσματα) καθορίζονται όλα τα γνωρίσματα της οντότητας. Εκεί επιλέγεται ο τύπος δεδομένων του γνωρίσματος. Οι τύποι δεδομένων που υποστηρίζονται από το Mendix είναι οι εξής: AutoNumber (αυτόματα παραγόμενοι αριθμοί, π.χ. IDs), Binary, Boolean, Date and time, Decimal, Enumeration

<sup>8</sup>Η έννοια της γενίκευσης στο Mendix βασίζεται σε μια λογική που θυμίζει την κληρονομικότητα (inheritance) στις αντικειμενοστραφείς γλώσσες προγραμματισμού, δηλαδή επιτρέπει σε μία οντότητα (entity) να κληρονομεί τις ιδιότητες και τις συσχετίσεις μιας άλλης υπεροντότητας, χρησιμοποιώντας τα χαρακτηριστικά και τις συσχετίσεις της, ενώ παράλληλα μπορεί να ορίσει πρόσθετες ιδιότητες ή συσχετίσεις που είναι μοναδικές για την ίδια. Η γενίκευση είναι ιδιαίτερα χρήσιμη καθώς συμβάλλει στη διατήρηση της ακεραιότητας και της ασφάλειας των δεδομένων. Για παράδειγμα, οι “Customers” που κληρονομούν τα γνωρίσματα της οντότητας System.User, ταυτόχρονα κληρονομούν και όλα τα χαρακτηριστικά ασφάλειας του User που το Mendix έχει προκατασκευάσει.



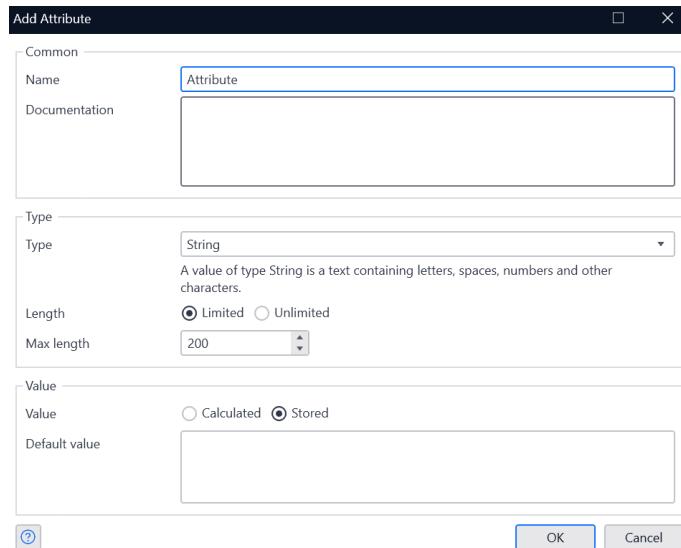
Σχήμα 4.15: Ιδιότητες μιας οντότητας Task (βλ. κεφ. 5)

(επιλέγεται κάποιο Enumeration έγγραφο), Hashed string, Integer, Long, String. Ο τύπος ενός γνωρίσματος είναι πιθανό να καθοριστεί αυτόματα από το Mendix βάσει του ονόματος που επιλέγεται κατά τον ορισμό του. Τέλος, μπορεί να οριστεί μια προ-επιλεγμένη τιμή του κάθε ορίσματος ή να οριστεί η τιμή να καθορίζεται από κάποιο microflow.

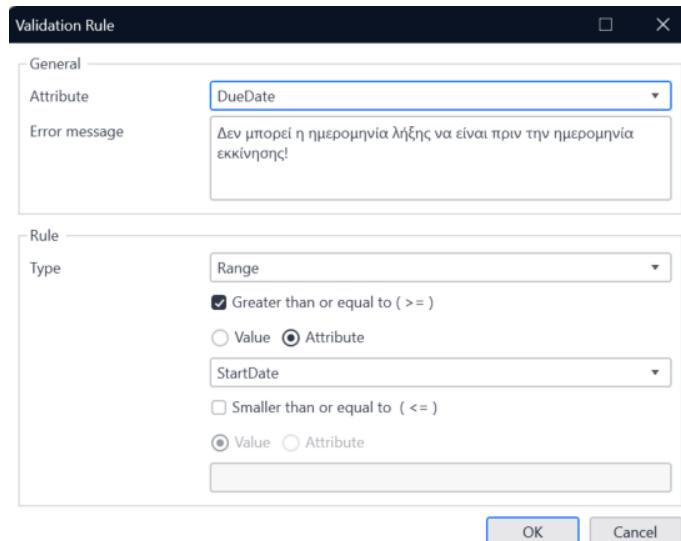
Η καρτέλα **Associations** (Συσχετίσεις) είναι ένας εναλλακτικός τρόπος που επιτρέπει να επεξεργαστούν οι συσχετίσεις μιας οντότητας. Ένας εναλλακτικός τρόπος είναι απευθείας από το domain model μέσω των συνδέσεων μεταξύ των οντοτήτων.

Η καρτέλα **Validation rules** (Κανόνες Επικύρωσης) δημιουργεί συνθήκες που πρέπει να ικανοποιούνται για να είναι έγκυρα τα γνωρίσματα κάθε οντότητας. Οι κανόνες επικύρωσης μπορούν να είναι απλές συνθήκες, όπως π.χ. ένα πεδίο να μην είναι κενό, ή πιο σύνθετες, όπως π.χ. η τιμή ενός πεδίου να καθορίζεται από ένα εύρος τιμών. Αν η συνθήκη δεν πληρούται, εμφανίζεται αυτόματα μήνυμα σφάλματος κάτω από το πεδίο.

Η καρτέλα **Event handlers** (Χειριστές Συμβάντων) είναι ένας τρόπος για να εκτελεστούν συγκεκριμένες ενέργειες (microflows) όταν συμβεί κάποιο συγκεκριμένο συμβάν στην οντότητα. Τα συμβάντα μπορεί να είναι η δημιουργία (create), η ενημέρωση (commit ή save), η διαγραφή (delete) ή η ακύρωση (rollback ή cancel) μιας οντότητας. Μπορεί να επιλεγεί η εκτέλεση του microflow πριν ή μετά την εκτέλεση των παρακάτω συμβάντων.



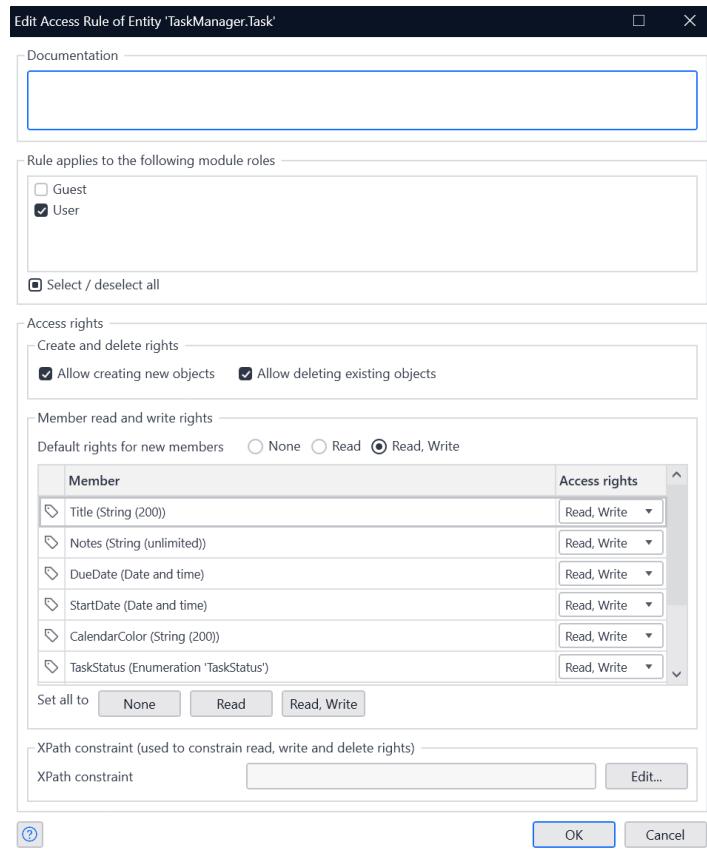
Σχήμα 4.16: Προσθήκη καινούριου γνωρίσματος σε μια οντότητα



Σχήμα 4.17: Προσθήκη κανόνα επικύρωσης (βλ. κεφ. 5)

Η καρτέλα **Access Rules** (Κανόνες Πρόσβασης) ορίζει τα δικαιώματα χρήστης από κάθε ρόλο χρήστη όσον αφορά την αλληλεπίδραση με τα γνωρίσματα της οντότητας. Μπορεί να επιλεγεί ποιος ρόλος χρήστη μπορεί να δημιουργήσει ή να διαγράψει στιγμιότυπα από οντότητες και να διαβάσει ή να τροποποιήσει τα γνωρίσματά τους.

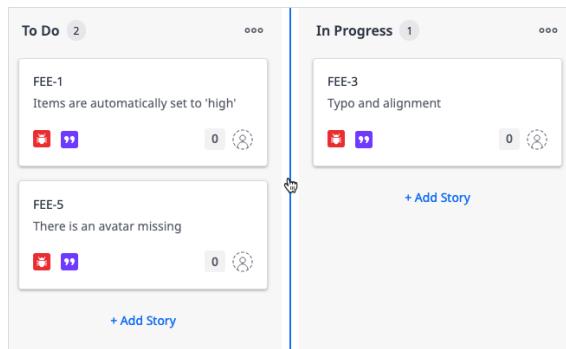
Τέλος, μπορούν να προστεθούν breakpoints στα Microflows ώστε να γίνουν debug πιθανά σφάλματα κατά την εκτέλεση τους.



Σχήμα 4.18: Επεξεργασία κανόνων πρόσβασης της οντότητας Task (βλ. κεφ. 5)

#### 4.3.7 Dashboard εφαρμογής

Παράλληλα με την ανάπτυξη της εφαρμογής στο Mendix Studio Pro, στο προσωπικό προιοφίλ κάθε χρήστη υπάρχει ένα dashboard για κάθε εφαρμογή που αναπτύσσουμε όπου παρέχονται project management εργαλεία και μεθοδολογίες ανάπτυξης λογισμικού.



Σχήμα 4.19: Πίνακας Kanban στο dashboard. [26]

## Κεφάλαιο 5

# Υλοποίηση εφαρμογής

Σε αυτό το κεφάλαιο...

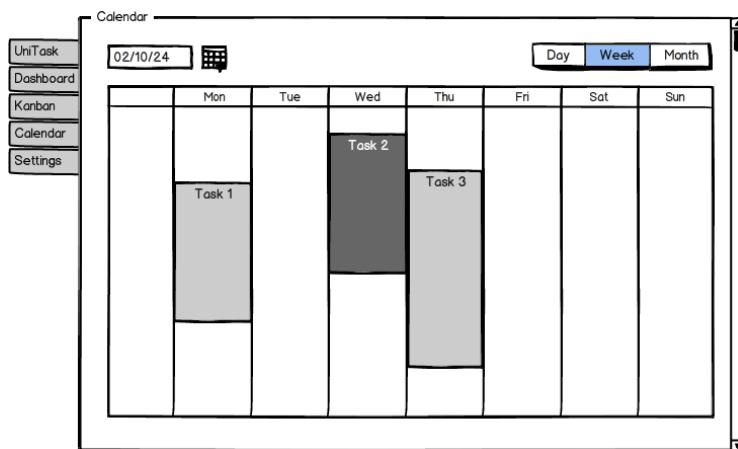
### 5.1 Mockups και σχεδιαστική προσέγγιση

Στην ενότητα 2.5.2, παρουσιάστηκε μια έρευνα με τα βασικά χαρακτηριστικά που θεωρήθηκαν απαραίτητα από τους φοιτητές για μια εφαρμογή τους. Λαμβάνοντας υπόψιν τις προτιμήσεις αυτές δόθηκε βάση στην υλοποίηση τους ώστε η εφαρμογή να ανταποκρίνεται στις ανάγκες της ακαδημαϊκής κοινότητας.

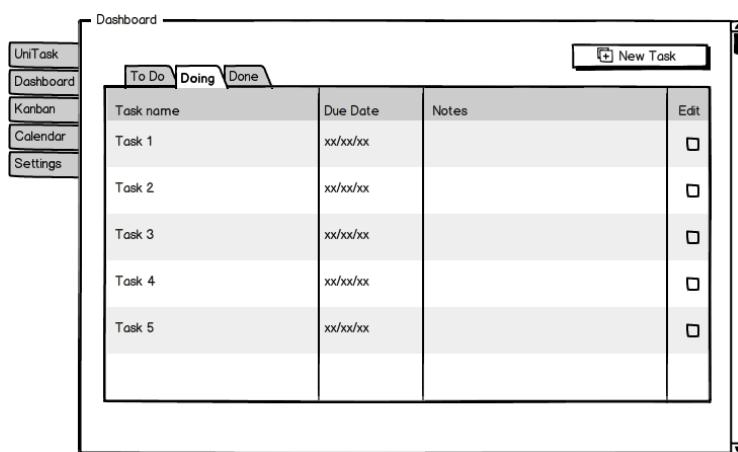
Αρχικά, ως μια εφαρμογή διαχείρισης εργασιών, θα περιλαμβάνει προφανώς ένα σύστημα δημιουργίας, τροποποίησης και διαγραφής εργασιών, καθορισμού του χρόνου έναρξης και λήξεώς τους, και η κατηγοριοποίηση των εργασιών σε κατηγορίες ανάλογα με το αν έχουν πραγματοποιηθεί, αν πραγματοποιούνται και αν έχουν σκοπό να πραγματοποιηθούν μελλοντικά. Επιπλέον, με βάση την έρευνα, είναι σημαντική η ενσωμάτωση ενός ημερολογίου, η δυνατότητα χρωματικής ταξινόμησης (color-coding) και η υλοποίηση ενός συστήματος ανταμοιβής για την ενίσχυση της παρακίνησης των χρηστών. Επίσης, θα ήταν εξίσου σημαντική η δημιουργία ενός Kanban πίνακα για την άμεση οπτικοποίηση των εργασιών και την ευκολότερη διαχείρισή τους.

Σχεδιαστικά θεωρείται σημαντική η τήρηση σύγχρονων σχεδιαστικών κανόνων με ένα καθαρό interface και συνοχή στον σχεδιασμό για τη δημιουργία μιας λειτουργικής, αισθητικά ευχάριστης και ευκολόχρηστης εμπειρίας χρήστη ώστε να εξασφαλιστεί ότι η εφαρμογή μπορεί να ανταποκριθεί στις ανάγκες διαφορετικών τύπων χρηστών, αλλά και να παρουσιαστεί ως ένα προϊόν έτοιμο για χρήση σε πραγματικά περιβάλλοντα.

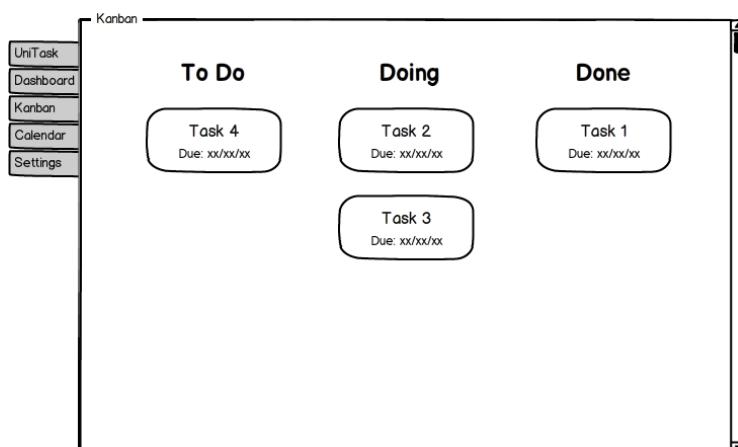
Στα σχήματα 5.1, 5.2 και 5.3 παρουσιάζονται κάποια αρχικά mockups που χρησιμοποιήθηκαν για τον σχεδιασμό της εφαρμογής.



Σχήμα 5.1: Mockup Calendar σελίδας



Σχήμα 5.2: Mockup Dashboard σελίδας

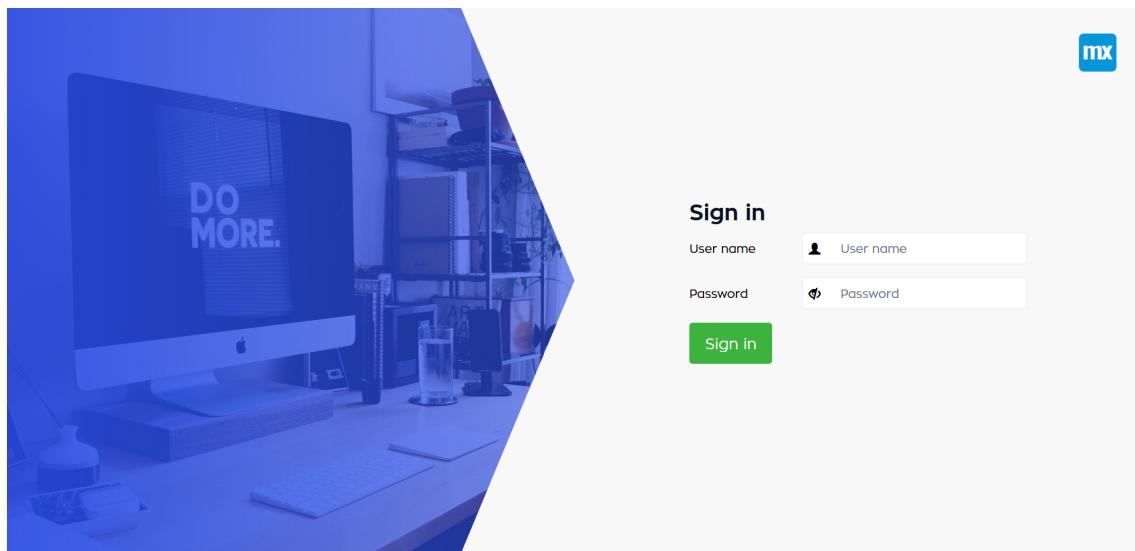


Σχήμα 5.3: Mockup Kanban σελίδας

## 5.2 Παρουσίαση της εφαρμογής

Κατά την εκτέλεση της εφαρμογής, είτε τοπικά είτε μέσω της απομακρυσμένης πρόσβασης στο cloud, στη διεύθυνση <https://unitask-sandbox.mxapps.io/>, εμφανίζεται αρχικά η **σελίδα σύνδεσης**, όπως φαίνεται στο σχήμα 5.4. Στη συγκεκριμένη σελίδα, οι χρήστες καλούνται να εισάγουν τα στοιχεία σύνδεσής τους για να αποκτήσουν πρόσβαση στις λειτουργίες της εφαρμογής.

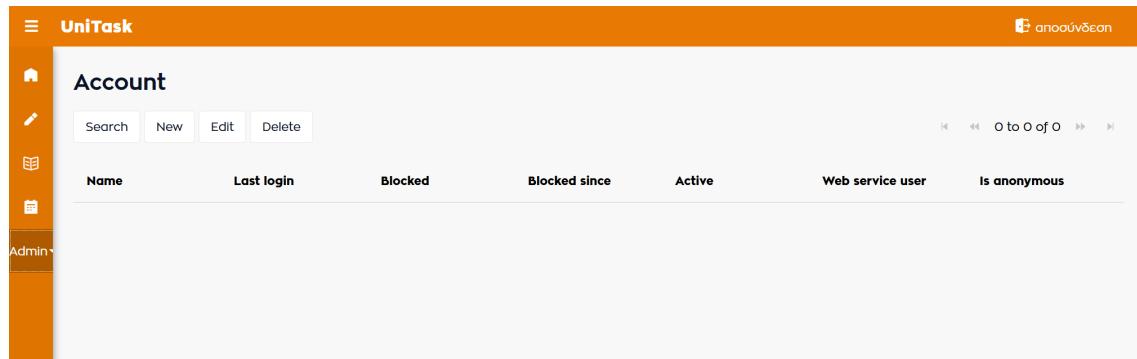
Το σύστημα αναγνωρίζει δύο επίπεδα πρόσβασης, ανάλογα με τα στοιχεία σύνδεσης που εισάγονται: δικαιώματα διαχειριστή (Administrator) και δικαιώματα χρήστη (User). Ο ρόλος του χρήστη (User) αντιστοιχεί σε φοιτητές που κάνουν χρήση της εφαρμογής, ενώ ο ρόλος του διαχειριστή παρέχει επιπλέον λειτουργίες διαχείρισης.



