

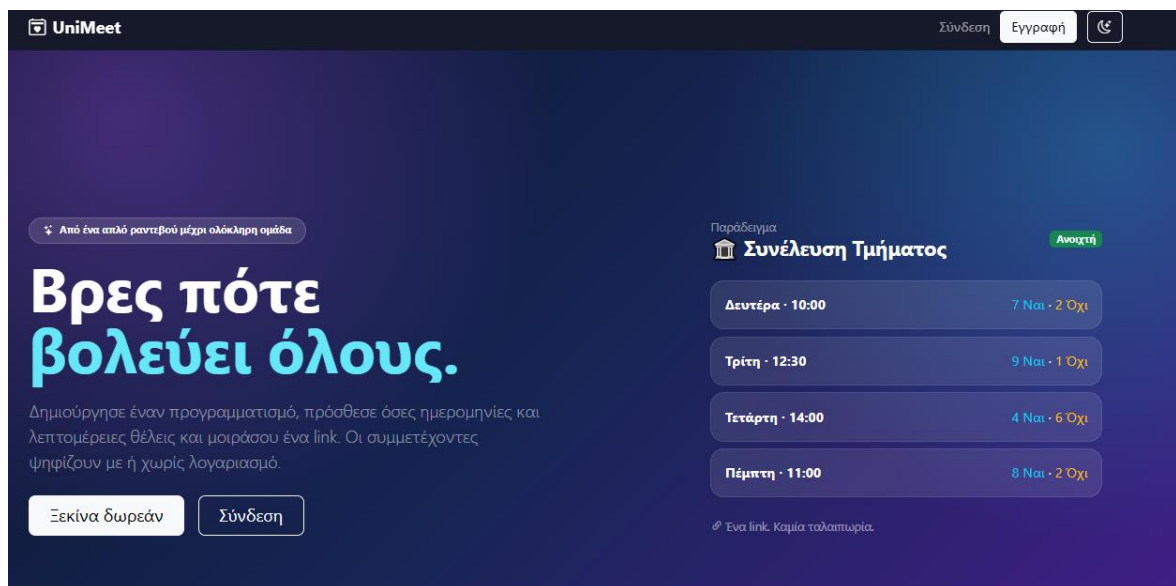


ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

«Σχεδίαση και ανάπτυξη συστήματος κράτησης και
προγραμματισμού διαδικτυακών συναντήσεων»



Φοιτητής
Καραγιάννης Λεωνίδας
175067

Επιβλέπων
Δρ. Κυριάκος Τσιακμάκης

Σεπτέμβριος 2026

Σχεδίαση και ανάπτυξη συστήματος κράτησης και προγραμματισμού διαδικτυακών συναντήσεων

Κωδικός: 26241

Φοιτητής: Καραγιάννης Λεωνίδα

Εισηγητής: Κυριάκος Τσιακμάκης

Ημερομηνία ανάληψης Π.Ε. 25-03-2026

Ημερομηνία περάτωσης Π.Ε. 05-09-2026

Βεβαιώνω ότι είμαι ο συγγραφέας αυτής της εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, έχω καταγράψει τις όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών, εικόνων και κειμένου, είτε αυτές αναφέρονται ακριβώς είτε παραφρασμένες. Επιπλέον, βεβαιώνω ότι αυτή η εργασία προετοιμάστηκε από εμένα προσωπικά, ειδικά ως πτυχιακή εργασία, στο Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του ΔΙ.ΠΑ.Ε.

*Η παρούσα εργασία αποτελεί πνευματική ιδιοκτησία του φοιτητή **Καραγιάννη Λεωνίδα** που την εκπόνησε. Στο πλαίσιο της πολιτικής ανοικτής πρόσβασης, ο συγγραφέας/δημιουργός εκχωρεί στο Διεθνές Πανεπιστήμιο της Ελλάδος άδεια χρήσης του δικαιώματος αναπαραγωγής, δανεισμού, παρουσίασης στο κοινό και ψηφιακής διάχυσης της εργασίας διεθνώς, σε ηλεκτρονική μορφή και σε οποιοδήποτε μέσο, για διδακτικούς και ερευνητικούς σκοπούς, άνευ ανταλλάγματος. Η ανοικτή πρόσβαση στο πλήρες κείμενο της εργασίας, δεν σημαίνει καθ' οιονδήποτε τρόπο παραχώρηση δικαιωμάτων διανοητικής ιδιοκτησίας του συγγραφέα/δημιουργού, ούτε επιτρέπει την αναπαραγωγή, αναδημοσίευση, αντιγραφή, πώληση, εμπορική χρήση, διανομή, έκδοση, μεταφόρτωση (downloading), ανάρτηση (uploading), μετάφραση, τροποποίηση με οποιονδήποτε τρόπο, τμηματικά ή περιληπτικά της εργασίας, χωρίς τη ρητή προηγούμενη έγγραφη συναίνεση του συγγραφέα/δημιουργού.*

Η έγκριση της πτυχιακής εργασίας από το Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του Διεθνούς Πανεπιστημίου της Ελλάδος, δεν υποδηλώνει απαραίτητως και αποδοχή των απόψεων του συγγραφέα, εκ μέρους του Τμήματος.