Σχήμα 5.4: Σελίδα σύνδεσης

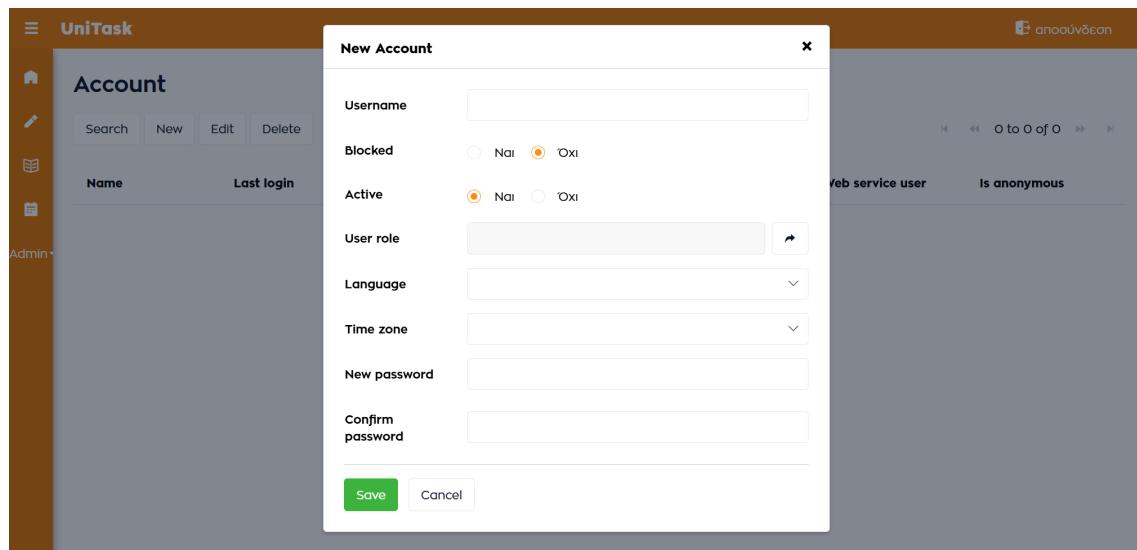
Αρχικά, προγματοποιείται σύνδεση με τον λογαριασμό διαχειριστή (Administrator) προκειμένου να παρουσιαστούν οι λειτουργίες διαχείρισης χρηστών, συμπεριλαμβανομένης της δυνατότητας προσθήκης νέου χρήστη. Μετά την επιτυχή εισαγωγή των διαπιστευτηρίων του διαχειριστή, τα οποία έχουν οριστεί προκαταβολικά κατά την ανάπτυξη της εφαρμογής (βλ. ενότητα 5.3), εμφανίζεται η **σελίδα διαχείρισης χρηστών**, όπως απεικονίζεται στο σχήμα 5.5. Η σελίδα αυτή παρέχει στους διαχειριστές μια ολοκληρωμένη επισκόπηση της λίστας χρηστών της εφαρμογής, καθώς και εργαλεία για τη διαχείρισή τους. Το layout της σελίδας αποτελείται από μια κάθετη μπάρα μενού η οποία περιλαμβάνει τις ίδιες δυνατότητες με τους απλούς χρήστες η οποίες θα αναλυθούν στη συνέχεια.

Πατώντας στο κουμπί New, εμφανίζεται η αναδυόμενη σελίδα του σχήματος 5.6 με μια φόρμα για την προσθήκη νέου χρήστη. Η φόρμα περιλαμβάνει πεδία για την εισαγωγή του ονόματος χρήστη (Username), του ρόλου του χρήστη (User role) όπου επιλέγεται αν πρόκειται για προσθήκη διαχειριστή ή χρήστη, του κωδικού πρόσβασης



Σχήμα 5.5: Σελίδα διαχείρισης χρηστών

(New password και Confirm password). Λόγω του ότι ο χρήστης User έχει κληρονομήσει γνωρίσματα από την κλάση System.User του Mendix, έχουν προστεθεί πεδία όπως το Blocked, η οποία γίνεται αληθής μετά από κάποιες αποτυχημένες προσπάθειες σύνδεσης, το Active που γίνεται αληθές όταν ο χρήστης συνδεθεί, το Time zone όπου ορίζεται η ζώνη ώρας του χρήστη και το Language όπου ορίζεται η γλώσσα του χρήστη.



Σχήμα 5.6: Φόρμα προσθήκης νέου χρήστη

Ένας νέος χρήστης δημιουργείται με το όνομα Foithths. Ο χρήστης προστίθεται στη λίστα χρηστών, όπως φαίνεται στο σχήμα 5.7. Πατώντας στο όνομά του, εμφανίζεται η σελίδα επεξεργασίας του χρήστη, όπως φαίνεται στο σχήμα 5.8. Στη λίστα των χρηστών υπάρχει η δυνατότητα αναζήτησης χρηστών βάσει όλων των στοιχείων τους (σχήμα 5.9), όπως επίσης και η δυνατότητα διαγραφής τους.

Μετά την αποσύνδεση από τον λογαριασμό διαχειριστή και τη σύνδεση ως Foithths, εμφανίζεται η αρχική σελίδα της εφαρμογής (σχήμα 5.10). Η σελίδα περιλαμβάνει

Name	Last login	Blocked	Blocked since	Active	Web service user	Is anonymous
Foithths	Oxi	Nai		Oxi	Oxi	Oxi

Σχήμα 5.7: Λίστα χρηστών με τον χρήστη Foithths

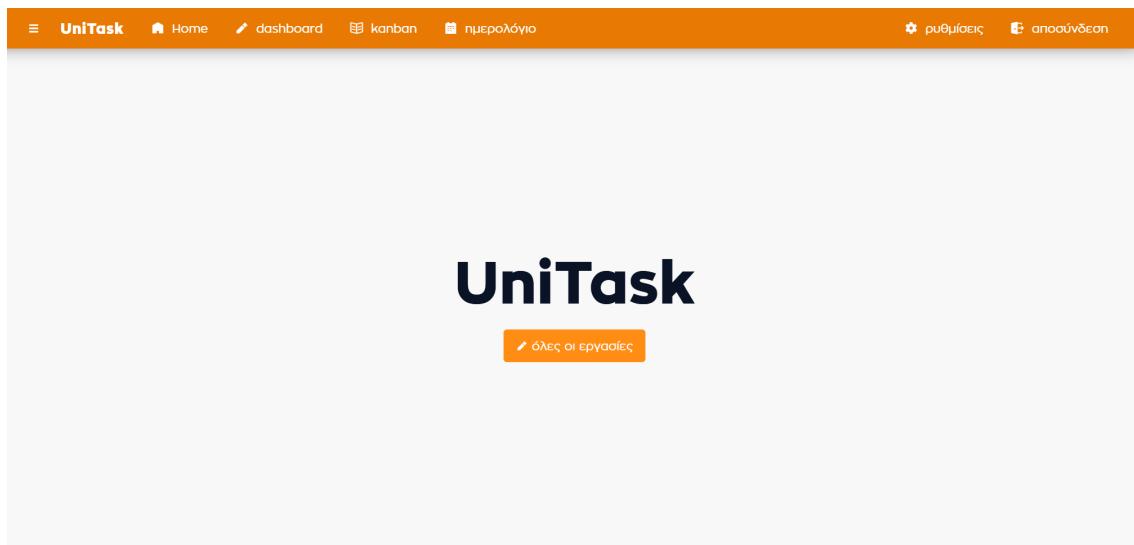
Σχήμα 5.8: Επεξεργασία στοιχείων χρήστη

Name	Last login	Blocked	Blocked since	Active	Web service user	Is anonymous
Foithths	Oxi	Nai		Oxi	Oxi	Oxi

Σχήμα 5.9: Αναζήτηση χρήστη

ένα κεντρικό call to action κουμπί (όλες οι εργασίες) το οποίο οδηγεί στο dashboard.

Στο σχήμα 5.11 εμφανίζεται η σελίδα dashboard. Πρόκειται για την κεντρική σελίδα προβολής δημιουργίας και επεξεργασίας των εργασιών του χρήστη. Περιλαμ-



Σχήμα 5.10: Αρχική σελίδα εφαρμογής

Σχήμα 5.11: Dashboard εργασιών

βάνονται τρεις καρτέλες (Επόμενες εργασίες, Εργασίες σε εξέλιξη, Ολοκληρωμένες εργασίες). Οι επόμενες εργασίες αφορούν εργασίες που έχουν σκοπό να πραγματοποιηθούν στο άμεσο μέλλον αλλά όχι τη δεδομένη χρονική στιγμή, οι εργασίες σε εξέλιξη αφορούν εργασίες που βρίσκονται σε εξέλιξη και οι ολοκληρωμένες εργασίες αφορούν εργασίες που έχουν ολοκληρωθεί.

Στη σελίδα επίσης περιλαμβάνεται ένα επεξηγηματικό παράθυρο που εμφανίζεται μόνο όταν ο χρήστης δεν έχει δημιουργήσει κάποια εργασία και του εξηγεί το τρόπο λειτουργίας της εφαρμογής. Στο dashboard επίσης περιλαμβάνονται μετρητές για το σύνολο των εργασιών που υπάρχουν ανά κατηγορία, όπως επίσης και ένα κεντρικό

κουμπί δημιουργίας εργασιών (Νέα εργασία).

The image displays two side-by-side screenshots of the UniTask application's interface, illustrating the process of creating new work items.

**Screenshot 1 (Top):** A modal window titled "Δημιουργία νέας εργασίας" (Create new job). The form fields include:

- Τίτλος:** Νέα εργασία
- Κατάσταση:** Επόμενη (blue icon), Σε εξέλιξη (blue icon with gear), Ολοκληρωμένη (orange icon with checkmark)
- Προτεραιότητα:** Χαμηλή (blue icon with downward arrow), Μεσαία (orange icon with double arrows), Υψηλή (blue icon with upward arrow)
- Χρώμα:** Μπλε
- Εκκίνηση:** 11/1/2025, 7:30 μμ.
- Λήξη:** 18/1/2025, 7:30 μμ.
- A note at the bottom states: "Η ημερομηνία έχει ήδη οριστεί σε μια εβδομάδα από τώρα!" (The date has already been set for a week ago!).

**Screenshot 2 (Bottom):** A modal window titled "Δημιουργία νέας εργασίας" (Create new job). The form fields include:

- Τίτλος:** Συγγραφή Διπλωματικής
- Κατάσταση:** Επόμενη (blue icon), Σε εξέλιξη (orange icon with gear), Ολοκληρωμένη (orange icon with checkmark)
- Προτεραιότητα:** Χαμηλή (blue icon with downward arrow), Μεσαία (orange icon with double arrows), Υψηλή (blue icon with upward arrow)
- Χρώμα:** Μπλε
- Εκκίνηση:** 1/12/2024, 7:30 μμ.
- Λήξη:** 1/2/2025, 7:30 μμ.

Σχήμα 5.12: Δημιουργία νέας εργασίας

Πατώντας το, εμφανίζεται ένα αναδυόμενο παράθυρο (σχήμα 5.12.1) με μια φόρμα για τη δημιουργία μιας νέας εργασίας. Η φόρμα αυτή περιλαμβάνει πεδία για την εισαγωγή του τίτλου της εργασίας (Τίτλος), της ημερομηνίας έναρξης (Εκκίνηση) και λήξης (Λήξη), της κατάστασης της εργασίας (Κατάσταση), της προτεραιότητας της εργασίας (Προτεραιότητα) όπως επίσης και του χρώματος της εργασίας (Χρώμα), όπως θα αποτυπωθεί μετέπειτα στο ημερολόγιο. Στο αναδυόμενο παράθυρο ορίζεται προεπιλεγμένα ως ημερομηνία λήξης της εργασίας μια εβδομάδα μετέπειτα

από την ημερομηνία δημιουργίας της, ενώ η κατάσταση της εργασίας έχει προκαθοριστεί (γίνεται να τροποποιηθεί φυσικά) ανάλογα με το ποια καρτέλα ήταν ανοιχτή στο dashboard. Αφού επεξεργαστούμε το παράθυρο όπως επιθυμούμε (σχήμα 5.12.2), πατάμε αποθήκευση για την αποθήκευση της εργασίας.

Σημειώνεται ότι οι επιλογές για την κατάσταση και την προτεραιότητα της εργασίας είναι χρωματικά κωδικοποιημένες (color-coded) και συνοδεύονται από γραφικά σύμβολα, όπως φαίνεται στο σχήμα 5.13 με σκοπό να διευκολύνει τον χρήστη για την άμεση αναγνώριση τους.



Σχήμα 5.13: Επιλογές κατάστασης και προτεραιότητας εργασίας

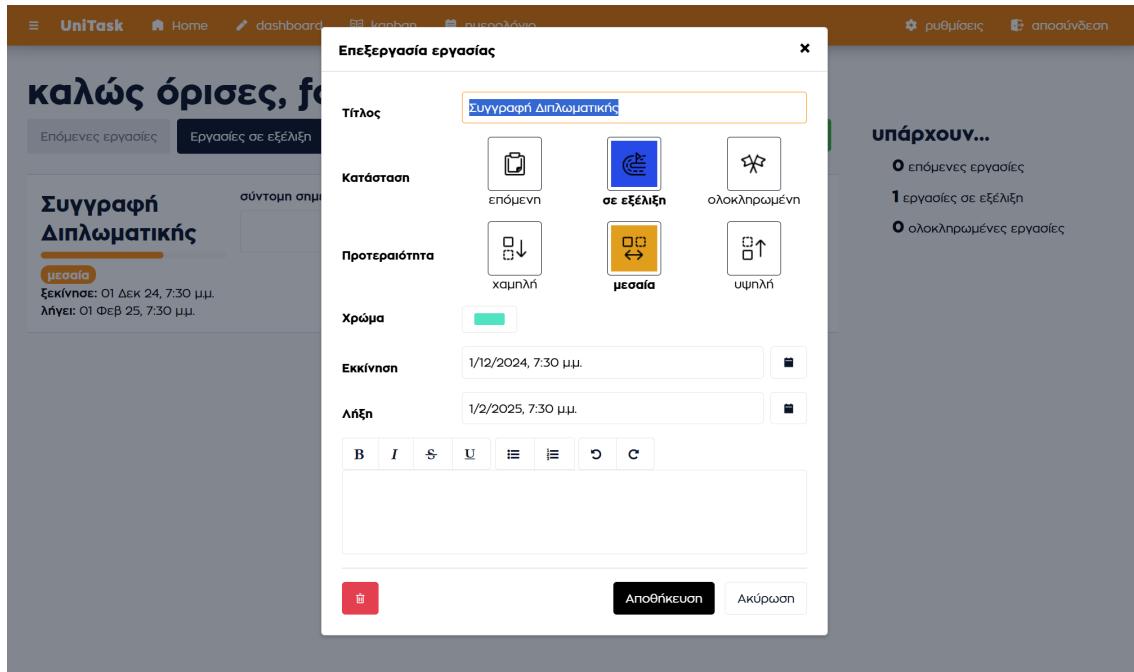
The screenshot shows a dashboard interface for 'UniTask'. At the top, there's a navigation bar with links for Home, dashboard, kanban, and ημερολόγιο. On the right, there are links for ρυθμίσεις and αποσύνδεση. Below the navigation, the main area has a title 'καλώς όρισες, foithths'. There are three tabs: 'Επόμενες εργασίες' (selected), 'Εργασίες σε εξέλιξη', and 'Ολοκληρωμένες εργασίες'. A green button '+ Νέα εργασία' is visible. To the right, a sidebar titled 'υπάρχουν...' lists: 0 επόμενες εργασίες, 1 εργασίες σε εξέλιξη, and 0 ολοκληρωμένες εργασίες. The main content area shows a card for a task named 'Συγγραφή Διπλωματικής' with a progress bar at 100%. It includes a note: 'μεσαία Έκκινηση: 01 Δεκ 24, 7:30 μμ. Λήγει: 01 Φεβ 25, 7:30 μμ.'.

Σχήμα 5.14: Dashboard με δημιουργημένη εργασία

Στη σελίδα πλέον είναι δημιουργημένη η κάρτα με την πρώτη μας εργασία (σχήμα 5.14). Το πλαίσιο σύντομη σημείωση επιτρέπει την εισαγωγή μιας συνοπτικής περιγραφής της εργασίας, στα αριστερά εμφανίζεται ο τίτλος της εργασίας, μια μπάρα προόδου (progress bar) η οποία δυναμικά αυξάνεται όσο πλησιάζουμε στη λήξη της εργασίας, όπως επίσης η προτεραιότητα της εργασίας και οι ημερομηνίες και ώρες εκκίνησης και λήξης της εργασίας, ενώ στο δεξιό μέρος της κάρτας εμφανίζονται το εικονίδιο για την επεξεργασία της εργασίας.

Η ταξινόμηση των εργασιών στο dashboard βασίζεται στην προτεραιότητα και

στον χρόνο λήξης τους. Έτσι μια εργασία με υψηλή προτεραιότητα θα εμφανίζεται φηλότερα από μια εργασία με χαμηλή προτεραιότητα που δε λήγει σύντομα.



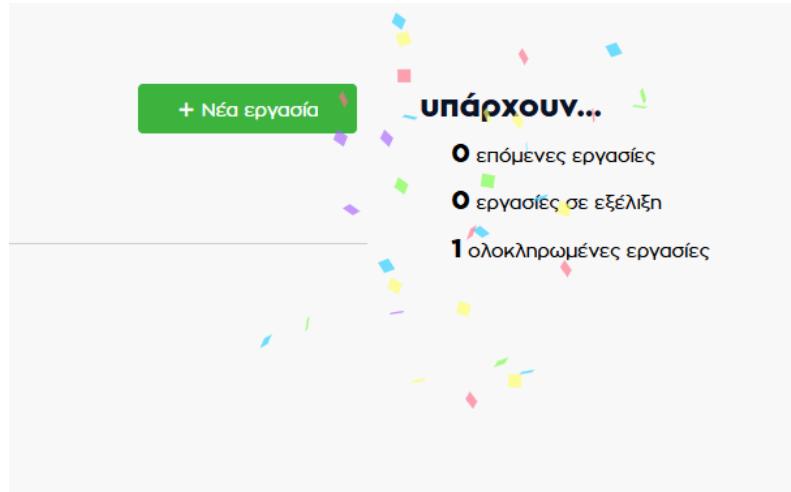
Σχήμα 5.15: Επεξεργασία εργασίας

Με την επιλογή του εικονιδίου επεξεργασίας, εμφανίζεται το αναδυόμενο παράθυρο (σχήμα 5.15), το οποίο επιτρέπει την τροποποίηση των στοιχείων της εργασίας, συμπεριλαμβανομένου του πλαισίου σύντομη σημείωσης. Το ίδιο παράθυρο περιλαμβάνει και το κουμπί διαγραφής της εργασίας.

Για την καλύτερη διευκόλυνση του χρήστη, την τελευταία εβδομάδα πριν λήξη της προθεσμίας εμφανίζεται ενημερωτικό κείμενο στην κάρτα (σχήμα 5.16.1), ενώ όταν η εργασία έχει λήξει, εμφανίζεται call-to-action κουμπί που μπορεί να μετακινήσει αυτήν την εργασία στις ολοκληρωμένες εργασίες (σχήμα 5.16.2).

Σχήμα 5.16: Ενημερωτικό κείμενο κάρτας εργασίας

Επίσης, κατά την αλλαγή κατηγορίας της εργασίας σε ολοκληρωμένη, εμφανίζονται κομφετί ως ένα σύστημα ανταμοιβής για τον χρήστη (σχήμα 5.17).



Σχήμα 5.17: Επιβράβευση όταν ολοκληρωθεί μια εργασία

Σχήμα 5.18: Σελίδα Kanban

Η σελίδα **Kanban** του σχήματος 5.18 εμφανίζει μια διαφορετική παρουσίαση στις εργασίες με τον τρόπο που έχει εξηγηθεί στην ενότητα 2.4.3. Στον πίνοντα Kanban οι εργασίες χωρίζονται σε τρεις κατηγορίες: τις εργασίες που έχουν ολοκληρωθεί, τις εργασίες που βρίσκονται σε εξέλιξη και τις εργασίες που έχουν σκοπό να πραγματοποιηθούν στο μέλλον. Κάθε στήλη περιλαμβάνει ένα κουμπί δημιουργίας νέας εργασίας που αυτόματα καθορίζει και την κατηγορία της.

Οι εργασίες απεικονίζονται ως κάρτες που περιλαμβάνουν τον τίτλο, την προτεραιότητα, καθώς και την ημερομηνία έναρξης και λήξης τους, ενώ πατώντας πάνω

στην κάρτα μιας εργασίας, εμφανίζεται το αναδυόμενο παράθυρο επεξεργασίας της εργασίας του σχήματος 5.15.

Στη σελίδα **ημερολόγιο** (σχήμα 5.19) εμφανίζεται ένα ημερολόγιο με τις εργασίες του χρήστη. Οι εργασίες εμφανίζονται στο ημερολόγιο με το χρώμα που έχει οριστεί στην επιλογή του χρήστη κατά τη δημιουργία της εργασίας, υπάρχουν προβολές ανά ημέρα, εβδομάδα ή μήνα ενώ εμφανίζεται και ένα σύντομο παράρτημα στα δεξιά με τη λίστα των εργασιών.

Σχήμα 5.19: Σελίδα Calendar

Η σελίδα **ρυθμίσεις** (σχήμα 5.20) παρέχει επιλογές για τη μαζική διαγραφή εργασιών ή την ενεργοποίηση της λειτουργίας γρήγορης διαγραφής. Αυτή η λειτουργία εισάγει ένα κουμπί διαγραφής στις κάρτες εργασιών στο dashboard, επιτρέποντας την άμεση διαγραφή των εργασιών που θέλουμε χωρίς να χρειάζεται να μεταβούμε στη σελίδα επεξεργασίας κάθε εργασίας (σχήμα 5.22.1). Τέλος, παρέχεται η δυνατότητα αρχικοποίησης των εργασιών. Στην αρχικοποίηση διαγράφονται οι υπάρχουσες εργασίες του χρήστη και δημιουργούνται κάποιες προκαθορισμένες οι οποίες λειτουργούν ως ένα demo (σχήμα 5.22).

Σε κάθε ρύθμιση εμφανίζονται επιβεβαιωτικά αναδυόμενα μηνύματα προκειμένου να αποφευχθούν ακούσιες διαγραφές εργασιών (σχήμα 5.21).

### 5.3 Δομή της εφαρμογής

Πέρα από τα προκατασκευασμένα modules του Mendix, η λειτουργικότητα της εφαρμογής έχει οργανωθεί σε τρία modules, το Administrator, το TaskManager και

Σχήμα 5.20: Σελίδα ρυθμίσεων

Σχήμα 5.21: Σελίδα ρυθμίσεων

το UniTask.

### 5.3.1 Module Administrator

To Administrator περιλαμβάνει τη λειτουργικότητα που αφορά τη διαχείριση των χρηστών της εφαρμογής. Όλες οι οντότητες του domain model, οι σελίδες και τα microflows του module έχουν δικαιώματα ανάγνωσης και εγγραφής από τον Administrator ρόλο, όπως ορίζεται στο Security της εφαρμογής.

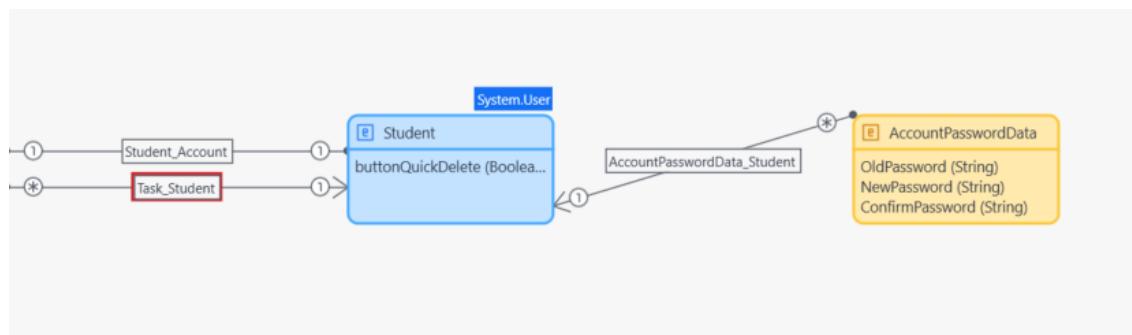
**Dashboard**

**Kanban**

**Ημερολόγιο**

Σχήμα 5.22: Οι σελίδες Dashboard, Kanban και Calendar μετά την αρχικοποίηση των εργασιών. Στο Dashboard φαίνεται η λειτουργία γρήγορης διαγραφής εργασιών.

### 5.3.1.1 Domain model του Administrator



Το domain model του Administrator περιλαμβάνει την οντότητα **Student** που αληρονομεί την οντότητα **System.User** του Mendix. Η οντότητα περιγράφει τον κάθε χρήστη της εφαρμογής και περιλαμβάνει την Boolean ιδιότητα `buttonQuickDelete` αρχικοποιημένη σε `False` η οποία χρησιμοποιείται για την ενεργοποίηση της λειτουργίας γρήγορης διαγραφής εργασιών. Η οντότητα **Student** συσχετίζεται με την οντότητα **Account** του **System.User** με σχέση 1-προς-1, η οντότητα **Task** του **TaskManager** με σχέση ένα-προς-πολλά (ένα **Student** συσχετίζεται με πολλά **Tasks**) και την οντότητα **AccountPasswordData** με σχέση 1-προς-πολλά (ένα **Student** συσχετίζεται με πολλά **AccountPasswordData**).

Η οντότητα **AccountPasswordData** είναι μη-διατηρήσιμη οντότητα (δεν αποθηκεύεται στη βάση δεδομένων αλλά μόνο στη μνήμη) και περιλαμβάνει τις ιδιότητες `OldPassword`, `NewPassword` και `ConfirmPassword` και χρησιμοποιείται για την αλλαγή του κωδικού πρόσβασης του χρήστη.

### 5.3.1.2 Σελίδες του Administrator

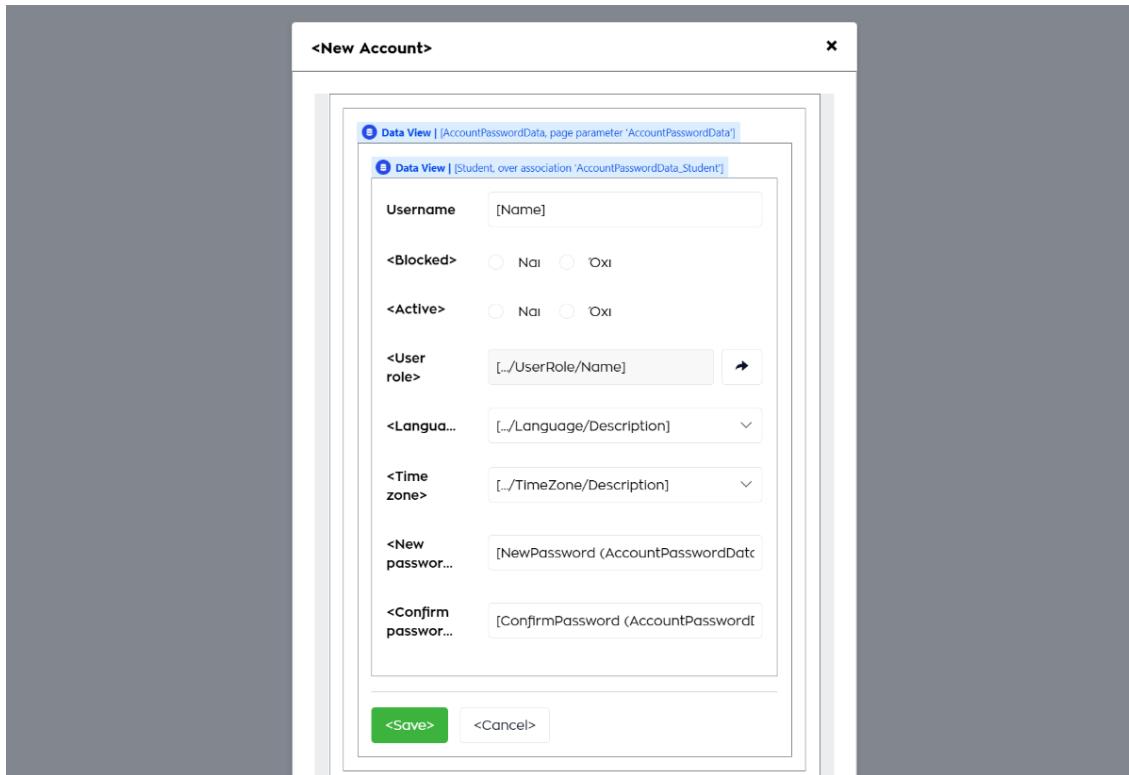
Στο Administrator περιλαμβάνονται οι εξής σελίδες:

**Account\_Overview**

Name	<Last login>	<Blocked>	<Blocked since>	<Active>	<Web service user>	<Is anonymous>
[Name]	[LastLogin]	[Blocked]	[BlockedSince]	[Active]	[WebServiceUser]	[IsAnonymous]
[Name]	[LastLogin]	[Blocked]	[BlockedSince]	[Active]	[WebServiceUser]	[IsAnonymous]
[Name]	[LastLogin]	[Blocked]	[BlockedSince]	[Active]	[WebServiceUser]	[IsAnonymous]
[Name]	[LastLogin]	[Blocked]	[BlockedSince]	[Active]	[WebServiceUser]	[IsAnonymous]

Η σελίδα χρησιμοποιείται για τη διαχείριση των χρηστών της εφαρμογής από την πλευρά των διαχειριστών.

Χρησιμοποιείται το UniTask\_SideBar layout του UniTaskDesignSystem module. Το κύριο μέρος της σελίδας αποτελείται από ένα Data Grid με Data source την οντότητα Student και με στήλες τις ιδιότητες Name, Last login, Blocked, Blocked since, Active, Web service user και Is anonymous.

**Account\_New**

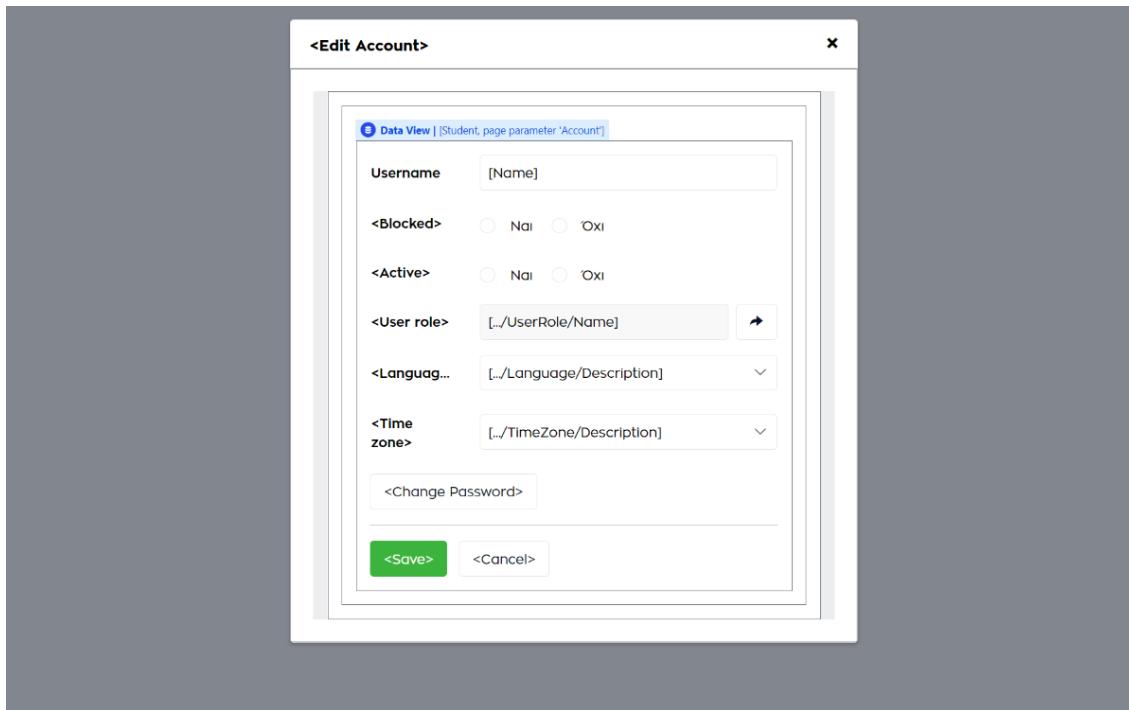
Η σελίδα χρησιμοποιείται για τη δημιουργία νέων χορηγών της εφαρμογής.

Χρησιμοποιείται το PopupLayout layout του `Atlas_Core` module. Η σελίδα περιλαμβάνει δύο Parameters, το `Student` και `AccountPasswordData` του module `Administrator`. Η σελίδα αποτελείται από δύο εμφωλευμένα Data Views, το εξωτερικό έχει ως Data source το `AccountPasswordData`, ενώ το εσωτερικό έχει ως Data source τη συσχέτιση του `AccountPasswordData` με το `Student`. Η χρήση του `AccountPasswordData` είναι απαραίτητη καθώς η δημιουργία ενός νέου χορήστη χρειάζεται την αποθήκευση του κωδικού πρόσβασής του.

Στο εσωτερικό Data View περιλαμβάνει Text Boxes, Radio Buttons και Input Reference Set Selectors όπου εισάγονται τιμές για τα `Username`, `Blocked`, `Active`, `User role`, `Language`, `Time zone`, `New password` και `Confirm password`. Έχει σημασία να σημειωθεί πως οι ιδιότητες (γνωρίσματα) που αποθηκεύουμε στην πραγματικότητα δεν είναι ιδιότητες του `Student` αλλά του `System.User` του οποίου αποτελεί παιδί. Το Input Reference Set Selector χρησιμοποιείται για την επιλογή του `UserRole`, που αποτελεί διαφορετική σελίδα που θα αναλυθεί στη συνέχεια.

Τέλος, περιλαμβάνεται κουμπί για την αποθήκευση, το οποίο καλεί το microflow `ACT_Account_Save` του `Administrator` για την αποθήκευση των τιμών, και κουμπί για την ακύρωση της διαδικασίας.

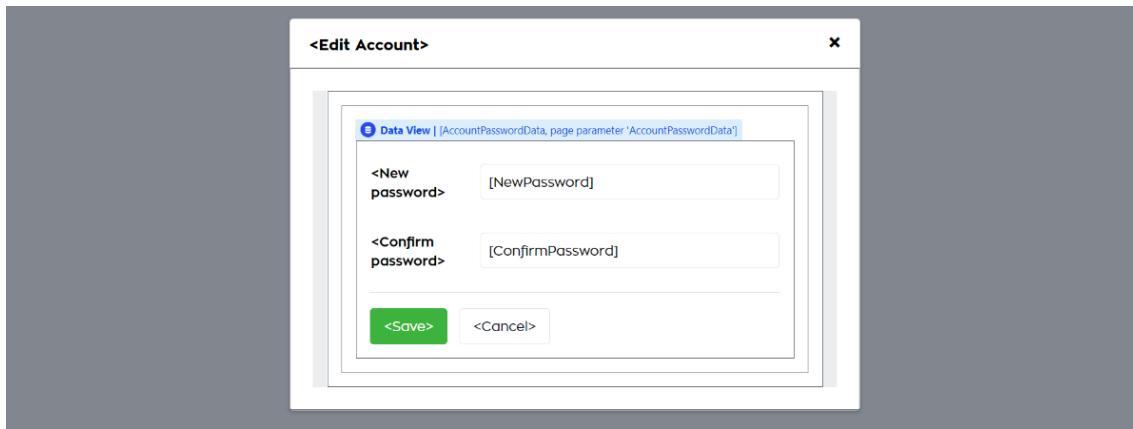
### Account\_Edit



Η σελίδα χρησιμοποιείται για την επεξεργασία υπαρχόντων χρηστών της εφαρμογής.

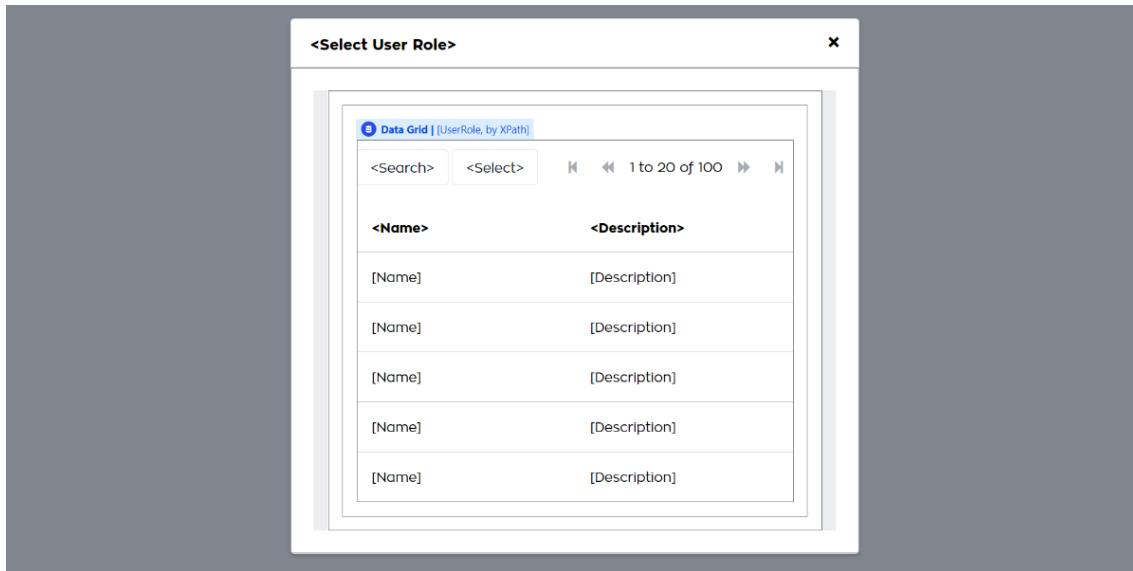
Χρησιμοποιείται το PopupLayout. Η σελίδα περιλαμβάνει το Parameter Student. Η σελίδα αποτελείται από ένα Data View με Data source το Student με παρόμοια Text Boxes και Radio Buttons όπως και το Account\_New. Επίσης, περιλαμβάνεται το κουμπί που καλεί το microflow ACT\_Password\_Change για την αλλαγή κωδικού.

Τέλος, περιλαμβάνεται κουμπί για την αποθήκευση και κουμπί για την ακύρωση της διαδικασίας. Τα κουμπιά καλούν προεπιλεγμένες ενέργειες του Mendix.

**Change\_Password**

Η σελίδα χρησιμοποιείται για την αλλαγή του κωδικού πρόσβασης υπάρχοντος χρήστη.

Χρησιμοποιείται το PopupLayout. Η σελίδα περιλαμβάνει το Parameter AccountPasswordData, ένα Data View με Data source το Student με τα απαραίτητα Text Boxes για την αλλαγή των τιμών. Τέλος, περιλαμβάνεται κουμπί για την αποθήκευση που καλεί το microflow ChangePassword του Administrator και κουμπί για την ακύρωση της διαδικασίας.

**UserRole\_Select**

Η σελίδα χρησιμοποιείται για την επιλογή του ρόλου του χρήστη κατά τη δημιουργία νέου χρήστη.

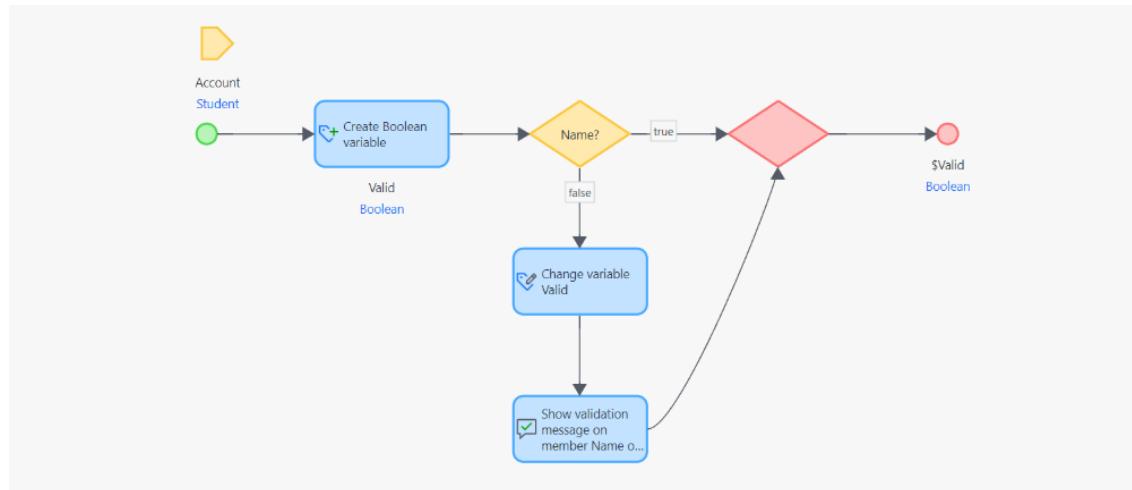
Χρησιμοποιείται το PopupLayout. Η σελίδα αποτελείται από ένα Data Grid με

Data source το UserRole του System.<sup>1</sup>

### 5.3.1.3 Microflows του Administrator

Στο Administrator περιλαμβάνονται τα εξής microflows:

**VAL\_Account**



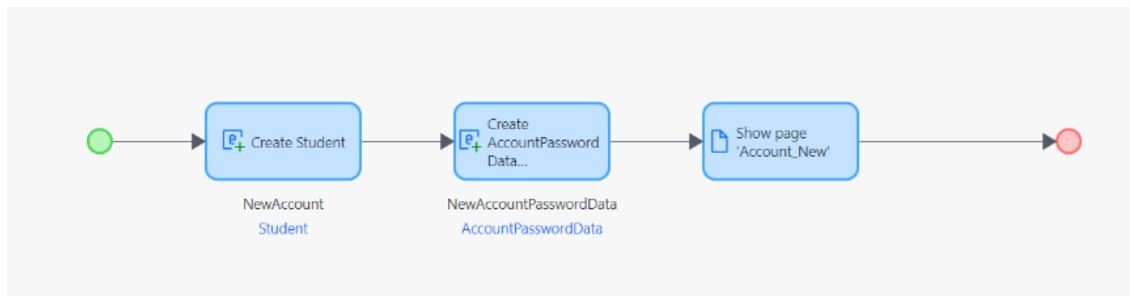
Το microflow καλείται από το microflow ACT\_Account\_Save για να επικυρώσει τον λογαριασμό του χρήστη πριν αποθηκευτούν οι τιμές του.<sup>2</sup>

Αρχικά δημιουργείται μια boolean μεταβλητή Valid με αρχική τιμή True η οποία θα επιστραφεί από το microflow. Στη συνέχεια ελέγχεται αν για το αντικείμενο Account τύπου Student ισχύει η συνθήκη (`trim($Account/Name) != ''`). Η έκφραση στη συνθήκη αφού καθαρίσει τα κενά (whitespaces) από το Name του Account, ελέγχει αν είναι διαφορετικό από το κενό string. Αν η συνθήκη δεν ισχύει, τότε η μεταβλητή Valid γίνεται False, η οποία επιστρέφεται μαζί με ένα popout μήνυμα. Αν η συνθήκη ισχύει, δηλαδή αν υπάρχει όνομα, τότε επιστρέφεται True.