Περίληψη

Η παρούσα εργασία αφορά τη μελέτη, σχεδίαση και ανάπτυξη της διαδικτυακής πλατφόρμας UniMeet για τον ομαδικό προγραμματισμό συναντήσεων. Το πρόβλημα που αντιμετωπίζεται είναι η δυσκολία εύρεσης κοινής ημερομηνίας και ώρας όταν συμμετέχουν πολλά άτομα με διαφορετική διαθεσιμότητα. Μελετήθηκαν σύγχρονα συστήματα scheduling polls, όπως τα Doodle, WhenAvailable και Rallly, και εντοπίστηκαν λειτουργίες που είναι χρήσιμες σε ακαδημαϊκό περιβάλλον.

Η εφαρμογή υλοποιήθηκε σε Python με Flask και MariaDB, με PyMySQL για πρόσβαση στη βάση και Bootstrap, JavaScript, Fetch API, Flatpickr και Chart.js στο επίπεδο παρουσίασης. Το UniMeet υποστηρίζει δημιουργία ψηφοφοριών, δημόσιους συνδέσμους, συμμετοχή εγγεγραμμένων χρηστών και επισκεπτών, αυτόματη εγγραφή κατά την πρώτη ψήφο, live αποτελέσματα, recommendation score, ειδοποιήσεις, email reminders, comments, templates, κατηγορίες και tags, επαναλαμβανόμενα meetings, QR Codes, iCalendar και συνδέσμους προς εξωτερικά ημερολόγια. Παρέχονται επίσης προσωπικά και διαχειριστικά analytics και audit log.

Το αποτέλεσμα είναι ένα ολοκληρωμένο ακαδημαϊκό πρωτότυπο που συγκεντρώνει τη διαδικασία προγραμματισμού σε ένα ενιαίο περιβάλλον, μειώνει τη χειροκίνητη επικοινωνία και μπορεί να αποτελέσει βάση για μελλοντική παραγωγική εγκατάσταση και σύνδεση με ιδρυματικές υπηρεσίες.

«Design and Development of a Booking and Online Meeting Scheduling System»

Abstract

This thesis studies, designs and develops UniMeet, a web platform for coordinating group meetings through scheduling polls. The problem addressed is the difficulty of finding a common date and time when many participants have different availability. Existing scheduling services such as Doodle, WhenAvailable and Rally were reviewed in order to identify useful patterns and requirements for an academic environment.

The system was implemented in Python using Flask and MariaDB, with PyMySQL for database access and Bootstrap, JavaScript, the Fetch API, Flatpickr and Chart.js on the client side. UniMeet supports poll creation, public links, registered and guest voting, automatic account creation on a guest's first vote, live results, recommendation scoring, notifications, email reminders, comments, templates, categories and tags, recurring meetings, QR codes, iCalendar export and calendar links. Personal and administrative analytics as well as an audit log are also included.

The final result is a functional academic prototype that centralizes the scheduling workflow, reduces fragmented communication and provides a clear basis for future production deployment, institutional authentication and deeper calendar integration.

Ευχαριστίες

Θέλω να ευχαριστήσω τους γονείς μου για τη συμπαράστασή τους και τον επιβλέποντα μου για τη συνεχή καθοδήγηση, τις επιστημονικές παρατηρήσεις και τη συμβολή του κατά την εκπόνηση της πτυχιακής εργασίας.

Περιεχόμενα

Περίληψη	iv
Abstract.....	v
Ευχαριστίες.....	vi
Περιεχόμενα.....	vii
Κατάλογος Σχημάτων	x
Κεφάλαιο 1ο: Τα εισαγωγικά.....	xi
1.1 Εισαγωγή	xi
1.2 Αντικείμενο της πτυχιακής εργασίας	xi
1.3 Το πρόβλημα του ομαδικού προγραμματισμού	xi
1.4 Σκοπός και στόχοι.....	xii
1.5 Μεθοδολογία ανάπτυξης.....	xii
1.6 Βασικές λειτουργικές απαιτήσεις.....	xii
1.7 Συνεισφορά της εργασίας	xiii
1.8 Δομή της εργασίας.....	xiii
Κεφάλαιο 2ο: Παρόμοια συστήματα και η ανάγκη για το Τμήμα.....	9
2.1 Εισαγωγή	9
2.2 Ηλεκτρονικά συστήματα ομαδικού προγραμματισμού	9
2.3 Doodle	9
2.4 WhenAvailable	11
2.5 Rallly	13
2.6 Συγκριτική αξιολόγηση των συστημάτων	15
2.7 Η ανάγκη για ένα σύστημα προσαρμοσμένο στο ακαδημαϊκό Τμήμα	16
2.8 Λειτουργικές απαιτήσεις που προκύπτουν από τη σύγκριση	17
2.9 Διαφοροποίηση του UniMeet	17
Κεφάλαιο 3ο: Τεχνολογία και εργαλεία.....	18
3.1 Εισαγωγή	18
3.2 Η γλώσσα προγραμματισμού Python.....	18
3.3 Το Flask ως web framework	18
3.4 MariaDB και σχεσιακή αποθήκευση δεδομένων	19
3.5 Η βιβλιοθήκη PyMySQL	19
3.6 HTML5, CSS3 και Bootstrap 5.....	19