<sup>1</sup>Τεχνικά, λόγω των ρόλων χρηστών που έχουν κληρονομηθεί από το Mendix περιλαμβάνεται και ο ρόλος Guest, ο οποίος στην πράξη δε χρησιμοποιείται καθώς έχει πρόσβαση μόνο στη σελίδα σύνδεσης.

<sup>2</sup>Το πρόθεμα VAL χρησιμοποιείται στην ονομασία των microflows για να δηλώσει επικύρωση (validation).

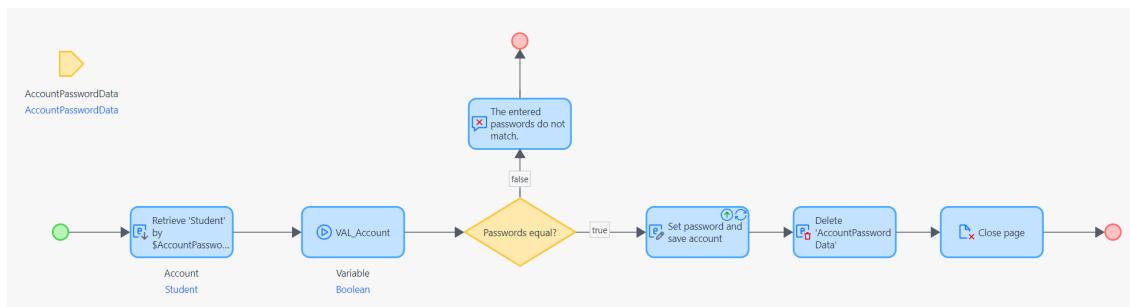
### ACT\_Account\_New



To microflow καλείται από τη σελίδα Account\_Overview με σκοπό τη δημιουργία ενός νέου χρήστη.<sup>3</sup>

Αρχικά δημιουργούνται δύο στιγμιότυπα τύπου Student και AccountPasswordData με ονόματα NewAccount και NewAccountPasswordData αντίστοιχα. Να σημειωθεί πως το NewAccountPasswordData συσχετίζεται με το Student. Τα αντικείμενα δε γίνονται commit ακόμα στη βάση, καθώς είναι κενά. Στη συνέχεια εμφανίζεται η σελίδα Account\_New με τα αντικείμενα NewAccount και NewAccountPasswordData ως Parameters.

### ACT\_Account\_Save



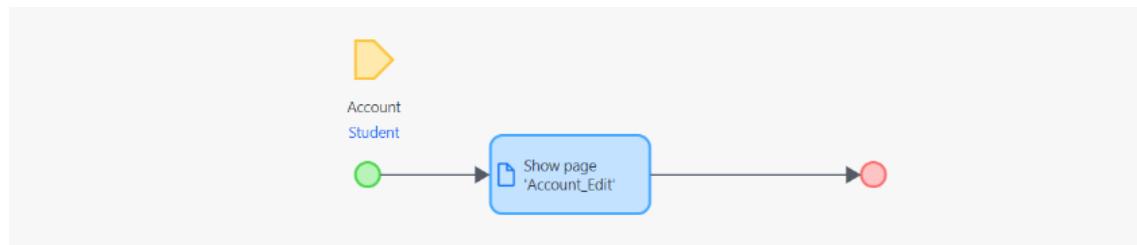
To microflow καλείται από τη σελίδα Account\_New με σκοπό την αποθήκευση των τιμών του νέου χρήστη.

To microflow έχει ως Parameter το AccountPasswordData. Μαζί με αυτό, ανακτάται το Student αφού συσχετίζονται, και καλείται το microflow VAL\_Account το οποίο ελέγχει αν το Name του Student είναι κενό. Αν το Name είναι κενό, τότε εμφανίζεται ένα popout μήνυμα και το microflow τερματίζεται. Αν το Name δεν είναι κενό, τότε ελέγχεται αν το newPassword του AccountPasswordData είναι ίσο με το ConfirmPassword, όπως έχουν δοθεί στη φόρμα Account\_New. Αν η συνθήκη δεν ισχύει, τότε εμφανίζεται ένα popout μήνυμα και το microflow τερματίζεται. Αν η

<sup>3</sup>To πρόθεμα ACT χρησιμοποιείται στην ονομασία των microflows για να δηλώσει μια ενέργεια (action).

συνθήκη ισχύει, τότε το NewPassword γίνεται commit στο Account τύπου Student στο γνώρισμα Password το οποίο είναι Hashed string. Στη συνέχεια το αντικείμενο AccountPasswordData διαγράφεται και κλείνει η σελίδα.

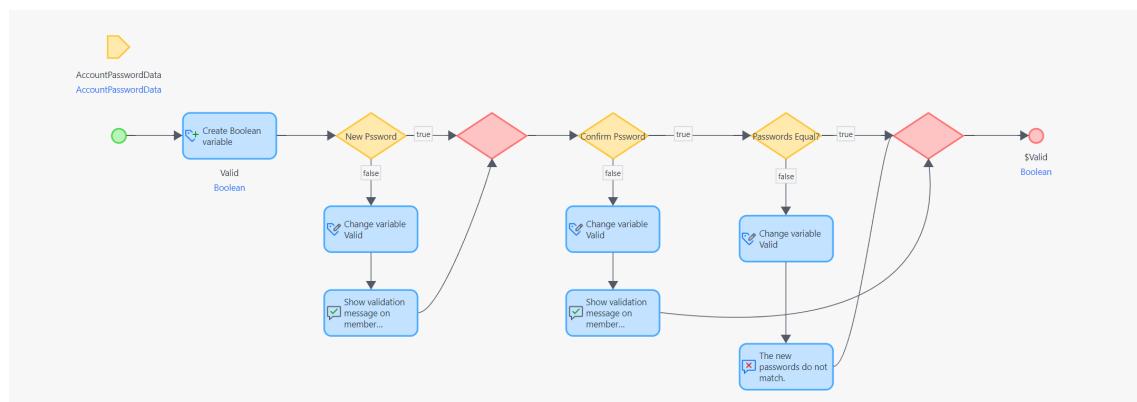
#### **ACT\_Account\_Edit**



To microflow καλείται από τη σελίδα Account\_Overview με σκοπό την επεξεργασία ενός υπάρχοντος χρήστη.

To microflow εμφανίζει τη σελίδα Account\_Edit με το Account τύπου Student ως Parameter.

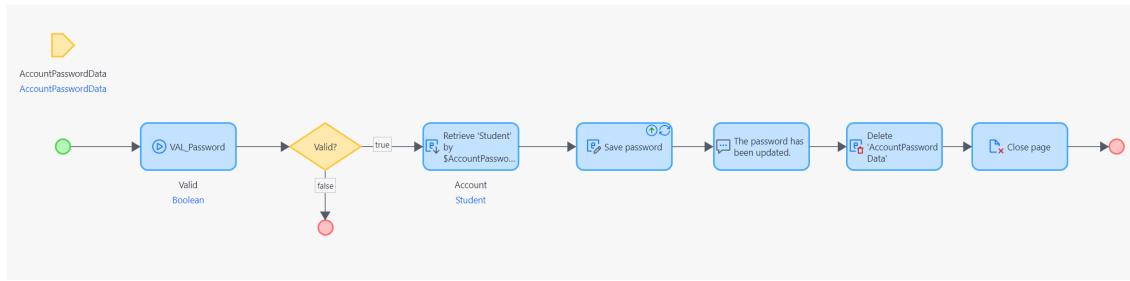
#### **VAL\_Password**



To microflow καλείται από το microflow ChangePassword με σκοπό τον έλεγχο των συνθηκών για την αλλαγή του κωδικού πρόσβασης.

Αρχικά δημιουργείται μια boolean μεταβλητή Valid με αρχική τιμή True. Στη συνέχεια ελέγχεται αν για το NewPassword του αντικειμένου AccountPasswordData ισχύει η συνθήκη (trim(\$AccountPasswordData/NewPassword) != ''). Η έκφραση στη συνθήκη ελέγχει αν έχει δοθεί όντως νέος κωδικός στη φόρμα της σελίδας Change>Password. Αν η συνθήκη ισχύει, ελέγχεται με παρόμοιο τρόπο και το ConfirmPassword όπως επίσης και το αν είναι ίσο με το NewPassword. Αν κάποια συνθήκη από τις προαναφερθείσες δεν ισχύει, η μεταβλητή Valid γίνεται False και εμφανίζεται κατάλληλο πορούτ μήνυμα. Αν όλες οι συνθήκες ισχύουν, τότε η μεταβλητή Valid παραμένει True και επιστρέφεται από το microflow.

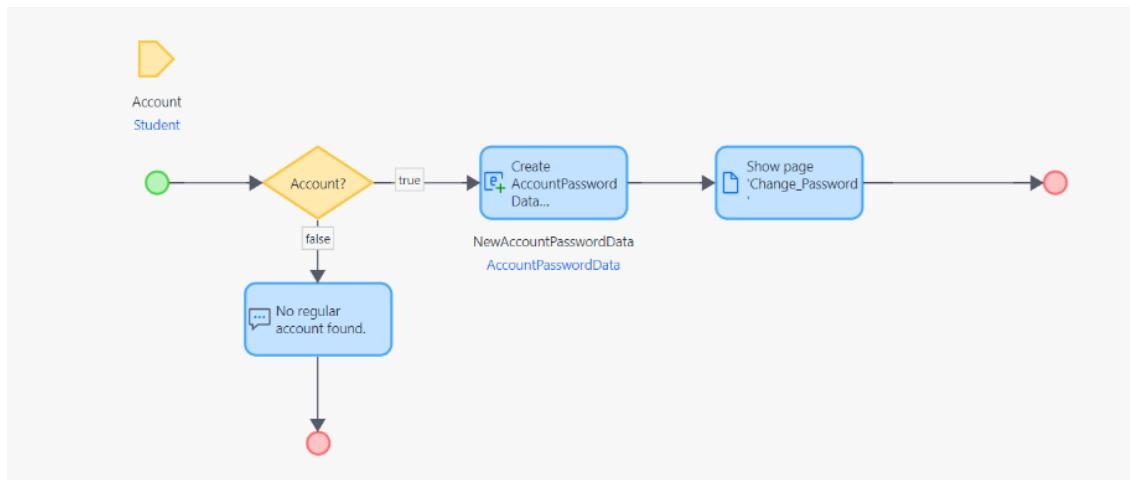
### ChangePassword



To microflow καλείται από τη σελίδα `Change_Password` με σκοπό την αποθήκευση του νέου κωδικού πρόσβασης για έναν υπάρχων χρήστη.

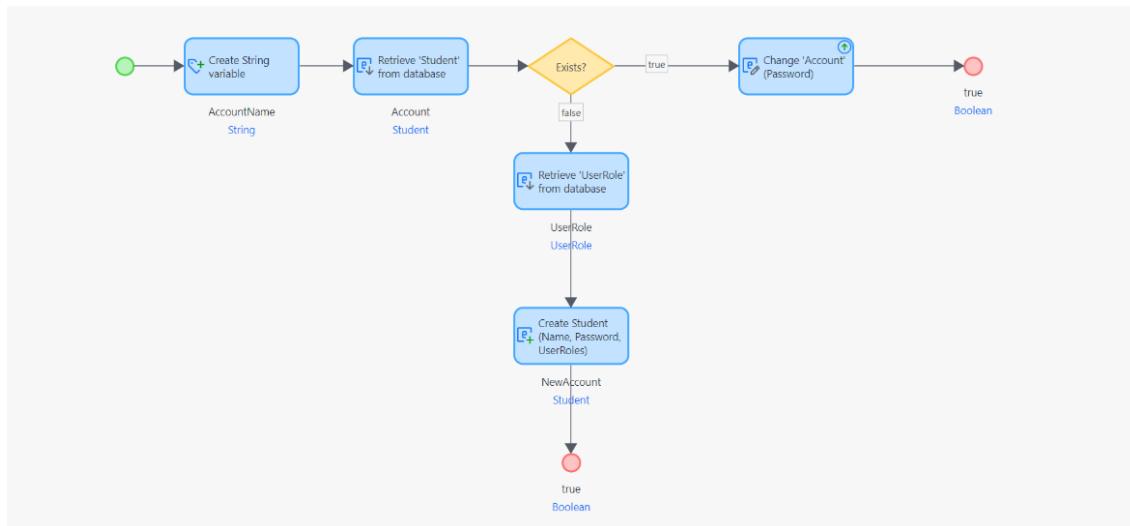
To microflow έχει ως Parameter το `AccountPasswordData`. Στην αρχή καλείται το microflow `VAL_Password` για τον έλεγχο των συνθηκών. Αν η μεταβλητή `Valid` που επιστρέφεται από το microflow είναι `False`, τότε το microflow τερματίζεται. Αν είναι `True`, τότε γίνεται `retrieve` και το αντικείμενο `Student` ως συσχέτιση, γίνεται `commit` το `NewPassword` ως `Hashed string Password` στο `Student`, εμφανίζεται κατάλληλο `popup` μήνυμα, διαγράφεται το `AccountPasswordData` και κλείνει η σελίδα.

### ACT\_Password\_Change



To microflow καλείται από τη σελίδα `Account_Edit` με σκοπό την αλλαγή του κωδικού πρόσβασης ενός υπάρχοντος χρήστη.

Με Parameter το `Account` τύπου `Student`, αρχικά ελέγχεται αν υπάρχει όντως κάποιο υπαρκτό `Account`. Αν δεν υπάρχει, εμφανίζεται `popup` μήνυμα και το microflow τερματίζεται. Αν υπάρχει, τότε δημιουργείται ένα νέο αντικείμενο `AccountPasswordData` συσχετισμένο με το `Account` και εμφανίζεται η σελίδα `Change_Password` με το `AccountPasswordData` και `Account` ως Parameters.

**ASU\_Administrator\_Create**

Microflow που καλείται κατά την αρχικοποίηση της εφαρμογής για τη δημιουργία του διαχειριστή της εφαρμογής.<sup>4</sup>

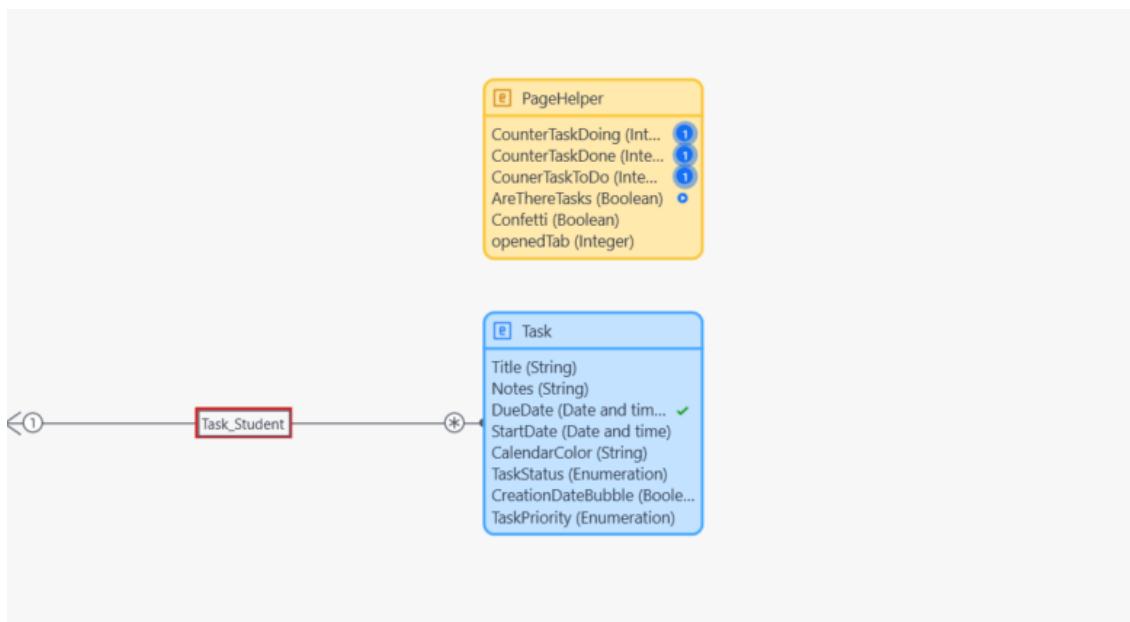
Αρχικά δημιουργείται το String AccountName με τιμή 'admin' με το username του διαχειριστή. Στη συνέχεια γίνεται retrieve από τη βάση δεδομένων το Student που έχει ως Name το AccountName. Αν δεν υπάρχει τέτοιος λογαριασμός, τότε γίνονται retrieve τα System.UserRoles και δημιουργείται ένα νέο αντικείμενο Student με Name το AccountName, Password η τιμή 'admin' και UserRole το UserRole. Αν υπάρχει ήδη λογαριασμός με το AccountName, τότε αλλάζει ο κωδικός σε 'admin'. Είναι προφανές ότι τα στοιχεία σύνδεσης του διαχειριστή έχουν οριστεί ως 'admin' και 'admin'.

### 5.3.2 Module TaskManager

Το TaskManager περιλαμβάνει τη λειτουργικότητα που αφορά τη διαχείριση των εργασιών της εφαρμογής. Όλες οι οντότητες του domain model, οι σελίδες και τα microflows του module έχουν δικαιώματα ανάγνωσης και εγγραφής από τον User ρόλο, όπως ορίζεται στο Security της εφαρμογής, με εξαίρεση το Custom\_Login\_-Page που έχει δικαίωμα ο Guest.

<sup>4</sup>Το πρόθεμα ASU (After Startup) χρησιμοποιείται στην ονομασία των microflows για να δηλώσει ότι καλείται αμέσως μετά την εκκίνηση της εφαρμογής.

### 5.3.2.1 Domain model του TaskManager



Το domain model του TaskManager περιλαμβάνει την οντότητα Task και τη μηδιατηρήσιμη οντότητα PageHelper.

Η οντότητα Task αναπαριστά την εκάστοτε εργασία του χρήστη της εφαρμογής. Περιλαμβάνει τις ιδιότητες Title τύπου String ως 200 χαρακτήρες με το όνομα της εργασίας, Notes τύπου String με απεριόριστους χαρακτήρες όπου μπορούν να προστεθούν σημειώσεις για αυτή και DueDate τύπου Date and time με την ημερομηνία λήξης. Επίσης, περιλαμβάνει τη StartDate τύπου Date and time με την ημερομηνία έναρξης, η οποία αρχικοποιείται με την τιμή '%CurrentDateTime%' (Token που επιστρέφει την τρέχουσα ημερομηνία και ώρα) και τη CalendarColor τύπου String που αποθηκεύει το χρώμα της εργασίας στο ημερολόγιο. Λόγω της φύσης του widget του ημερολογίου, το String θα έχει πάντα τη μορφή 'rgb(<0-255>, <0-255>, <0-255>)' και στην οντότητα αρχικοποιείται με την τιμή 'rgb(38, 74, 229)' που αντιστοιχεί στο μπλε χρώμα.

Επιπλέον, η οντότητα περιλαμβάνει τις ιδιότητες TaskStatus και TaskPriority όπου είναι Enumeration ιδιότητες των Enumeration εγγράφων TaskStatus και TaskPriority, αρχικοποιημένες με To\_Do και Low αντίστοιχα. Το TaskStatus καθορίζει την κατάσταση της εργασίας με τις δυνατές καταστάσεις να είναι 'To\_Do', 'Doing' και 'Done' ενώ το TaskPriority καθορίζει αν η προτεραιότητα της εργασίας είναι χαμηλή, μεσαία ή υψηλή. Επίσης, περιλαμβάνεται η ιδιότητα CreationDateBubble τύπου Boolean που αρχικοποιείται με True. Η ιδιότητα αυτή χρησιμοποιείται για την εμφάνιση επεξηγηματικού μηνύματος στον χρήστη όταν δημιουργεί μια νέα εργασία. Να σημειωθεί επίσης πως το DateDue περιλαμβάνει ένα Validation rule που ελέγχει αν η ημερομηνία λήξης είναι μετά την ημερομηνία έναρξης StartDate, και αν δεν ισχύει, τότε εμφανίζεται κατάλληλο μήνυμα.

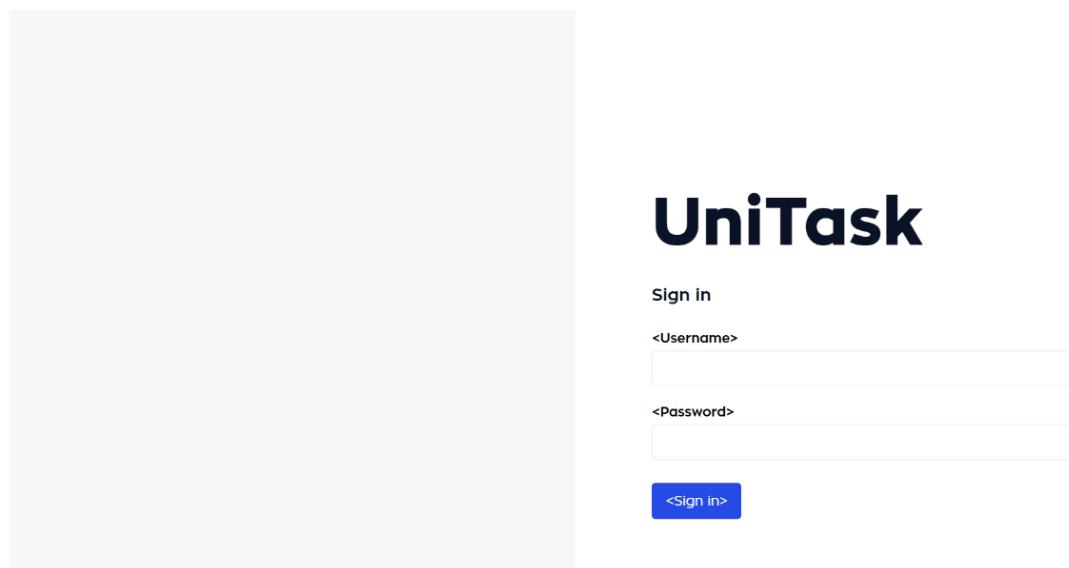
Η οντότητα PageHelper περιλαμβάνει τις ιδιότητες CounterTaskDoing,

CounterTaskDone και CounterTaskToDo τύπου Integer, των οποίων οι τιμές καθορίζονται από τα microflows CounterTaskDoing, CounterTaskDone και CounterTaskToDo αντίστοιχα. Οι ιδιότητες αυτές χρησιμοποιούνται για την εμφάνιση των τριών μετρητών. Επίσης, περιλαμβάνει την Boolean ιδιότητα AreThereTasks που καθορίζεται από το microflow AreThereTasks και χρησιμεύει για την εμφάνιση του επεξηγηματικού παραθύρου στο Dashboard, την Boolean ιδιότητα Confetti που αρχικοποιείται με False και χρησιμοποιείται για την εμφάνιση του κομφετί (σύστημα επιβράβευσης), και τέλος την ιδιότητα openedTab τύπου Integer αρχικοποιημένη με μηδέν που χρησιμεύει για την αποθήκευση της τρέχουσας καρτέλας του Dashboard που έχει ανοίξει ο χρήστης.

### 5.3.2.2 Σελίδες του TaskManager

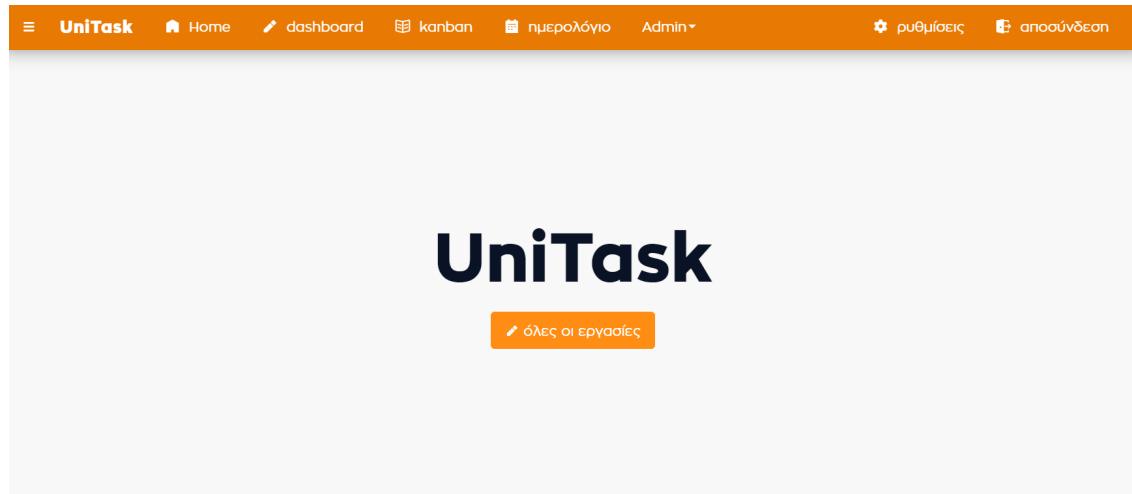
Στο TaskManager περιλαμβάνονται οι εξής σελίδες:

**Custom\_LogIn\_Overview**



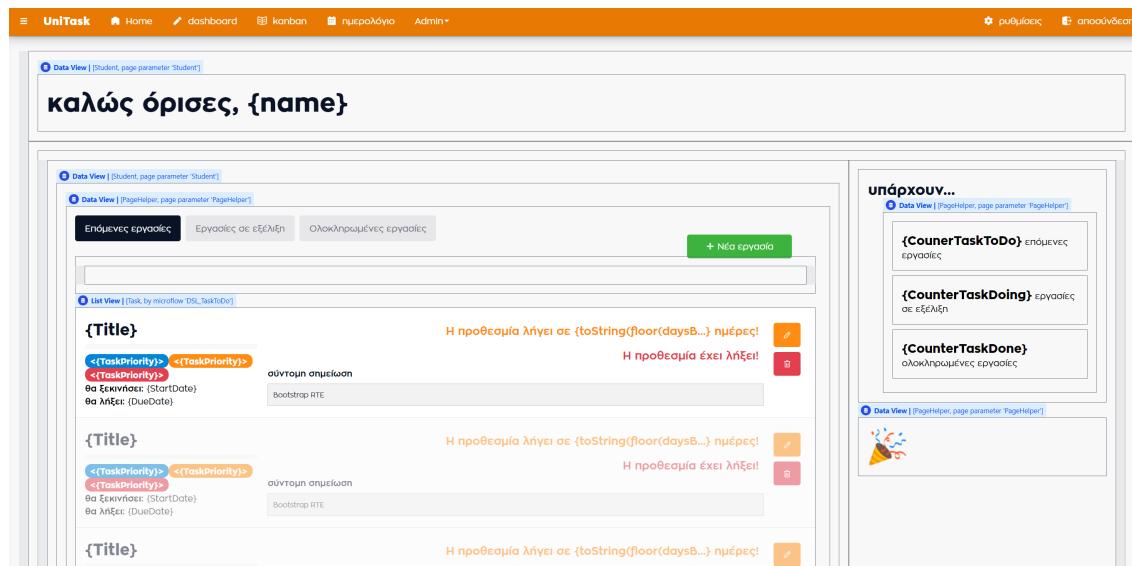
Η σελίδα χρησιμοποιείται για την είσοδο του χρήστη στην εφαρμογή. Πρόσβαση στη σελίδα έχουν μόνο οι Guest χρήστες.

Χρησιμοποιείται το Layout\_LogIn layout του UniTaskDesignSystem module, με μια φόρμα που περιλαμβάνει τα Text Boxes για το Username και Password του χρήστη και το κουμπί για την είσοδο. Τα κουμπιά καλούν προεπιλεγμένες ενέργειες του Mendix.

**PAGE\_Home\_Page**

Η αρχική σελίδα της εφαρμογής που εμφανίζεται μετά την είσοδο του χρήστη.

Η σελίδα χρησιμοποιεί το UniTask\_TopBar layout του UniTaskDesignSystem module, το οποίο εμφανίζει το κεντρικό μενού της εφαρμογής στο πάνω μέρος της σελίδας. Περιλαμβάνει το τίτλο UniTask με ένα call to action κουμπί που καλεί το microflow ShowPage\_TasksOverview.

**PAGE\_Tasks\_Overview**

Η σελίδα χρησιμοποιείται για την εμφάνιση και διαχείριση των εργασιών του χρήστη. Χρησιμοποιεί το UniTask\_TopBar layout του UniTaskDesignSystem module και έχει ως Parameters το Student και το PageHelper.

Η σελίδα αποτελείται από ένα Layout Grid με διαφορετικά rows. Το πρώτο row

χρησιμοποιείται για το καλωσόρισμα του χρήστη. Για την εμφάνιση του ονόματός του έχει χρησιμοποιηθεί ένα Data View με Data source το Student. Πίσω από το text widget που εμφανίζει το μήνυμα καλωσορίσματος υπάρχει στην πραγματικότητα η έκφραση καλώς άριστες, {1}. Τα {X} αποτελούν placeholders για μεταβλητές στη συγκεκριμένη περίπτωση το {1} αντιστοιχεί στο Name του Student.

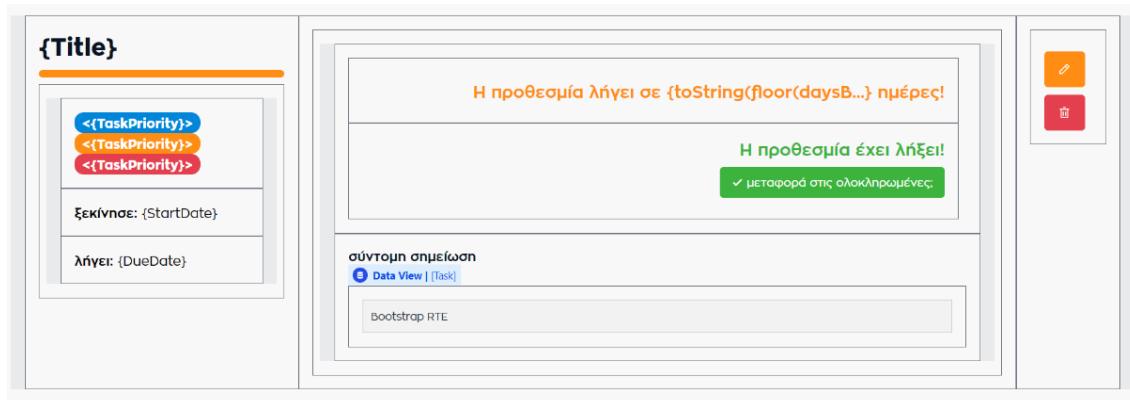
Το δεύτερο row του Layout Grid περιλαμβάνει ένα εσωτερικό Layout Grid που δημιουργεί δύο στήλες (9 και 3)<sup>5</sup>. Στην αριστερή στήλη περιλαμβάνονται οι κάρτες με τις εργασίες. Για να επιτευχθεί αυτό έχουν δημιουργηθεί δύο εμφωλευμένα Data Views με Data sources τα Student και PageHelper. Μέσα στο Data View περιλαμβάνεται ένα Tab Container με τρία tabs που αντιστοιχούν στις τρεις καταστάσεις των εργασιών. Σε κάθε tab αντιστοιχίζεται ένα ξεχωριστό πράσινο κουμπί (Νέα εργασία) που, ανάλογα με το tab που είναι ενεργό, καλεί το microflow CreateNewTask\_ToDo, CreateNewTask\_Doing ή CreateNewTask\_Done. Το περιεχόμενο της καρτέλας είναι ένα List View με τις κάρτες εργασιών. Το περιεχόμενο των καρτών γίνεται populate (έχει δηλαδή Data source) από το microflow DSL\_TaskToDo, DSL\_TaskDoing ή DSL\_TaskDone αντίστοιχα. Για την καλύτερη οργάνωση, η κάρτα ουσιαστικά αποτελεί ένα Snippet που επαναχρησιμοποιείται. Έχουν δημιουργηθεί τα Snippets TaskCard\_Overview\_Doing, TaskCard\_Overview\_ToDo και TaskCard\_Overview\_Done που θα αναλυθούν στη συνέχεια.

Η δεξιά στήλη με τους μετρητές περιλαμβάνει ένα Data View με Data source το PageHelper. Εσωτερικά περιλαμβάνονται containers με text widgets με τις μεταβλητές CounterTaskToDo, CounterTaskDoing και CounterTaskDone.

Όσον αφορά για το σύστημα επιβράβευσης, το κομφετί αποτελεί ένα add-on widget από το Marketplace του Mendix. Το widget αυτό βρίσκεται τοποθετημένο μέσα σε ένα Data View με Data source το PageHelper και εμφανίζεται μόνο όταν η μεταβλητή Confetti γίνεται True.

Στο κάτω μέρος του Tab container υπάρχει ένα επεξηγηματικό μπλε παράθυρο (βλέπε σχήμα 5.11) που εμφανίζεται μόνο όταν δεν έχουν δημιουργηθεί εργασίες. Αυτό επιτυγχάνεται με την έκφραση not (\$PageHelper/AreThereTasks) στην συνθήκη Visibility του container που αντιπροσωπεύει το παράθυρο. Αξίζει επίσης να σημειωθεί πως τα call-to-action κουμπιά για τη νέα εργασία και τη σελίδα Kanban είναι και αυτά clickable και έχουν χρησιμοποιηθεί custom CSS κλάσεις για την εμφάνισή τους.

<sup>5</sup>Χρησιμοποιώντας παρόμοια λογική όπως το Bootstrap, το Mendix χρησιμοποιεί ένα σύστημα 12 στηλών ώστε να καθορίσουμε το πλάτος των στηλών.

**TaskCard\_Overview\_Doing**

Πρόκειται για ένα Snippet που χρησιμοποιείται για την εμφάνιση των καρτών με τις εργασίες που βρίσκονται στην κατάσταση Doing. Έχει ως Parameters τα Task, PageHelper και Student.

Αποτελείται από ένα Layout Grid με τρεις στήλες. Η πρώτη στήλη περιλαμβάνει τον τίτλο, το Progress Bar widget και ένα εσωτερικό Layout Grid με την προτεραιότητα της εργασίας και τις ημερομηνίες έναρξης και λήξης.

Το Progress Bar χρησιμοποιείται για την εμφάνιση της προόδου της εργασίας και ολοκληρώνεται όσο πλησιάζει η ημερομηνία λήξης. Αυτό επιτυγχάνεται με τον καθορισμό τριών τιμών: Current, Minimum και Maximum value. Στις τιμές Minimum και Maximum value καθορίζονται οι εκφράσεις `dateTimeToEpoch($Task/StartTime)` και `dateTimeToEpoch($Task/DueDate)` αντίστοιχα, στις οποίες ουσιαστικά μετατρέπεται η ημερομηνία και ώρα σε ακέραιο αριθμό. Αυτό βοηθάει στο να είναι καθορισμένο ένα άνω και κάτω αριθμητικό όριο στο οποίο θα κινείται το Current value. Το Current value καθορίζεται από την έκφραση<sup>6</sup>:

```
1 if [%CurrentDateTime%] > $Task/DueDate then dateTimeToEpoch($Task/DueDate)
2 else if [%CurrentDateTime%] < $Task/StartTime then dateTimeToEpoch($Task/StartTime)
3 else dateTimeToEpoch[%CurrentDateTime%]
```

η οποία καταφέρνει τη συγκράτηση της τιμής μέσα σε αυτά τα όρια.

Η προτεραιότητα εμφανίζεται ως ένα Badge widget. Στην κάρτα βρίσκονται το ποιο είναι τα τρία, και ανάλογα με το ποιο είναι το TaskPriority εμφανίζεται και εξαφανίζονται τα αντίστοιχα Badges. Τέλος, η ημερομηνία έναρξης και λήξης έχει επιλεχθεί να εμφανίζεται με τη μορφή dd MMM yy, h:mm a, που αντιστοιχεί σε “23 Απρ 18, 1:37 μ.μ.” για παράδειγμα.

<sup>6</sup>Η έκφραση επιστρέφει την τρέχουσα ημερομηνία σε Epoch μορφή, ή –σε περίπτωση που βρισκόμαστε πριν την ημερομηνία έναρξης ή μετά την ημερομηνία λήξης– αυτές τις ημερομηνίες πάλι σε Epoch μορφή. Η αναπαράσταση μιας χρονικής στιγμής σε Epoch βοηθάει στη σύγχριση μεταξύ ακεραίων για το Progress Bar. Να σημειωθεί πως λέγονται Epoch μορφή εννοούμε τα δευτερόλεπτα που έχουν περάσει από τη 1η Ιανουαρίου 1970 μέχρι τη χρονική στιγμή που καθορίζουμε.

Η δεύτερη στήλη περιλαμβάνει ένα Layout Grid με διαφορετικά rows, το πρώτο αφορά δύο containers που εμφανίζονται υπό συνθήκη και περιλαμβάνουν ενημερώσεις για το αν λήγει μια εργασία σε λιγότερο από μια εβδομάδα ή αν έχει ήδη λήξει. Το πρώτο container εμφανίζεται από την έκφραση<sup>7</sup>:

```
1 if daysBetween($Task/DueDate, [%CurrentDateTime%]) < 7 and daysBetween($Task/DueDate,
2 [%CurrentDateTime%]) > 0 and $Task/DueDate > [%CurrentDateTime%]
then true else false
```

και εμφανίζει “Η προθεσμία λήγει σε {1} ημέρες!” όπου το {1} αντιστοιχεί σε:

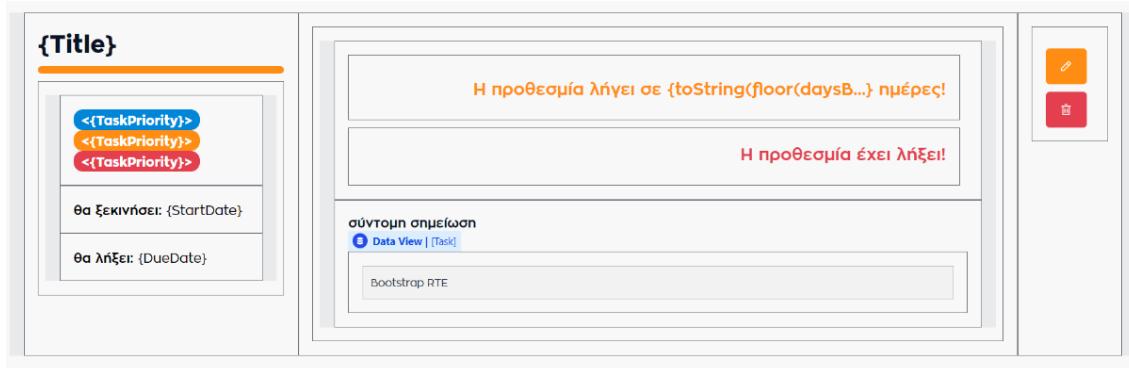
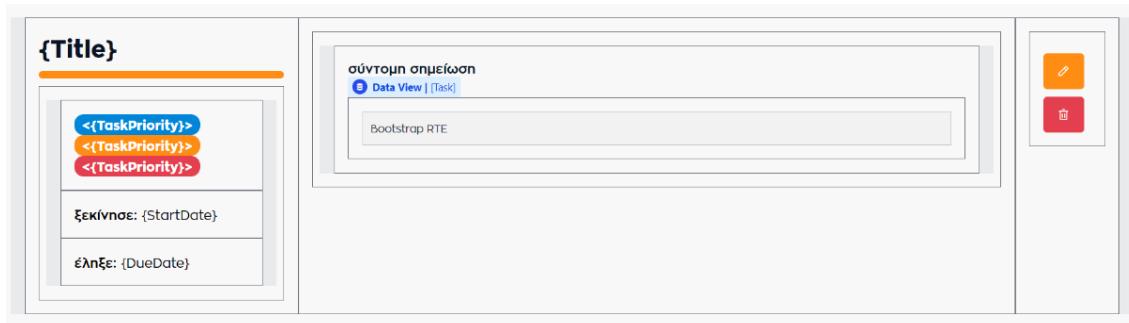
```
1 toString(floor(daysBetween($Task/DueDate, [%CurrentDateTime%])))
```

Το δεύτερο container εμφανίζεται το [%CurrentDateTime] είναι μικρότερο ή ίσο από το DueDate και το κουμπί του καλεί το microflow ChangeTaskStatus\_TaskDone.

Κάτω από το container υπάρχει ένα Data View με Data source το Task και το Bootstrap RTE widget. Το Rich Text Editor παρέχει στους χρήστες τη δυνατότητα δημιουργίας εμπλουτισμένου περιεχομένου, όπως έντονο (bold) και πλάγιο (italic) κείμενο. Το widget αποθηκεύει το κείμενο σε String μορφή.

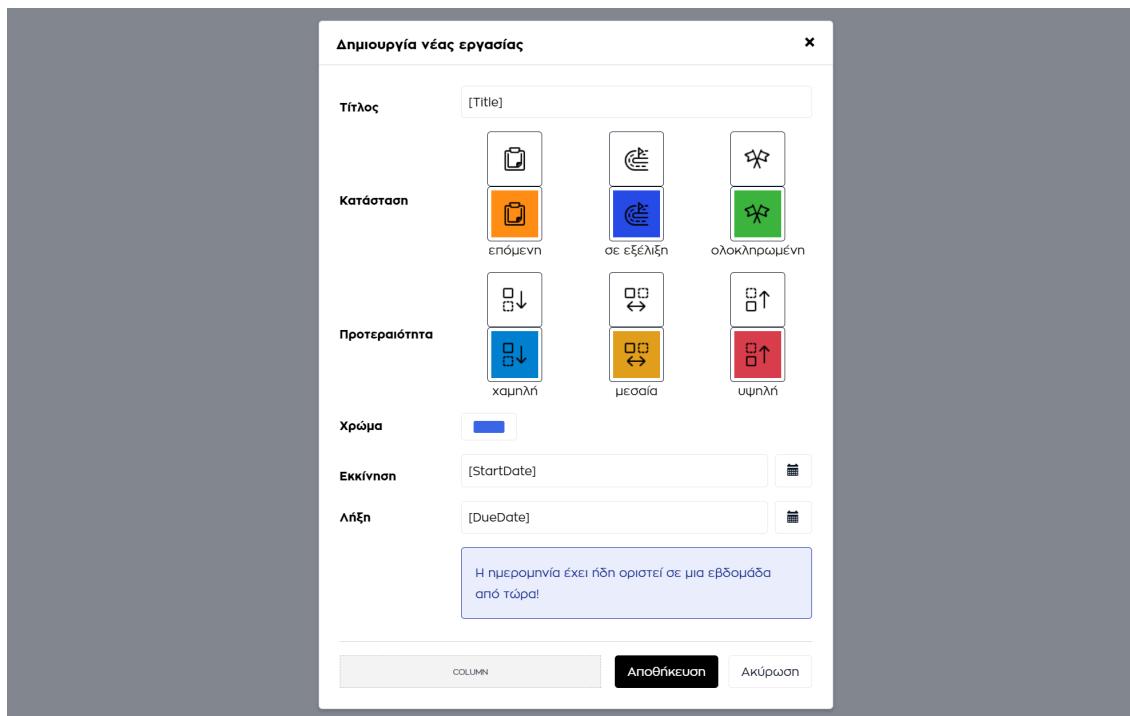
Στη δεξιά στήλη υπάρχουν τα κουμπιά επεξεργασίας και διαγραφής που καλούν τα microflows EditTask και DeleteTask (μετά από επιβεβαίωση) αντίστοιχα. Να σημειωθεί πως το κουμπί διαγραφής εμφανίζεται ανάλογα με την τιμή της Boolean ιδιότητας buttonQuickDelete του Student.