3.7	JavaScript και Fetch API.....	20
3.8	Flatpickr για ημερομηνία και ώρα.....	20
3.9	Chart.js και απεικόνιση αναλυτικών στοιχείων.....	20
3.10	SMTP και ειδοποιήσεις ηλεκτρονικού ταχυδρομείου	20
3.11	QR Codes.....	20
3.12	iCalendar, αρχεία ICS και ημερολόγια.....	21
3.13	Αρχιτεκτονική και διαχωρισμός επιπέδων	21
3.14	Πλεονεκτήματα και περιορισμοί του τεχνολογικού συνόλου.....	22
3.15	3.15 Συνοπτική αντιστοίχιση τεχνολογιών και λειτουργιών	22
Κεφάλαιο 4ο: Η πλατφόρμα UniMeet - Παρουσίαση και ανάλυση		23
4.1	Εισαγωγή	23
4.2	Γενική εικόνα και λογική αρχιτεκτονική.....	23
4.3	Σχεδιασμός της βάσης δεδομένων.....	23
4.3.1	Η βάση.....	24
4.4	Ρόλοι χρηστών και έλεγχος πρόσβασης	25
4.4.1	Εγγραφή, σύνδεση και διαχείριση λογαριασμού.....	25
4.4.2	Dashboard και προσωπική πλοήγηση	25
4.4.3	Δημιουργία νέας ψηφοφορίας.....	26
4.4.4	Κύκλος ζωής meeting	26
4.4.5	Δημόσια σελίδα meeting.....	26
4.4.6	Συμμετοχή επισκέπτη και αυτόματη εγγραφή	26
4.4.7	Αυτόματη αποθήκευση ψήφου	27
4.4.8	Live αποτελέσματα.....	27
4.4.9	Μηχανισμός recommendation	28
4.4.10	Επεξεργασία, κλείσιμο, επανάνοιγμα και διαγραφή	29
4.4.11	Υπενθυμίσεις και email outbox	29
4.4.12	Εσωτερικές ειδοποιήσεις	29
4.4.13	Σχόλια και συζήτηση.....	29
4.4.14	Templates και επαναχρησιμοποίηση	29
4.4.15	QR Code και δημόσιος σύνδεσμος.....	29
4.4.16	Calendar integration	29
4.4.17	Κατηγορίες, tags και επαναλαμβανόμενα meetings.....	30
4.4.18	Personal analytics	30
4.4.19	Admin panel, analytics και audit log	30
4.4.20	Αναζήτηση, φίλτρα και pagination.....	30

4.4.21	Ασφάλεια και validation.....	30
4.5	Παρουσίαση Πλατφόρμας	30
Κεφάλαιο 5ο:	Συμπεράσματα και μελλοντικές βελτιώσεις.....	46
5.1	Αποτίμηση της εργασίας.....	46
5.1.1	Πλεονεκτήματα της πλατφόρμας UniMeet.....	46
5.1.2	Περιορισμοί της παρούσας υλοποίησης.....	46
5.2	Προτεινόμενες μελλοντικές βελτιώσεις	46
5.2.1	Σύνδεση με ιδρυματικούς λογαριασμούς.....	46
5.2.2	Πλήρης συγχρονισμός με ημερολόγια	47
5.2.3	WebSockets ή Server-Sent Events.....	47
5.2.4	Εφαρμογή για κινητές συσκευές και PWA.....	47
5.2.5	Βελτίωση του αλγορίθμου πρότασης.....	47
BIBΛΙΟΓΡΑΦΙΑ		xlvi
ΠΑΡΑΡΤΗΜΑ		xlix
A.1	Δομή του έργου	xlix
A.2	Εξαρτήσεις και εκτέλεση.....	xlix
A.3	Μηχανισμός πρότασης καλύτερης επιλογής.....	xlix
A.4	Δημιουργία meeting και validation	xlix
A.5	Δημόσια ψήφος και αυτόματη εγγραφή επισκέπτη	1
A.6	Live αποτελέσματα και API	1
A.7	Αυτόματη αποθήκευση ψήφου στο frontend	1
A.8	Βασικά routes της εφαρμογής	1

Κατάλογος Σχημάτων

Εικόνα 2.1: Δημιουργία ομαδικής ψηφοφορίας στο Doodle	10
Εικόνα 2.2: Οθόνη συμμετοχής και δήλωσης διαθεσιμότητας στο Doodle	11
Εικόνα 2.3: Δημιουργία poll και προσθήκη ημερομηνιών στο WhenAvailable	12
Εικόνα 2.4: Παράδειγμα scheduling poll στο WhenAvailable	13
Εικόνα 2.5: Παράδειγμα scheduling poll του Rally	14
Εικόνα 2.6: Σύγκριση βασικών δυνατοτήτων των πλάνων Rally	15
Εικόνα 3.1: Λογική αρχιτεκτονική της εφαρμογής UniMeet	18
Εικόνα 3.2: Κύκλος επεξεργασίας ενός HTTP αιτήματος στην εφαρμογή.....	19
Εικόνα 3.3: Μηχανισμός live ενημέρωσης αποτελεσμάτων	20
Εικόνα 3.4: Εξωτερικές ενσωματώσεις της εφαρμογής	21
Εικόνα 3.5: Πλεονεκτήματα και τεχνικά σημεία προσοχής του επιλεγμένου stack.....	22
Εικόνα 4.1: Λειτουργικά επίπεδα της πλατφόρμας UniMeet	23
Εικόνα 4.2: Απλοποιημένο διάγραμμα οντοτήτων της βάσης	24
Εικόνα 4.3: Ρόλοι και επίπεδα πρόσβασης	25
Εικόνα 4.4: Κύκλος ζωής μιας ψηφοφορίας συνάντησης	26
Εικόνα 4.5: Ροή συμμετοχής εγγεγραμμένου χρήστη και επισκέπτη	27
Εικόνα 4.6: Αποθήκευση ψήφου και ζωντανή ενημέρωση αποτελεσμάτων	28
Εικόνα 4.7: Λογική κατάταξης προτεινόμενων ημερομηνιών	29
Εικόνα 4.8: Αρχική σελίδα της πλατφόρμας UniMeet	31
Εικόνα 4.9: Σελίδα δημιουργίας νέου λογαριασμού	31
Εικόνα 4.10: Σελίδα σύνδεσης χρήστη στην πλατφόρμα	32
Εικόνα 4.11: Σελίδα επαναφοράς κωδικού πρόσβασης	33
Εικόνα 4.12: Γραμμή πλοήγησης συνδεδεμένου χρήστη.....	33
Εικόνα 4.13: Συνοπτική προβολή του προσωπικού dashboard	33
Εικόνα 4.14: Αναλυτική προβολή του dashboard χρήστη.....	34
Εικόνα 4.15: Σελίδα διαχείρισης ψηφοφοριών που δημιουργήσε ο χρήστης	34
Εικόνα 4.16: Φόρμα δημιουργίας νέας ψηφοφορίας.....	36
Εικόνα 4.17: Φόρμα επεξεργασίας υπάρχουσας ψηφοφορίας	37
Εικόνα 4.18: Δημόσια σελίδα ψηφοφορίας με QR Code και πρόταση συστήματος.....	38
Εικόνα 4.19: Δημόσια σελίδα με live αποτελέσματα και συζήτηση.....	39
Εικόνα 4.20: Μενού διαχείρισης ψηφοφορίας από τον ιδιοκτήτη	40
Εικόνα 4.21: Σελίδα αποθηκευμένων templates ψηφοφοριών	40
Εικόνα 4.22: Σελίδα ψηφοφοριών στις οποίες συμμετείχε ο χρήστης	41
Εικόνα 4.23: Σελίδα προσωπικών αναλυτικών στοιχείων	41
Εικόνα 4.24: Σελίδα διαχειριστικών αναλυτικών στοιχείων	42
Εικόνα 4.25: Διαχειριστική προβολή χρηστών και συναντήσεων	43
Εικόνα 4.26: Σελίδα προφίλ χρήστη.....	44
Εικόνα 4.27: Σελίδα καταγραφής ενεργειών Audit Log.....	44
Εικόνα 4.28: Σελίδα Email Outbox	45
Εικόνα 4.29: Dashboard χρήστη σε σκοτεινή λειτουργία.....	45

Κεφάλαιο 1ο: Τα εισαγωγικά

1.1 Εισαγωγή

Ο προγραμματισμός μιας συνάντησης αποτελεί μια απλή διαδικασία, η οποία στην πράξη γίνεται αρκετά σύνθετη όταν συμμετέχουν πολλά άτομα. Η εύρεση κοινής ημερομηνίας και ώρας απαιτεί ανταλλαγή μηνυμάτων, συλλογή απαντήσεων, σύγκριση διαθεσιμότητας και τελικά επιλογή της καταλληλότερης πρότασης. Όσο αυξάνεται ο αριθμός των συμμετεχόντων, τόσο αυξάνεται και η πιθανότητα καθυστερήσεων, αντικρουόμενων απαντήσεων ή απώλειας πληροφορίας. [1] [2]

Το πρόβλημα είναι σημαντικό σε ένα ακαδημαϊκό περιβάλλον. Μέλη διδακτικού προσωπικού, φοιτητές, διοικητικό προσωπικό και ομάδες εργασίας χρειάζεται συχνά να συντονίσουν συνελεύσεις, συναντήσεις έργων, παρουσιάσεις πτυχιακών εργασιών, επιτροπές και άλλες δραστηριότητες. Η επικοινωνία μέσω email ή εφαρμογών μηνυμάτων είναι χρήσιμη, αλλά δεν έχει σχεδιαστεί για συστηματική συλλογή και αξιολόγηση διαθεσιμότητας.

Στο πλαίσιο της παρούσας πτυχιακής εργασίας σχεδιάστηκε και αναπτύχθηκε η διαδικτυακή πλατφόρμα UniMeet. Βασικός σκοπός της είναι να συγκεντρώνει σε ένα ενιαίο περιβάλλον τη δημιουργία ψηφοφορίας για πιθανές ημερομηνίες, τη συμμετοχή εγγεγραμμένων και νέων χρηστών, τη ζωντανή παρουσίαση των αποτελεσμάτων και την υποστήριξη του χρήστη στην επιλογή της καταλληλότερης ημερομηνίας.