<sup>7</sup>Η έκφραση συγκρίνει τις ημέρες μεταξύ στην τρέχουσα μέρα και της ημερομηνίας λήξης της εργασίας, και επιστέψει True όταν η διαφορά είναι μικρότερη από μια εβδομάδα και η ημερομηνία λήξης ακόμη είναι στο μέλλον.

**TaskCard\_Overview\_ToDo****TaskCard\_Overview\_Done**

Tα Snippets TaskCard\_Overview\_ToDo και TaskCard\_Overview\_Done είναι παρόμοια με το TaskCard\_Overview\_Doing με τις απαραίτητες διαφοροποιήσεις.

Να σημειωθεί πως στο TaskCard\_Overview\_Done το Progress Bar πάντα είναι ολοκληρωμένο, ενώ το TaskCard\_Overview\_ToDo πάντα κενό, και επίσης πως στο δεύτερο δεν υπάρχει call-to-action κουμπί για την αλλαγή κατάστασης της εργασίας μετά την ημερομηνία λήξης, καθώς σε ένα σενάριο που ο χρήστης έχει προσθέσει μια εργασία ως To-Do και έχει τελειώσει η προθεσμία της πριν περάσει στην Doing κατάσταση, η προβλεπόμενη κίνηση θα είναι να τη διαγράψει.

**POPOUT\_Task\_NewEdit**

Η σελίδα καλείται στα microflows CreateNewTask\_ToDo, CreateNewTask\_Doing και CreateNewTask\_Done και χρησιμοποιείται για τη δημιουργία νέων εργασιών. Χρησιμοποιεί το Popout\_Layout layout του Atlas\_Core και έχει ως Parameters το Task και το PageHelper.

Περιλαμβάνει ένα Data View με ένα Layout Grid με τα απαραίτητα Text Boxes και Date Pickers για την εισαγωγή του τίτλου και των ημερομηνιών έναρξης και λήξης. Για την επιλογή του χρώματος χρησιμοποιείται το widget ColorPicker που αποθηκεύει το επιλεγμένο χρώμα στη μεταβλητή CalendarColor.

Για την επιλογή της κατάστασης και της προτεραιότητας έχουν δημιουργηθεί ένα σύνολο από ζεύγη κουμπιών, το ένα άχρωμο και το άλλο έγχρωμο. Ανάλογα με το ποιο είναι το TaskStatus και το TaskPriority εμφανίζεται και εξαφανίζεται το αντίστοιχο σύνολο κουμπιών. Ταυτόχρονα, οι λεζάντες κάτω από τα κουμπιά περιέχουν τη δυναμική κλάση που καθορίζεται από την έκφραση:

```
1 if $Task/TaskPriority = TaskManager.TaskPriority.Low then 'labelSelected'
2 else ''
```

Αντίστοιχα στο αρχείο Styling/web/custom-variables.scss έχει δημιουργηθεί η κλάση:

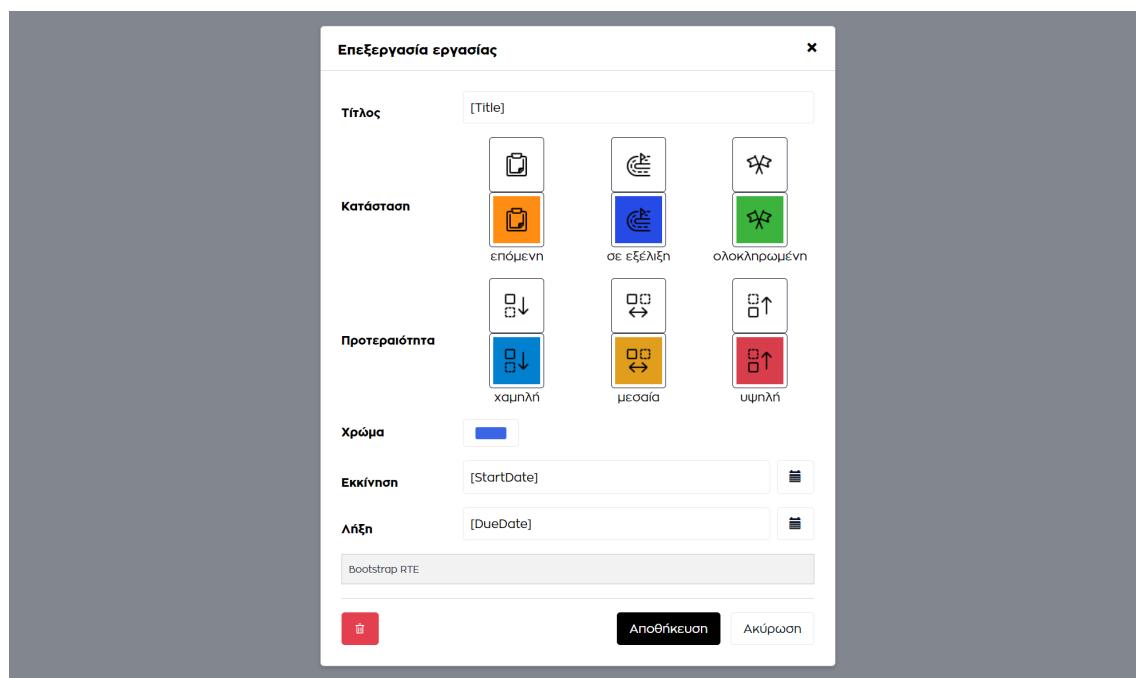
```
1 .labelSelected {
2     font-weight: bold;
3 }
```

Με αυτόν τον τρόπο επιτυγχάνεται η εμφάνιση της επιλεγμένης κατάστασης και προτεραιότητας με έντονη γραφή. Επιπλέον, το κάθε σύνολο από τα ζεύγη κουμπιών βρίσκεται τοποθετημένο σε ένα container, το οποίο αν πατηθεί καλεί τα microflow ChangeTaskStatus\_TaskToDo, ChangeTaskStatus\_TaskDoing, ChangeTaskStatus\_TaskDone και ChangeTaskPriority\_Low, ChangeTaskPriority\_Medium και ChangeTaskPriority\_High αντίστοιχα.

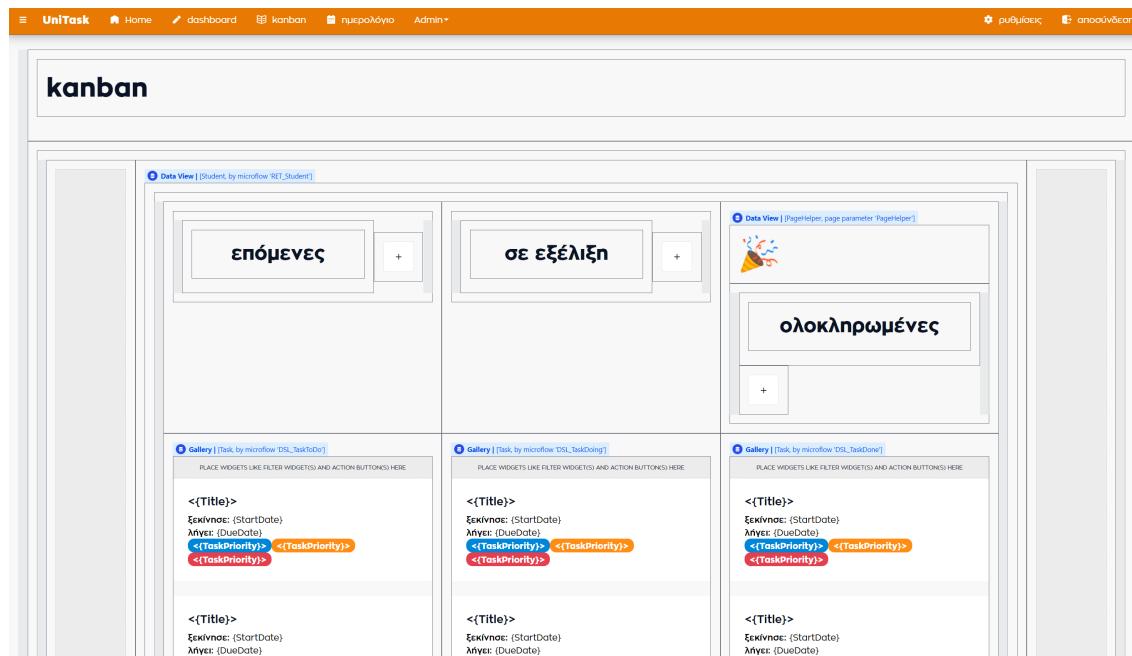
Τέλος, η εμφάνιση του μπλε παραθύρου που ενημερώνει ότι η ημερομηνία λήξης έχει προκαθοριστεί αυτόματα βασίζεται στην Boolean μεταβλητή CreationDateBubble του Task, και επίσης όταν γίνεται κλικ στο Date Picker για την ημερομηνία λήξης, καλείται το microflow DisableCreationDateBubble.

Για να αποθηκευτεί η νέα εργασία καλείται το microflow SaveTask.

#### **POPOUT\_Task\_NewEdit\_Edit**



Η σελίδα καλείται από το microflow Edit\_Task. Χρησιμοποιείται αντίστοιχη λογική με την POPOUT\_Task\_NewEdit με τη διαφορά ότι δεν υπάρχει επεξηγηματικό παράθυρο για την ημερομηνία λήξης, υπάρχει το widget Bootstrap RTE για την επεξεργασία της ιδιότητας Notes και επιπλέον υπάρχει το κουμπί διαγραφής της εργασίας που καλεί το microflow DeleteTask μετά από επιβεβαίωση.

**PAGE\_Tasks\_Kanban**

Η σελίδα χρησιμοποιείται ως ένας εναλλακτικός τρόπος εμφάνισης των εργασιών του χρήστη σε έναν Kanban πίνακα. Χρησιμοποιεί το UniTask\_TopBar layout του UniTaskDesignSystem module και έχει ως Parameter το PageHelper.

Η σελίδα αποτελείται από ένα Layout Grid με 3 στήλες, με τις δύο ακραίες να αποτελούν negative space. Η κεντρική στήλη περιέχει ένα Layout Grid με 3 στήλες. Το πρώτο row του περιλαμβάνει τις επικεφαλίδες του πίνακα μαζί με κουμπιά για την προσθήκη νέας εργασίας. Κάθε κουμπί είναι προσαρμοσμένο να δημιουργεί μια εργασία που αντιστοιχεί στην κατάσταση της επικεφαλίδας, καλώντας τα microflows CreateNewTask\_ToDo, CreateNewTask\_Doing και CreateNewTask\_Done αντίστοιχα. Το δεύτερο row περιλαμβάνει Galleries με Data sources τα DSL\_TaskToDo, DSL\_TaskDoing και DSL\_TaskDone που δημιουργούν τις κάρτες, που αν πατηθούν καλούνται το microflow EditTask. Τέλος, περιλαμβάνεται το Confetti widget, όπως και στην Dashboard σελίδα.

**TaskCard\_Kanban**

```
<{Title}>
Σεκίνησε: {StartDate}
Λήγει: {DueDate} <{TaskPriority}> <{TaskPriority}> <{TaskPriority}>
```

Πρόκειται για ένα Snippet που χρησιμοποιείται για την εμφάνιση των καρτών με τις εργασίες στον πίνακα Kanban. Έχει ως Parameters τα Task και Student.

Αποτελείται από ένα Layout Grid με τον τίτλο, τις ημερομηνίες έναρξης και λή-

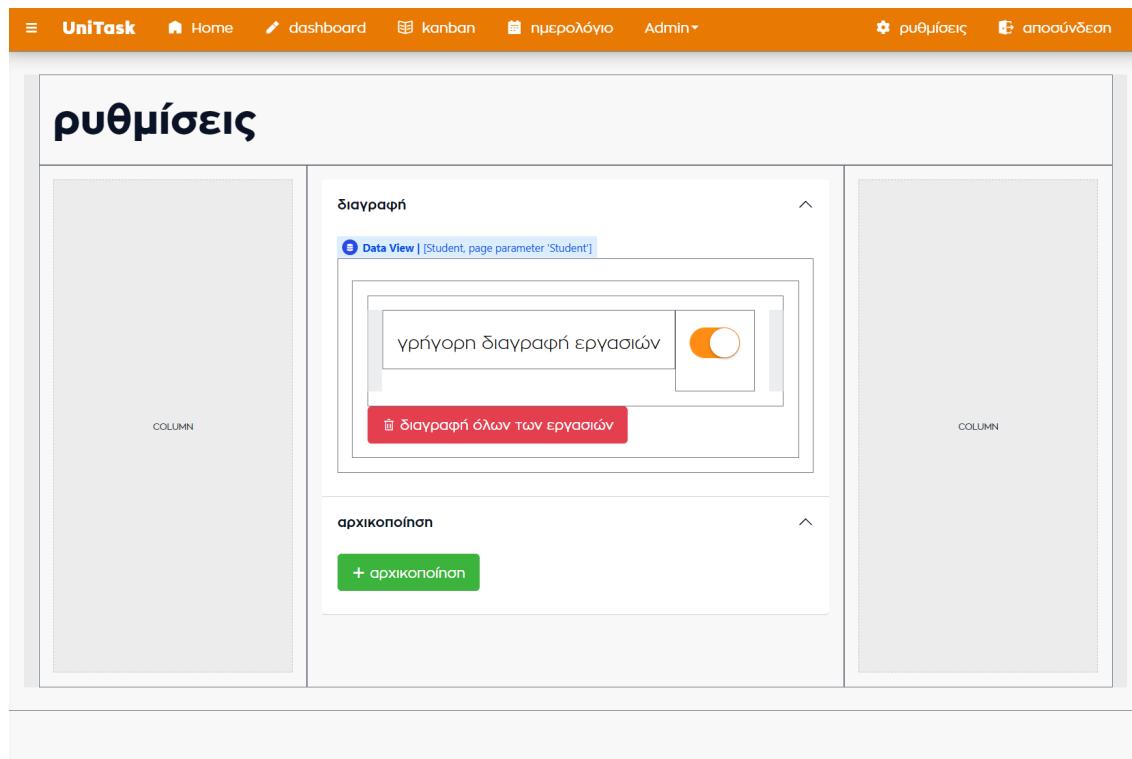
Έντονος και Badges για την προτεραιότητα της εργασίας, με παρόμοιο τρόπο όπως στα Snippets του Dashboard.

### PAGE\_Calendar

The screenshot displays the UniTask application's calendar feature. On the left, a monthly calendar for January 2025 shows various events and tasks. Notable entries include 'Leave' (blue bars) on multiple days, a red bar labeled 'BD' on January 16, and grey bars labeled 'Bank Holiday' on January 19 and 24. On the right, a detailed timeline view for January lists five tasks, each with its title, status, and scheduled start and due dates. The interface is clean with orange header bars and a white background.

Η σελίδα χρησιμοποιείται για την εμφάνιση των εργασιών του χρήστη σε έναν ημερολόγιο. Χρησιμοποιεί το UniTask\_TopBar layout του UniTaskDesignSystem module και έχει ως Parameter το Task και PageHelper.

To Calendar widget είναι τοποθετημένο μέσα σε ένα Data View με Data source το Task. Έτσι καθορίζεται πως το Event entity, δηλαδή ο τύπος αντικειμένου που θα εμφανίζεται στο ημερολόγιο, είναι το Task. Το ημερολόγιο γίνεται populate μέσω του microflow ShowTasks\_Calendar και στα Properties του Widget επιλέγονται οι ιδιότητες Title, StartDate, DueDate και CalendarColor του Task για να καθοριστεί η εμφάνισή του στο ημερολόγιο. Όσον αφορά το Timeline στα δεξιά του, έχει επιλεχθεί να εμφανίζονται οι εργασίες ομαδοποιημένες βάσει της ημερομηνίας έναρξής τους και μάλιστα να μπορεί να γίνει επεξεργασία τους αν γίνει κλικ πάνω τους.

**PAGE\_Settings**

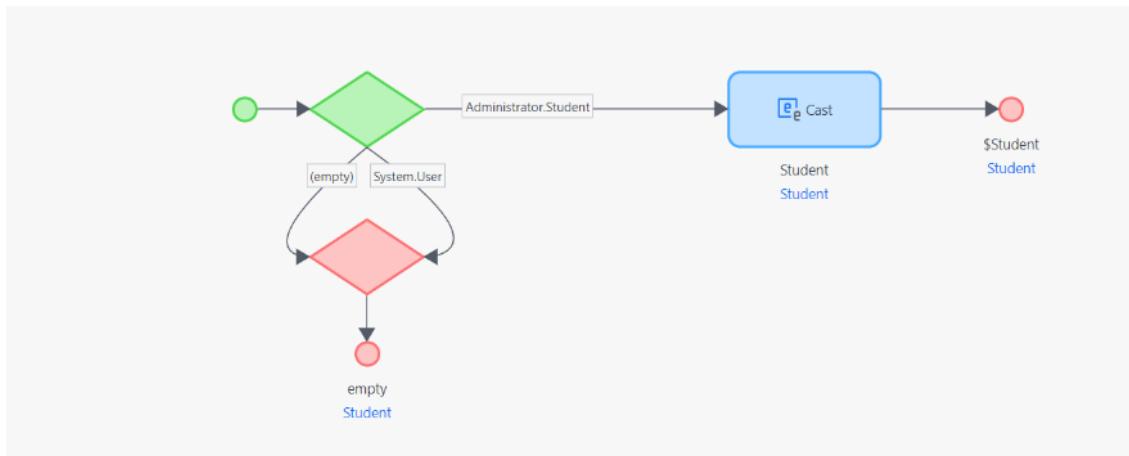
Η σελίδα χρησιμοποιείται για την εμφάνιση των ρυθμίσεων του χρήστη. Χρησιμοποιεί το UniTask\_TopBar layout του UniTaskDesignSystem module και έχει ως Parameters το Student και το PageHelper.

Οι ρυθμίσεις βρίσκονται τοποθετημένες σε ένα Accordion widget με δύο ομάδες: διαγραφή και αρχικοποίηση. Στη διαγραφή περιλαμβάνεται ένα Data View με Data source το Student. Εσωτερικά του υπάρχει ένα Switch widget που κάνει toggle την Boolean ιδιότητα buttonQuickDelete του Student και ένα κουμπί (διαγραφή όλων των εργασιών) που καλεί το microflow DeleteAllTasks μετά από επιβεβαίωση. Στην αρχικοποίηση περιλαμβάνεται το κουμπί (αρχικοποίηση) που καλεί το microflow InitializeTasks μετά από επιβεβαίωση.

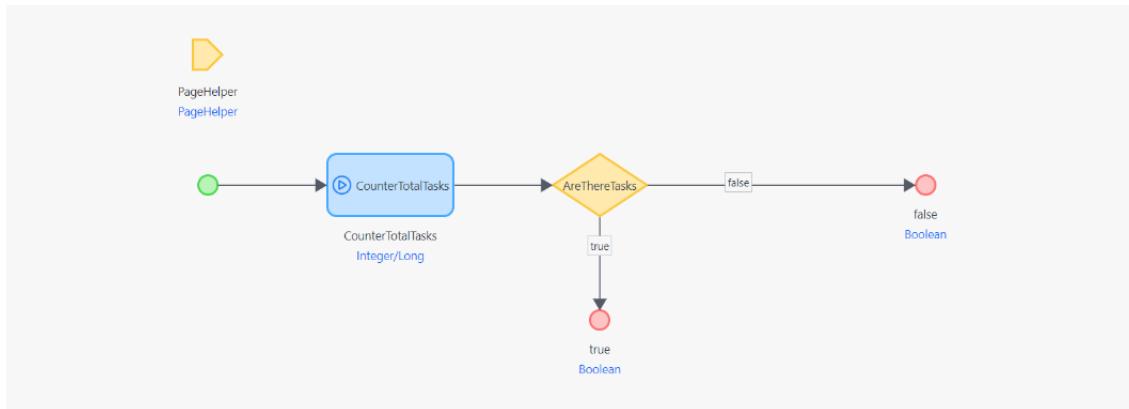
### 5.3.2.3 Microflows του TaskManager

Στο TaskManager περιλαμβάνονται τα εξής microflows<sup>8</sup>:

<sup>8</sup>Στα ονόματα των microflows περιλαμβάνεται ο parent φάκελος όπου βρίσκονται για τον πιο εμφανή διαχωρισμό τους.

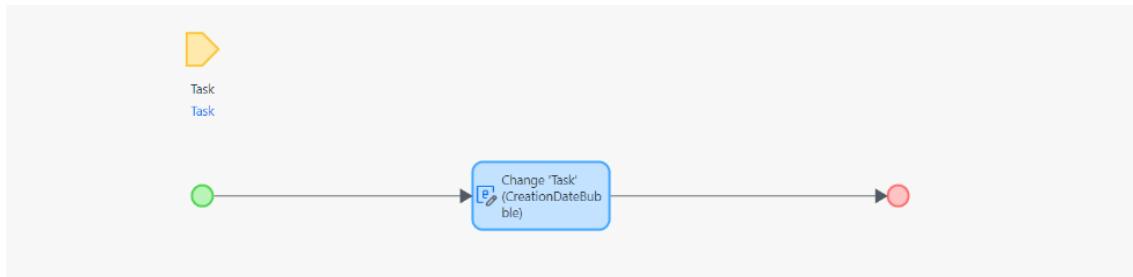
**RET\_Student**

To microflow χρησιμοποιείται για την ανάκτηση του Student με βάση των τρέχοντα System.User. Αν δεν υπάρχει τέτοιος χρήστης, τότε το microflow επιστρέφει ένα κενό αντικείμενο, αλλιώς επιστρέφεται το Student.

**Auxiliary/AreThereTasks**

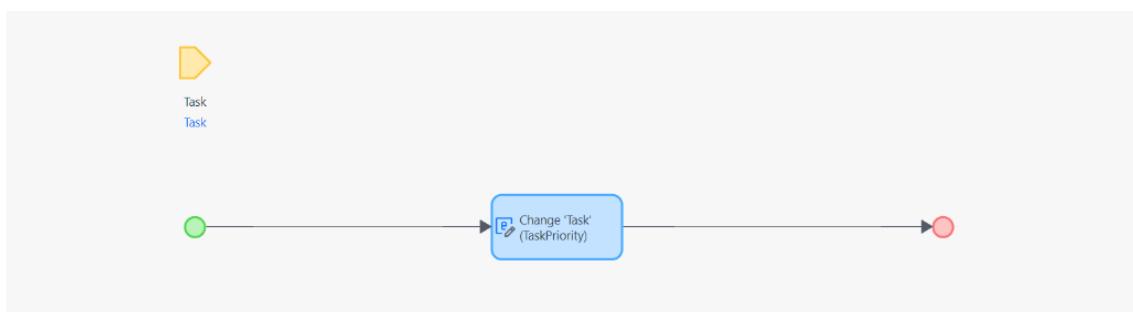
To microflow χρησιμοποιείται από το domain model για τον υπολογισμό της τιμής της Boolean ιδιότητας AreThereTasks του PageHelper.

To microflow καλεί το microflow CounterTotalTasks, το οποίο επιστρέφει ως Integer τον συνολικό αριθμό των εργασιών του χρήστη. Αν ο αριθμός είναι μεγαλύτερος ή ίσος του 1, τότε το microflow επιστρέφει true, αλλιώς false.

**Auxiliary/DisableCreationDateBubble**

To microflow χρησιμοποιείται από τη σελίδα `POPOUT_Task_NewEdit` για να απενεργοποιήσει το μπλε παράθυρο που ενημερώνει ότι η ημερομηνία λήξης έχει προκαθοριστεί αυτόματα. Καλείται όταν ο χρήστης αλλάζει την ημερομηνία λήξης της εργασίας.

To microflow αλλάζει την τιμή της Boolean ιδιότητας `CreationDateBubble` του Task σε `false`.

**ChangeTaskPriorities/ChangeTaskPriority\_TaskLow****ChangeTaskPriorities/ChangeTaskPriority\_TaskMedium****ChangeTaskPriorities/ChangeTaskPriority\_TaskHigh**

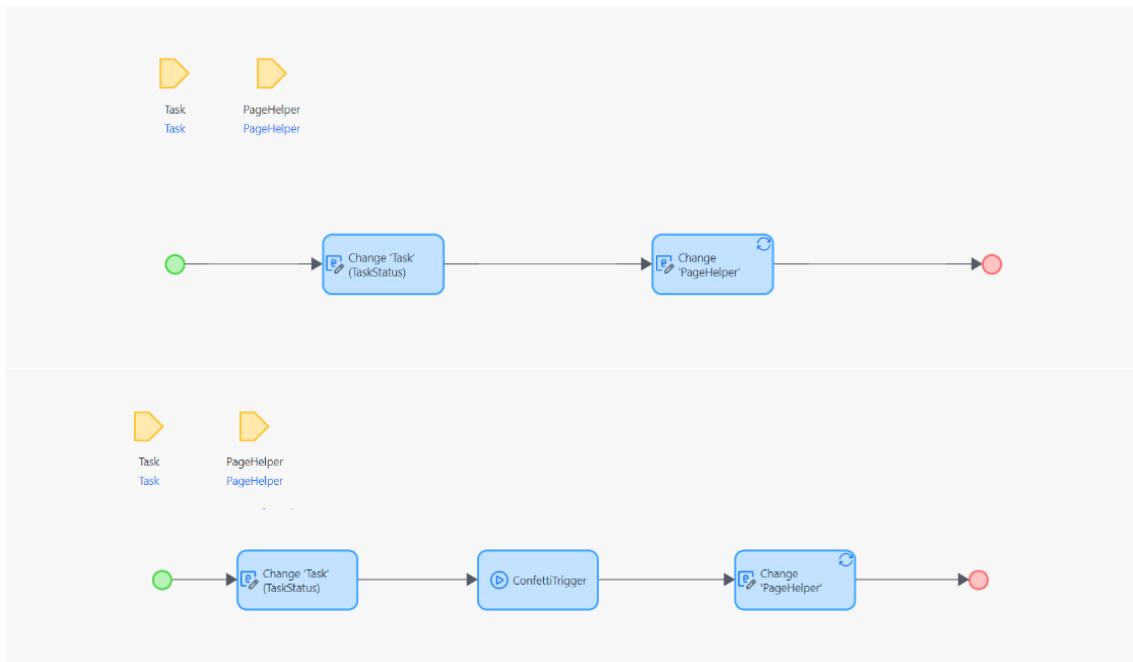
Tα microflows χρησιμοποιούνται για την αλλαγή της προτεραιότητας κάποιας εργασίας. Καλούνται από τα κουμπιά της σελίδας `POPOUT_Task_NewEdit` και `POPOUT_Task_NewEdit_Edit`.

To κάθε microflow αλλάζει την τιμή της Enumeration ιδιότητας `TaskPriority` του Task σε `Low`, `Medium` και `High` αντίστοιχα.

**ChangeTaskStatuses/ChangeTaskStatus\_TaskDoing**

**ChangeTaskStatuses/ChangeTaskStatus\_TaskToDo**

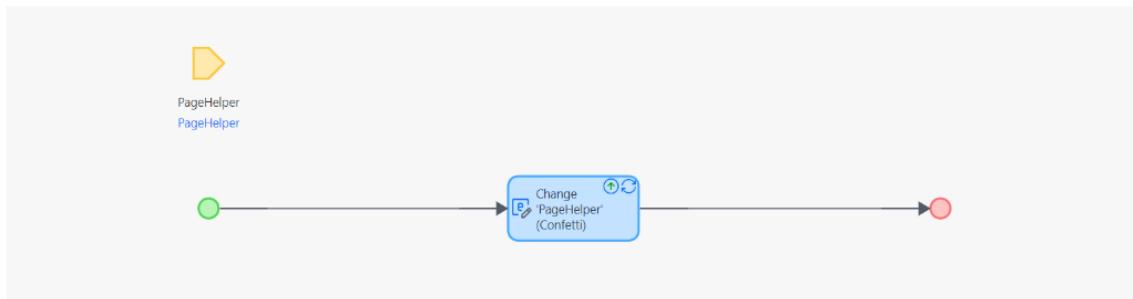
**ChangeTaskStatuses/ChangeTaskStatus\_TaskDone**



Τα microflows χρησιμοποιούνται για την αλλαγή της κατάστασης κάποιας εργασίας. Καλούνται από τα κουμπιά της σελίδας POPOUT\_Task\_NewEdit και POPOUT\_Task\_NewEdit\_Edit.

Το κάθε microflow αλλάζει την τιμή της Enumeration ιδιότητας TaskStatus του Task σε TaskStatus.Doing, TaskStatus.ToDo και TaskStatus.Done αντίστοιχα. Επιπλέον, στην περίπτωση του microflow ChangeTaskStatus\_TaskDone καλεί το microflow ConfettiTrigger για το εφέ του κομφετί.

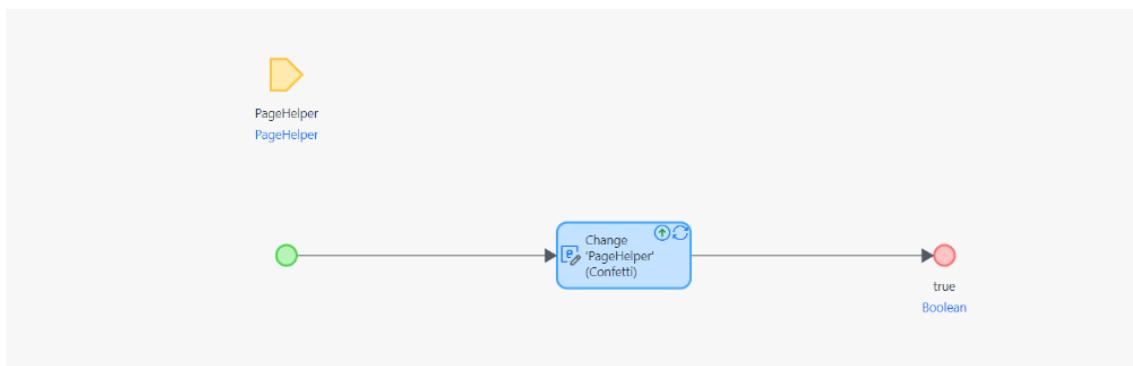
**Confetti/Confetti**



Το microflow καλείται από το microflow ChangeTaskStatus\_TaskDone και χρησιμοποιείται για την εμφάνιση του εφέ κομφετί στην οθόνη του χρήστη. Αλλάζει την

τιμή της Boolean ιδιότητας Confetti του PageHelper σε true.

#### Confetti/Confetti

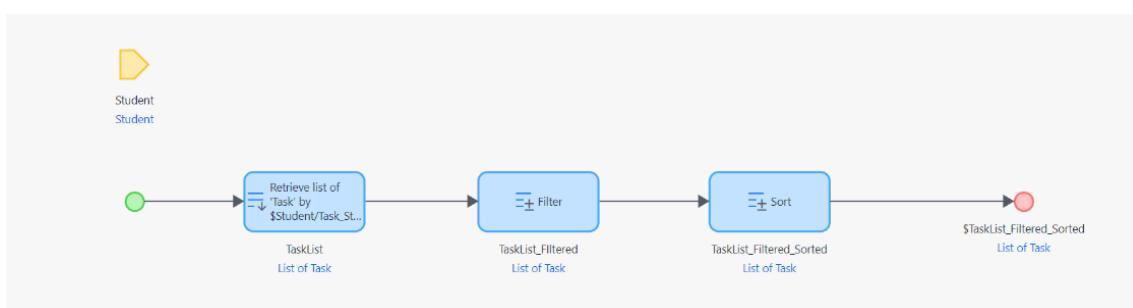


Το microflow καλείται από το microflow SaveTask και χρησιμοποιείται για να επαναφέρει την τιμή της Boolean ιδιότητας Confetti του PageHelper σε false.

#### ShowTasks/DSL\_TaskDoing

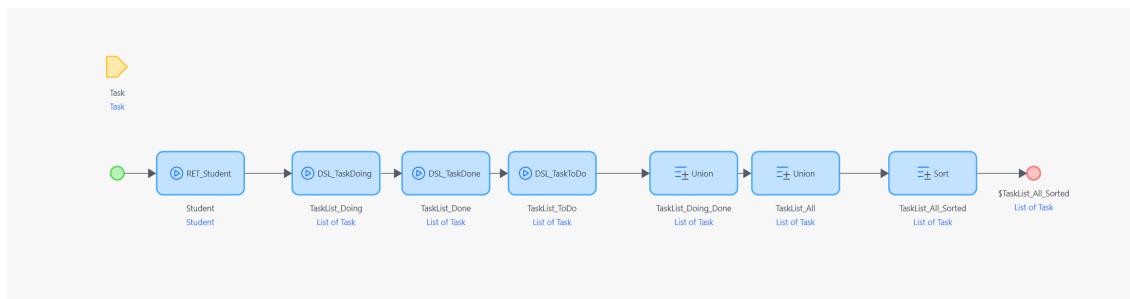
#### ShowTasks/DSL\_TaskDone

#### ShowTasks/DSL\_TaskToDo



Τα microflows φιλτράρουν και επιστρέφουν τις εργασίες του χρήστη ανάλογα με την κατάστασή τους. Χρησιμοποιούνται από τις σελίδες PAGE\_Tasks\_Overview και PAGE\_Tasks\_Kanban και τα microflows ShowTasks\_Calendar και CounterTaskDoing.

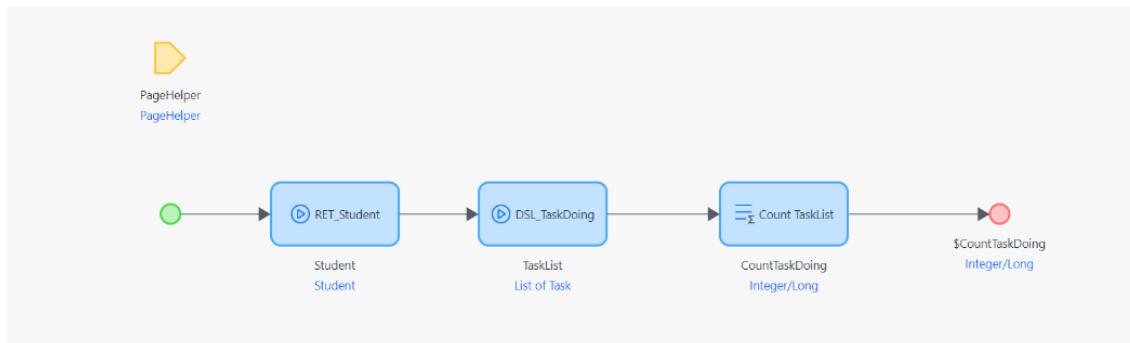
Με Parameter το Student, το microflow κάνει retrieve τη λίστα των Tasks (αφού συσχετίζονται). Η λίστα φιλτράρεται βάσει του TaskStatus, ταξινομείται βάσει δύο ιδιοτήτων σε αύξουσα σειρά, τα TaskPriority και DueDate, και επιστρέφεται.

**ShowTasks/ShowTasks\_Calendar**

To microflow επιστρέφει το σύνολο των εργασιών του χρήστη για την εμφάνισή τους στο ημερολόγιο.

To RET\_Student επιστρέφει το Student, ώστε να χρησιμοποιηθεί ως παράμετρος στα microflows DSL\_TaskDoing, DSL\_TaskDone και DSL\_TaskToDo, τα οποία καλούνται και οι λίστες τους επιστρέφονται. Στη συνέχεια με List Operations οι λίστες συνενώνονται, ταξινομούνται βάσει της ημερομηνίας έναρξης και επιστρέφονται.

Ο λόγος που δε χρησιμοποιείται κάποια παράμετρος Student απευθείας και καλείται η RET\_Student είναι γιατί από τη φύση του Calendar widget είναι απαραίτητο να βρίσκεται σε ένα Data View με Data source το Task, άρα κατά συνέπεια θα έχει παράμετρο το Task.

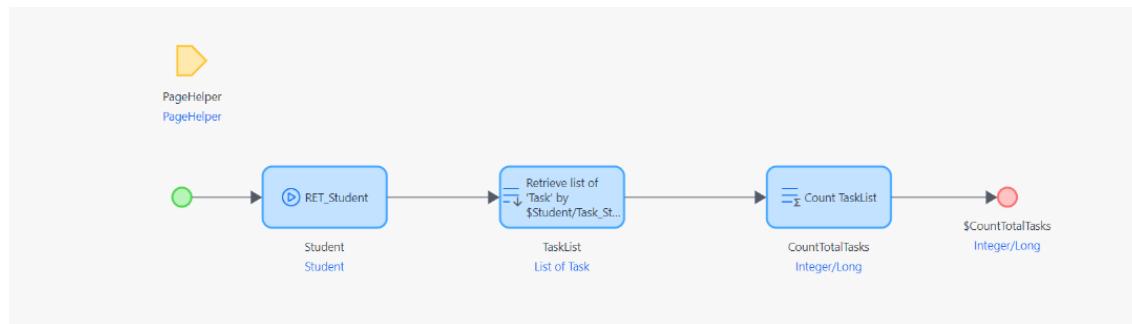
**CounterGenerators/CounterTaskDoing****CounterGenerators/CounterTaskDone****CounterGenerators/CounterTaskToDo**

Tα microflows χρησιμοποιούνται για τον υπολογισμό του συνολικού αριθμού των εργασιών του χρήστη ανάλογα με την κατάστασή τους, και καλούνται από το domain model για τον υπολογισμό των τιμών των ιδιοτήτων TaskToDoCount, TaskDoingCount και TaskDoneCount του PageHelper.

To κάθε microflow καλεί το RET\_Student και τα microflows DSL\_TaskToDo, DSL\_TaskDoing και DSL\_TaskDone αντίστοιχα. Έπειτα μέσω του Function Count του

Aggregate List Action του Mendix, υπολογίζει τον αριθμό των εργασιών και τον επιστρέφει.

#### CounterGenerators/CounterTotalTasks

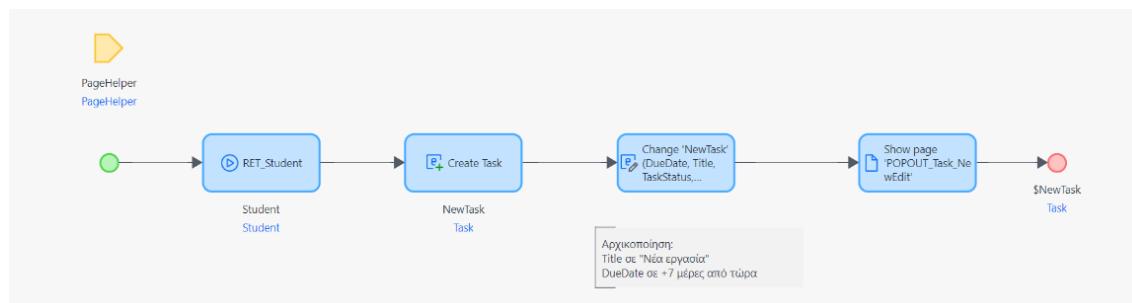


Το microflow καλείται από το microflow AreThereTasks και χρησιμοποιείται για τον υπολογισμό του συνολικού αριθμού των εργασιών του χρήστη. Χρησιμοποιείται παρόμοια λογική με τα προηγούμενα microflows: γίνεται retrieve το σύνολο των εργασιών για έναν συγκεκριμένο χρήστη και καταμετρούνται.

#### CreateNewTask/CreateNewTask\_Done

#### CreateNewTask/CreateNewTask\_Done

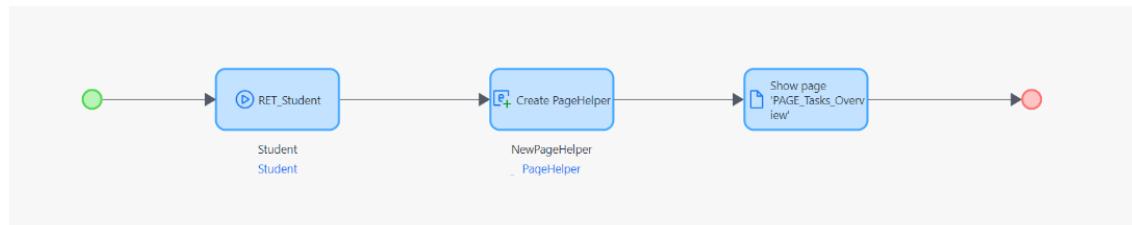
#### CreateNewTask/CreateNewTask\_ToDo



Το microflow καλείται από τα κουμπιά νέα εργασία των σελίδων PAGE\_Tasks\_Kanban και PAGE\_Tasks\_Overview με σκοπό τη δημιουργία μιας νέας εργασίας.

Αρχικά καλείται το microflow RET\_Student για την ανάκτηση του Student. Στη συνέχεια δημιουργείται ένα νέο αντικείμενο τύπου Task όπου ορίζονται κάποιες αρχικές τιμές. Συγκεκριμένα ορίζεται η ημερομηνία λήξης DueDate ως addDays([%CurrentDateTime%], 7) (μια εβδομάδα μετά την τρέχουσα ημερομηνία), ο τίτλος Title ως 'Νέα εργασία', και το TaskStatus ανάλογα με το τύπο του microflow. Επειτα, εμφανίζεται η αναδυόμενη σελίδα POPOUT\_Task\_NewEdit για την επεξεργασία των ιδιοτήτων της εργασίας και την τελική αποθήκευσή της.