Η εφαρμογή δεν περιορίζεται σε μια απλή φόρμα ψηφοφορίας. Περιλαμβάνει λογαριασμούς χρηστών, αυτόματη εγγραφή επισκέπτη κατά την πρώτη συμμετοχή, προφίλ, dashboards, ειδοποιήσεις, audit log, σχόλια, κατηγορίες και tags, templates, επαναλαμβανόμενες συναντήσεις, QR Codes, iCalendar και διασύνδεση με εξωτερικά ημερολόγια. Παράλληλα, υπολογίζεται αυτόματα βαθμολογία για κάθε επιλογή ώστε να προτείνεται η καλύτερη ημερομηνία με βάση τις συγκεντρωμένες ψήφους.

1.2 Αντικείμενο της πτυχιακής εργασίας

Αντικείμενο της εργασίας είναι η μελέτη, ο σχεδιασμός και η υλοποίηση ενός ολοκληρωμένου πληροφοριακού συστήματος για τον προγραμματισμό συναντήσεων μέσω διαδικτυακών ψηφοφοριών. Η εργασία συνδυάζει τη μελέτη υπάρχοντων συστημάτων με την ανάπτυξη μιας λειτουργικής web εφαρμογής και την αξιολόγηση των τεχνολογικών επιλογών που απαιτούνται για την υλοποίησή της.

Ο δημιουργός μιας ψηφοφορίας ορίζει το θέμα της συνάντησης, την περίοδο κατά την οποία επιτρέπεται η ψηφοφορία και μία ή περισσότερες προτεινόμενες ημερομηνίες και ώρες. Στη συνέχεια μοιράζεται έναν μοναδικό σύνδεσμο στους συμμετέχοντες. Οι χρήστες δηλώνουν θετική ή αρνητική προτίμηση και το σύστημα συγκεντρώνει τα αποτελέσματα σε πραγματικό χρόνο.

Ιδιαίτερη έμφαση δίνεται στην απλοποίηση της συμμετοχής. Ένας εγγεγραμμένος χρήστης ψηφίζει με τον λογαριασμό του, ενώ ένας επισκέπτης μπορεί να συμπληρώσει όνομα, επώνυμο και email κατά τη στιγμή της πρώτης ψήφου. Εφόσον το email δεν υπάρχει ήδη, δημιουργείται αυτόματα λογαριασμός και παράγεται προσωρινός τετραψήφιος κωδικός πρόσβασης.

1.3 Το πρόβλημα του ομαδικού προγραμματισμού

Σε μικρές ομάδες η εύρεση κοινού χρόνου μπορεί να ολοκληρωθεί με λίγα μηνύματα. Σε μεγαλύτερες ομάδες η διαδικασία μετατρέπεται σε πρόβλημα συντονισμού και οι απαντήσεις δεν

αποστέλλονται ταυτόχρονα, οι προτιμήσεις αλλάζουν και ο διοργανωτής αναγκάζεται να συγκεντρώνει χειροκίνητα πληροφορίες από διαφορετικές πηγές.

Ένα πληροφοριακό σύστημα μετατρέπει τη διαδικασία σε δομημένα δεδομένα. Κάθε προτεινόμενη ημερομηνία αποτελεί συγκεκριμένη επιλογή και κάθε απάντηση αποθηκεύεται ως ψήφος, επιτρέποντας τον αυτόματο υπολογισμό θετικών και αρνητικών απαντήσεων, ποσοστού αποδοχής και συνολικής βαθμολογίας.

1.4 Σκοπός και στόχοι

Κύριος σκοπός είναι η δημιουργία μιας εύχρηστης, λειτουργικής και επεκτάσιμης διαδικτυακής πλατφόρμας που μειώνει τον χρόνο και την πολυπλοκότητα της οργάνωσης συναντήσεων.

- Μελέτη γνωστών συστημάτων scheduling polls και αναγνώριση βασικών χαρακτηριστικών.
- Καταγραφή αναγκών ενός ακαδημαϊκού περιβάλλοντος.
- Δημιουργία ψηφοφορίας με μοναδικό δημόσιο σύνδεσμο και χρονικό όριο.
- Υποστήριξη εγγεγραμμένων χρηστών και επισκεπτών με αυτόματη εγγραφή κατά την πρώτη ψήφο.
- Αυτόματη αποθήκευση επιλογών και live ενημέρωση αποτελεσμάτων.
- Μηχανισμός recommendation για ανάδειξη της καταλληλότερης ημερομηνίας.
- Προσωπικό dashboard, analytics και διαχειριστικά εργαλεία.
- Ενσωμάτωση email, notifications, QR Code και iCalendar.
- Αρχιτεκτονική που μπορεί να επεκταθεί με πρόσθετες λειτουργίες.

1.5 Μεθοδολογία ανάπτυξης

Αρχικά προσδιορίστηκε το πρόβλημα και μελετήθηκαν υπάρχουσες πλατφόρμες. Στη συνέχεια σχεδιάστηκε το σχεσιακό μοντέλο και υλοποιήθηκε ο βασικός πυρήνας χρηστών, meetings, επιλογών και ψήφων. Μετά την επιβεβαίωση της βασικής ροής προστέθηκαν live αποτελέσματα, recommendation scoring, analytics, notifications, templates και εξωτερικές ενσωματώσεις.

Κάθε λειτουργία δοκιμαζόταν σε πραγματικά σενάρια χρήσης. Η διαδικασία guest συμμετοχής απλοποιήθηκε ώστε η εγγραφή να γίνεται κατά την πρώτη ψήφο, ενώ η εισαγωγή ημερομηνίας ανασχεδιάστηκε ώστε κάθε επιλογή να αντιστοιχεί σε μία σαφή ημερομηνία και ώρα.