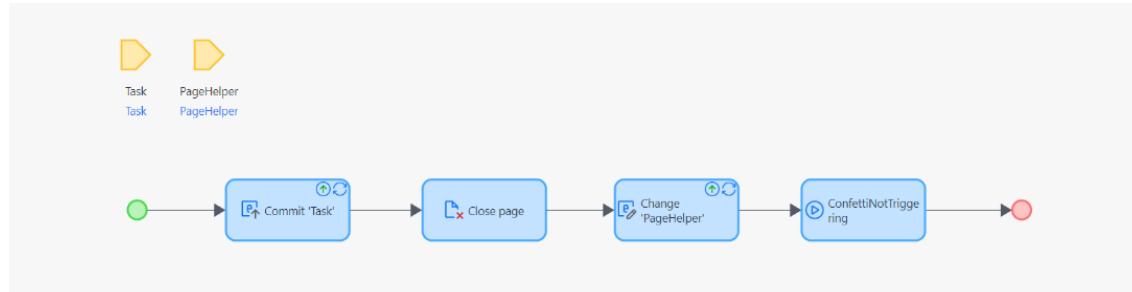
**ShowPages/ShowPage\_Calendar**  
**ShowPages/ShowPage\_Homepage**  
**ShowPages/ShowPage\_Kanban**  
**ShowPages/ShowPage\_Settings**  
**ShowPages/ShowPage\_TasksOverview**



Τα microflows χρησιμοποιούνται για την ανακατεύθυνση του χρήστη σε μια συγκεκριμένη σελίδα. Καλούνται όποτε χρειάζεται η εμφάνιση μιας σελίδας, κυρίως από το Navigation μενού.

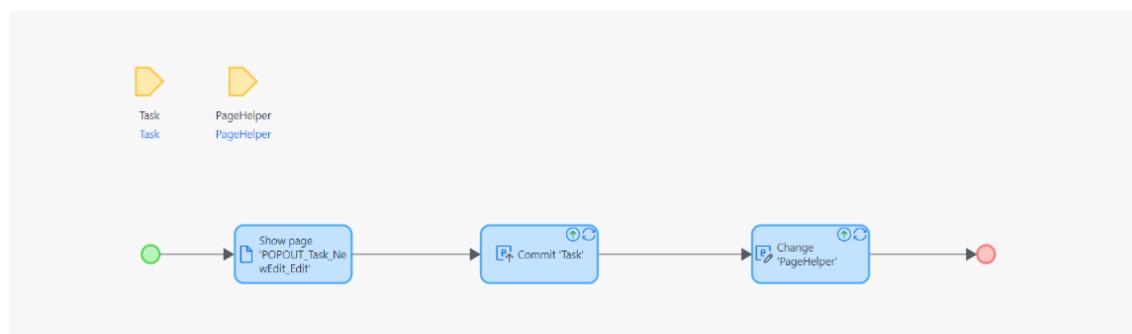
Αφού ανακτήθει το Student μέσω του microflow RET\_Student, το εκάστοτε microflow δημιουργεί ένα PageHelper αντικείμενο και εμφανίζει την αντίστοιχη σελίδα. Η δημιουργία του PageHelper είναι απαραίτητη καθώς χρησιμοποιείται ως Parameter στις σελίδες.

#### SaveTask



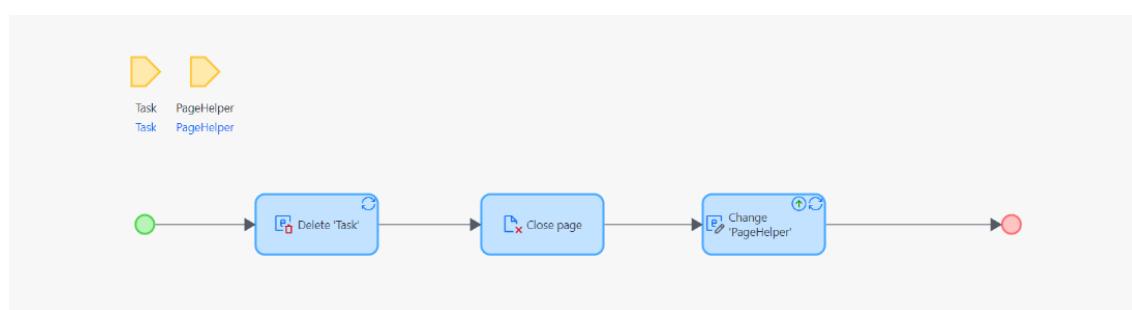
Το microflow καλείται από τις αναδυόμενες σελίδες POPOUT\_Task\_NewEdit και POPOUT\_Task\_NewEdit\_Edit για την αποθήκευση των αλλαγών που έγιναν στις ιδιότητες της εργασίας.

Γίνεται commit στο αντικείμενο Task, κλείνει η σελίδα, γίνεται ένα Change Object στο PageHelper για να ανανεωθούν οι τιμές του, όπως υπολογίζονται από τα microflows και καλείται το microflow ConfettiNotTriggering για την επαναφορά της τιμής της Boolean ιδιότητας Confetti του PageHelper σε false.

**EditTask**

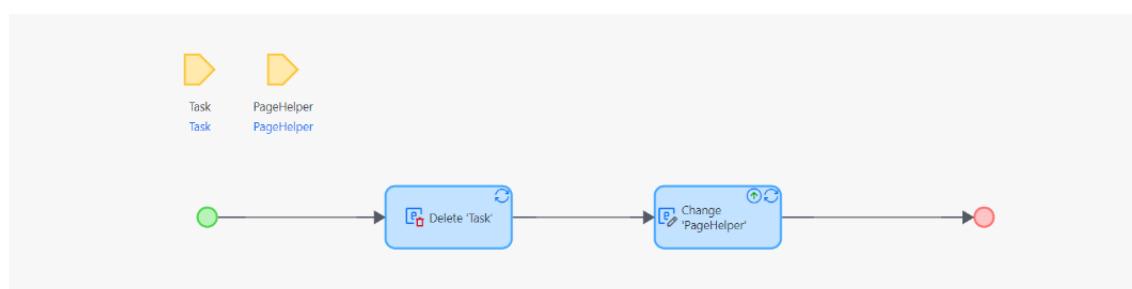
Το microflow καλείται από τα κουμπιά των καρτών στην Dashboard σελίδα και τις κάρτες στην πίνακα Kanban και του ημερολογίου για την επεξεργασία μιας εργασίας.

Το microflow εμφανίζει την αναδυόμενη σελίδα POPOUT\_Task\_NewEdit\_Edit, κάνει commit στο αντικείμενο Task και ανανεώνει το PageHelper.

**DeleteTask**

Το microflow καλείται από το κουμπί της αναδυόμενης σελίδας POPOUT\_Task\_NewEdit\_Edit για τη διαγραφή μιας εργασίας.

Γίνεται delete το αντικείμενο Task, κλείνει η σελίδα και ανανεώνεται το PageHelper.

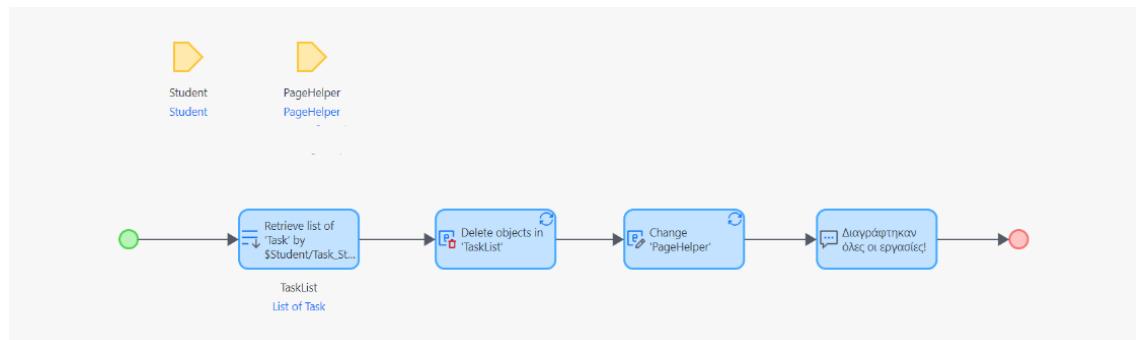
**DeleteTask\_Snippet**

Το microflow καλείται από το κουμπί της κάρτας στην Dashboard σελίδα για

τη διαγραφή μιας εργασίας, στην περίπτωση που ο χρήστης έχει ενεργοποιήσει την επιλογή γρήγορης διαγραφής.

Γίνεται delete το αντικείμενο Task και ανανεώνεται το PageHelper.

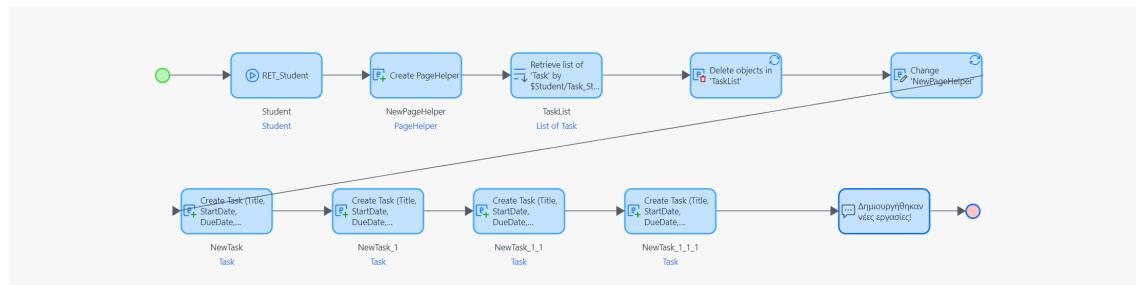
#### DeleteAllTasks



Το microflow καλείται από το κουμπί της σελίδας `POPOUT_Settings` για τη διαγραφή όλων των εργασιών του χρήστη.

Αρχικά γίνεται retrieve το σύνολο των εργασιών του χρήστη ως μια λίστα και στη συνέχεια γίνεται διαγραφή της λίστας. Τέλος, ανανεώνεται το PageHelper και στέλνεται επιβεβαιωτικό μήνυμα.

#### InitializeTasks



Το microflow καλείται από το κουμπί της σελίδας `POPOUT_Settings` για την αρχικοποίηση των εργασιών του χρήστη.

Αρχικά επιστρέφεται το Student μέσω του `RET_Student` και δημιουργείται ένα `PageHelper` που θα χρησιμοποιηθεί ως όρισμα. Πριν δημιουργηθούν οι νέες εργασίες, γίνονται retrieve οι υπάρχουσες, διαγράφονται και ανανεώνεται το `PageHelper`. Στη συνέχεια δημιουργούνται τέσσερις εργασίες με hard-coded προκαθορισμένες τιμές και στέλνεται επιβεβαιωτικό μήνυμα.

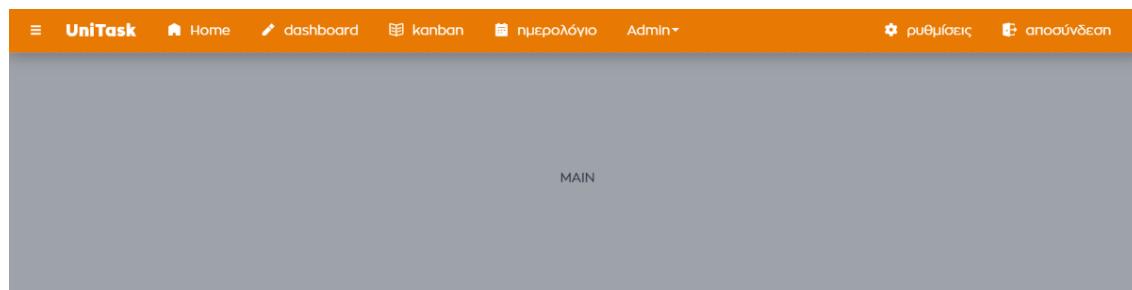
### 5.3.3 Module UniTaskDesignSystem

Το UniTaskDesignSystem περιλαμβάνει τα layouts της εφαρμογής όπως επίσης παρεμβάσεις που αφορούν το styling της εφαρμογής.

#### 5.3.3.1 Layouts του UniTaskDesignSystem

To module UniTaskDesignSystem περιλαμβάνει τα εξής layouts:

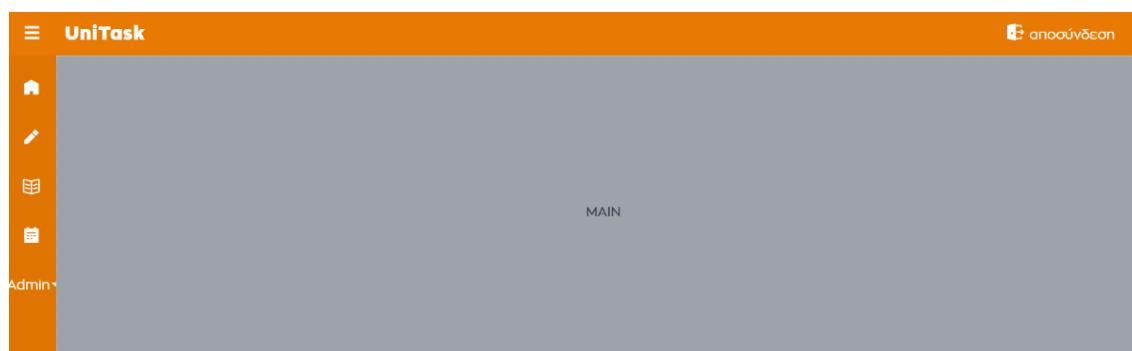
##### UniTask\_TopBar



Το layout χρησιμοποιείται για την εμφάνιση της μπάρας πλοήγησης στην κορυφή της σελίδας. Η μπάρα πλοήγησης αποτελείται από δύο διαφορετικά Navigation μενού, το αριστερό (το κύριο Project navigation) και το δεξί (ένα Menu document του UniTaskDesignSystem) τοποθετημένα σε ένα Layout Grid με δύο στήλες. Τα Navigation μενού καλού τα αντίστοιχα ShowPage microflows για την ανακατεύθυνση του χρήστη στην επιλεγμένη σελίδα. Εξαίρεση αποτελεί το κουμπί Home που καλεί τη σελίδα PAGE\_Home\_Page απευθείας και το κουμπί αποσύνδεση που αποσυνδέει τον χρήστη.

Ο τίτλος της εφαρμογής **UniTask** που εμφανίζεται στα αριστερά χρησιμοποιεί custom CSS και περιλαμβάνεται ένα hamburger μενού στα αριστερά για μεγαλύτερη προσβασιμότητα. Το Admin περιλαμβάνει το υπομενού Account Overview και είναι προσβάσιμα μόνο από τον Administrator.

##### UniTask\_SideBar



Ισχύουν τα αντίστοιχα με το προηγούμενο layout με τη διαφορά ότι δεν εμφανίζονται οι Ρυθμίσεις και το κύριο Project navigation εμφανίζεται στα αριστερά.

### 5.3.3.2 Styling του UniTaskDesignSystem

Στα αρχεία `custom-variables.scss` και `design.scss` των UniTaskDesignSystem και App modules περιλαμβάνεται custom CSS κώδικας για το styling της εφαρμογής. Εκεί ορίζεται για παράδειγμα το πορτοκαλί χρώμα που έχουν οι μπάρες πλοήγησης, η custom γραμματοσειρά Zona Pro που χρησιμοποιείται ή κάποιες custom δυναμικές κλάσεις.

## **Κεφάλαιο 6**

**<Συμπεράσματα - Προεκτάσεις>**

# 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](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](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 Mobile App Development Platforms — gartner.com.* <https://www.gartner.com/en/documents/3882864>. [Accessed 31-12-2024].

- [13] *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].
- [14] Dmitry Golovin. *OutSystems as a Rapid Application Development Platform for Mobile and Web Applications*. 2017.
- [15] Jack Goody. «Memory in Oral Tradition». Στο: Cambridge University Press, Μαρ. 2013, σσ. 73–94. doi: 10.1017/cbo9781139171137.005.
- [16] *Guide to the Project Management Body of Knowledge*. Project Management Institute, 2021. ISBN: 1628256648.
- [17] *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].
- [18] *Hoover Dam | Description, Location, Construction, Facts, History, & Pictures | Britannica* — britannica.com. <https://www.britannica.com/topic/Hoover-Dam>. [Accessed 25-12-2024].
- [19] 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.
- [20] D. L. Kuhn. *Selecting and Effectively Using a Computer-Aided Software Engineering Tool*. 1989.
- [21] Tim Leung. «Introducing Power Apps». Στο: *Beginning Power Apps: The Non-Developer's Guide to Building Business Applications*. Berkeley, CA: Apress, 2021, σσ. 3–19. ISBN: 978-1-4842-6683-0. doi: 10.1007/978-1-4842-6683-0\_1. URL: [https://doi.org/10.1007/978-1-4842-6683-0\\_1](https://doi.org/10.1007/978-1-4842-6683-0_1).
- [22] Benne Lientz και Kathryn Rea. *Project Management for the 21st Century*. 2007.
- [23] *Manage Your Team's Projects From Anywhere | Trello* — trello.com. <https://trello.com/>. [Accessed 16-10-2024].
- [24] *MDA FAQ | Object Management Group* — omg.org. [https://www.omg.org/mda/faq\\_mda.htm](https://www.omg.org/mda/faq_mda.htm). [Accessed 08-11-2024].
- [25] *Mendix Cloud* — docs.mendix.com. <https://docs.mendix.com/developerportal/deploy/mendix-cloud-deploy/>. [Accessed 04-01-2025].
- [26] *Mendix Documentation* — docs.mendix.com. <https://docs.mendix.com/>. [Accessed 04-01-2025].
- [27] *Mendix Forum - System Module* — community.mendix.com. <https://community.mendix.com/link/space/studio-pro/questions/88842>. [Accessed 05-01-2025].

- [28] *OutSystems Platform Architecture | Evaluation Guide | OutSystems* — [outsystems.com](https://www.outsystems.com/evaluation-guide/architecture/). URL: <https://www.outsystems.com/evaluation-guide/architecture/>.
- [29] G. Premkumar και Michael Potter. *Adoption of Computer Aided Software Engineering (CASE) Technology: An Innovation Adoption Perspective*.
- [30] QuickBase. *The State Of Citizen Development Report – September 2015*. [https://cdn2.hubspot.net/hubfs/172645/QuickBase\\_Citizen\\_Developer\\_Report.pdf](https://cdn2.hubspot.net/hubfs/172645/QuickBase_Citizen_Developer_Report.pdf). [Accessed 25-11-2024].
- [31] Quickbase. *Gartner® Report: Future of Work Trends* — [quickbase.com](https://quickbase.com/gartner-future-of-work). <https://www.quickbase.com/gartner-future-of-work>. [Accessed 25-11-2024].
- [32] Quipu - Wikipedia — [en.wikipedia.org](https://en.wikipedia.org/wiki/Quipu). <https://en.wikipedia.org/wiki/Quipu>. [Accessed 22-10-2024].
- [33] Rapid application development - Wikipedia — [en.wikipedia.org](https://en.wikipedia.org). [https://en.wikipedia.org/wiki/Rapid\\_application\\_development](https://en.wikipedia.org/wiki/Rapid_application_development). [Accessed 29-12-2024].
- [34] Thomas Q Reefe. *Lukasa: A Luba Memory Device*. doi: [doi:10.2307/3335144](https://doi.org/10.2307/3335144).
- [35] E. G. Richards. *Mapping time: The calendar and its history*. Oxford University Press, 2000.
- [36] 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). <https://www.cnbc.com/2020/04/01/new-microsoft-google-amazon-cloud-battle-over-world-without-code.html>. [Accessed 25-11-2024].
- [37] 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](https://doi.org/10.1007/s10270-021-00970-2).
- [38] 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](https://doi.org/10.1109/SEAA51224.2020.00036).
- [39] Matthias Book Sami Beydeda και Volker Gruhn. *Model-Driven Software Development*.
- [40] Phil Simon. *Low-Code/No-Code: Citizen Developers and the Surprising Future of Business Applications*. 2022.
- [41] Andrew Stellman. *Learning agile: Understanding scrum, XP, lean, and Kanban*. Findaway World, 2023.
- [42] O'Reilly Editorial Team. «Low-Code and the Democratization of Programming». Στο: *O'Reilly Media* (2021).
- [43] Todoist | A To-Do List to Organize Your Work & Life — [todoist.com](https://todoist.com). <https://todoist.com/>. [Accessed 16-10-2024].

- [44] 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](https://lumenmarketing.com/wp-content/uploads/2017/11/TV_Citizen_Dev.pdf). [Accessed 25-11-2024]. 2014.
- [45] Julia Castillo Trujillo. *Designing A Time Management App For And With Informatics Students.* 2020.
- [46] *What Is Low-Code? | IBM — ibm.com.* <https://www.ibm.com/topics/low-code>. [Accessed 11-10-2024].
- [47] *WinWorld: Welcome — winworldpc.com.* <https://winworldpc.com/home>. [Accessed 31-10-2024].
- [48] *Your connected workspace for wiki, docs & projects | Notion — notion.so.* <https://www.notion.so/>. [Accessed 16-10-2024].
- [49] Μιχαήλ Ξένος. *Ποιότητα Λογισμικού.* GOTSIS, 2021. ISBN: 9786185560102.