1.6 Βασικές λειτουργικές απαιτήσεις

- Εγγραφή, σύνδεση, προφίλ και αλλαγή/επαναφορά κωδικού.
- Ρόλοι χρήστη και διαχειριστή.
- Δημιουργία, επεξεργασία και διαγραφή meeting από τον ιδιοκτήτη.
- Υποχρεωτικά χρονικά όρια ψηφοφορίας και τουλάχιστον μία επιλογή ημερομηνίας/ώρας.
- Μοναδικό public link και QR Code.
- Guest συμμετοχή με όνομα, επώνυμο και email.
- Ναι/Όχι με άμεση αποθήκευση χωρίς τελικό submit.
- Live counts και εμφάνιση συμμετεχόντων ανά επιλογή.
- Score, comments, categories, tags, templates και recurrence.

- In-app/email notifications, audit log και analytics.
- Εξαγωγή σε .ics και links ημερολογίου.

1.7 Συνεισφορά της εργασίας

Η συνεισφορά βρίσκεται στον συνδυασμό πολλών λειτουργιών σε μία συνεπή ροή. Το UniMeet συνδέει την ψηφοφορία με λογαριασμούς, live ενημέρωση, ιστορικό, analytics, ειδοποιήσεις και ημερολόγια.

1.8 Δομή της εργασίας

- Το Κεφάλαιο 1 παρουσιάζει το πρόβλημα, τον σκοπό, τους στόχους και τις απαιτήσεις.
- Το Κεφάλαιο 2 εξετάζει παρόμοια συστήματα και την ανάγκη προσαρμογής σε ακαδημαϊκό Τμήμα.
- Το Κεφάλαιο 3 παρουσιάζει τις τεχνολογίες και τα εργαλεία ανάπτυξης.
- Το Κεφάλαιο 4 αναλύει τη βάση, τους ρόλους, τις ροές και τις λειτουργίες του UniMeet.
- Το Κεφάλαιο 5 συνοψίζει τα αποτελέσματα, τους περιορισμούς και τις μελλοντικές βελτιώσεις.

Κεφάλαιο 2ο: Παρόμοια συστήματα και η ανάγκη για το Τμήμα

2.1 Εισαγωγή

Ο προγραμματισμός μιας συνάντησης στην οποία συμμετέχουν πολλά άτομα αποτελεί σύνθετο πρόβλημα συντονισμού. Για τον λόγο αυτό αναπτύχθηκαν διαδικτυακά συστήματα ομαδικού προγραμματισμού, τα οποία συγκεντρώνουν διαθέσιμες επιλογές και απαντήσεις σε ενιαίο περιβάλλον. Η βασική ιδέα των online scheduling polls είναι η δημιουργία ψηφοφορίας με προτεινόμενες ημερομηνίες και ώρες, η αποστολή συνδέσμου και η καταγραφή της διαθεσιμότητας.

Στο κεφάλαιο εξετάζονται Doodle, WhenAvailable και Rallly. Τα συστήματα ακολουθούν κοινό πυρήνα αλλά διαφοροποιούνται ως προς λογαριασμούς, όρια δωρεάν χρήσης, calendar integration, ειδοποιήσεις, αποτελέσματα και ιδιωτικότητα. [7] [9] [11].

2.2 Ηλεκτρονικά συστήματα ομαδικού προγραμματισμού

Τα online scheduling systems λειτουργούν ως ενδιάμεσο στάδιο ανάμεσα στην αρχική πρόθεση για πραγματοποίηση μιας συνάντησης και στην τελική καταχώρισή της σε ημερολόγιο. Η τυπική ροή περιλαμβάνει τέσσερα βήματα: ο διοργανωτής ορίζει το γεγονός, προσθέτει πιθανές ημερομηνίες και ώρες, διαμοιράζει την ψηφοφορία και συγκεντρώνει τις απαντήσεις. Στη συνέχεια επιλέγεται η καταλληλότερη χρονική στιγμή με βάση τη συνολική διαθεσιμότητα [2].

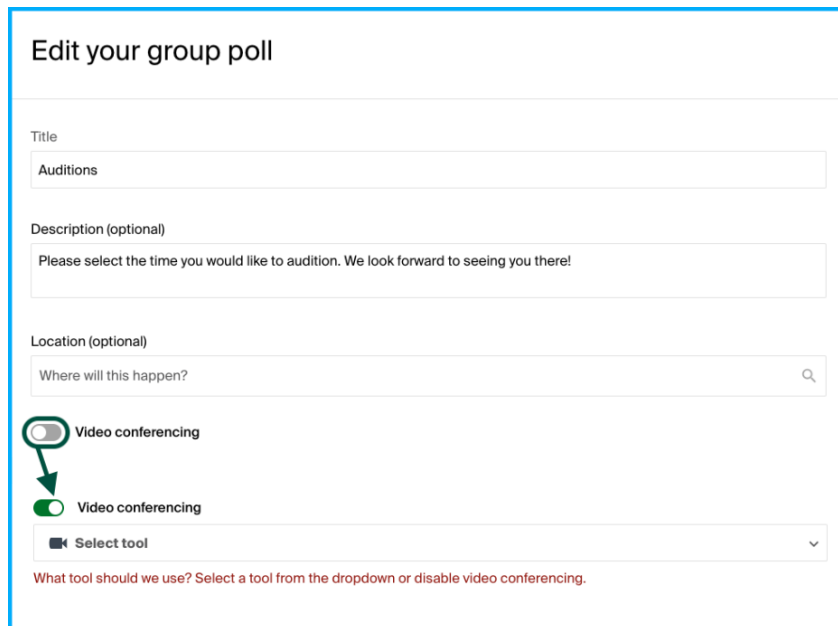
Σε πιο ώριμες υπηρεσίες η απλή καταμέτρηση επεκτείνεται με calendar integration, reminders, comments, τελική επιβεβαίωση, export και ρυθμίσεις ιδιωτικότητας. Η πρόσφατη έρευνα εξετάζει επίσης προσαρμοστικές αναπαραστάσεις διαθεσιμότητας και έξυπνη υποστήριξη της διαδικασίας, με στόχο τη μείωση του γνωστικού φόρτου όταν οι συμμετέχοντες και οι εναλλακτικές είναι πολλές [3].

Η αξία αυτών των εργαλείων είναι κυρίως οργανωτική και αντί κάθε απάντηση να βρίσκεται σε διαφορετικό μήνυμα, όλες οι επιλογές συγκεντρώνονται στο ίδιο σημείο, είναι σαφές ποιος έχει απαντήσει και ο διοργανωτής μπορεί να αναγνωρίσει γρήγορα τις επικρατέστερες επιλογές. Αυτή η ιδέα αποτελεί και τον βασικό λειτουργικό πυρήνα του UniMeet.

2.3 Doodle

Το Doodle είναι μία από τις πιο αναγνωρίσιμες υπηρεσίες ομαδικού προγραμματισμού. Η λειτουργία Group Poll επιτρέπει στον διοργανωτή να δημιουργήσει ένα γεγονός, να ορίσει τίτλο, περιγραφή και τοποθεσία, να προτείνει πολλές ημερομηνίες και ώρες και στη συνέχεια να δημιουργήσει έναν σύνδεσμο συμμετοχής. Η επίσημη τεκμηρίωση περιγράφει τη ροή ως δημιουργία των στοιχείων του γεγονότος, προσθήκη time slots, δημιουργία του poll και διαμοιρασμό του link [7].

Η δημιουργία μιας νέας Group Poll γίνεται από το dashboard του διοργανωτή. Κατά τη δημιουργία μπορούν να επιλεγούν week ή month views, να προστεθούν χρονικές επιλογές διαφορετικής διάρκειας και, ανάλογα με το πρόγραμμα χρήσης, να ενεργοποιηθούν επιπλέον ρυθμίσεις όπως automatic reminders ή απόκρυψη της λίστας συμμετεχόντων [7]. Με αυτή τη δομή ο διοργανωτής μπορεί να ρυθμίσει τα βασικά στοιχεία της ψηφοφορίας χωρίς να χρειάζεται να ακολουθήσει μια ιδιαίτερα σύνθετη διαδικασία.



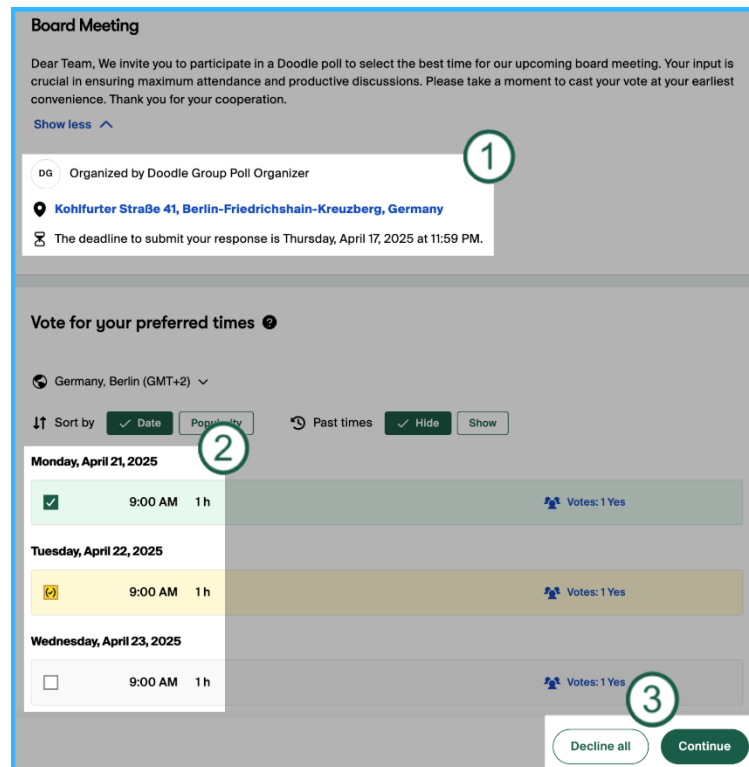
Εικόνα 2.1: Δημιουργία ομαδικής ψηφοφορίας στο Doodle

[\[https://help.doodle.com/en/articles/9457353-how-do-i-create-a-group-poll\]](https://help.doodle.com/en/articles/9457353-how-do-i-create-a-group-poll)

Στην Εικόνα 2.1 προτείνεται να χρησιμοποιηθεί στιγμιότυπο από την επίσημη σελίδα οδηγιών δημιουργίας Group Poll. Η σελίδα περιλαμβάνει τα βήματα εισαγωγής meeting information, επιλογής χρόνων και διαμοιρασμού του poll, επομένως αποτελεί άμεση και κατάλληλη πηγή για την οπτική τεκμηρίωση της διαδικασίας [7].

Για τον συμμετέχοντα η διαδικασία είναι ακόμη απλούστερη. Η Doodle αναφέρει ρητά ότι δεν απαιτείται λογαριασμός για συμμετοχή σε Group Poll. Ο χρήστης ανοίγει τον σύνδεσμο, δηλώνει τη διαθεσιμότητά του, επιλέγει Continue, συμπληρώνει όνομα και email και υποβάλλει την ψήφο [8]. Αυτό είναι ιδιαίτερα χρήσιμο όταν κάποιοι από τους συμμετέχοντες δεν έχουν ήδη λογαριασμό στην υπηρεσία.

Οι δυνατές απαντήσεις σε Group Poll είναι YES, IF-NEED-BE και NO. Η ενδιάμεση επιλογή IF-NEED-BE επιτρέπει στον συμμετέχοντα να δηλώσει ότι μία ώρα δεν είναι ιδανική αλλά μπορεί να χρησιμοποιηθεί εφόσον χρειαστεί [8]. Σε μια ομάδα όπου είναι δύσκολο να συμφωνήσουν όλοι απόλυτα, η ενδιάμεση επιλογή βοηθά τον διοργανωτή να καταλάβει ποιες ώρες είναι αποδεκτές ακόμη και αν δεν είναι ιδανικές.



Εικόνα 2.2: Οθόνη συμμετοχής και δήλωσης διαθεσιμότητας στο Doodle

[<https://help.doodle.com/en/articles/9457279-how-do-i-participate-in-a-group-poll>]

Στην Εικόνα 2.2 μπορεί να τοποθετηθεί στιγμιότυπο από την επίσημη οδηγία συμμετοχής, στην οποία παρουσιάζονται οι διαθέσιμες χρονικές επιλογές και ο τρόπος υποβολής της απάντησης. Το ίδιο υλικό τεκμηριώνει και την απουσία υποχρεωτικής εγγραφής για τον συμμετέχοντα [8].

Το Doodle δίνει επίσης έμφαση στη σύνδεση με ημερολόγια και υπηρεσίες τηλεδιάσκεψης. Η σύνδεση ημερολογίου βοηθά τον διοργανωτή να αποφεύγει προσωπικές συγκρούσεις κατά την επιλογή πιθανών ωρών, ενώ μπορούν να χρησιμοποιηθούν εργαλεία τηλεδιάσκεψης όπως Google Meet, Zoom, Microsoft Teams ή Webex ανάλογα με τις ρυθμίσεις και το πρόγραμμα χρήσης [7].

Μετά τη συλλογή των απαντήσεων, ο διοργανωτής επιλέγει την τελική ημερομηνία και ολοκληρώνει το event. Η πλατφόρμα επομένως καλύπτει όχι μόνο τη συλλογή προτιμήσεων αλλά και τη μετάβαση προς την τελική συνάντηση. Ωστόσο, αρκετές δυνατότητες διαχείρισης - όπως συγκεκριμένοι τύποι reminders, hidden participant list ή ορισμένες ενέργειες πάνω στις απαντήσεις - συνδέονται με το Professional plan [7].

Για ένα ακαδημαϊκό Τμήμα το Doodle προσφέρει ώριμη εμπειρία χρήσης, γνωστό interface και χαμηλή απαίτηση εκμάθησης. Από την άλλη πλευρά, πρόκειται για εξωτερική εμπορική υπηρεσία: οι πολιτικές λογαριασμών, οι λειτουργίες των πλάνων και η διαχείριση δεδομένων καθορίζονται από τον πάροχο. Ειδικές ανάγκες, όπως εσωτερικό audit, custom admin analytics ή πλήρως προσαρμοσμένη guest registration, δεν αποτελούν τον βασικό στόχο της υπηρεσίας.

2.4 WhenAvailable

Το WhenAvailable είναι υπηρεσία προσανατολισμένη σε availability polls και δίνει ιδιαίτερη έμφαση στην εύκολη δημιουργία πολλών πιθανών ημερομηνιών. Σύμφωνα με το επίσημο Help Center,

ο διοργανωτής εισέρχεται στη σελίδα Create Poll, συμπληρώνει τίτλο, organizer name και poll deadline και στη συνέχεια επιλέγει ημερομηνίες και ώρες απευθείας πάνω σε ημερολόγιο [9].

Ένα από τα βασικά χαρακτηριστικά του WhenAvailable είναι ότι οι ημερομηνίες επιλέγονται απευθείας μέσα από ημερολόγιο. Ο χρήστης μπορεί να επιλέγει ημερομηνίες πάνω στο calendar και να προσθέτει time slots με συγκεκριμένη διάρκεια. Η επίσημη τεκμηρίωση προτείνει ακόμη εισαγωγή του Google Calendar, ώστε ο διοργανωτής να ελέγχει τη δική του διαθεσιμότητα πριν δημιουργήσει τις επιλογές [9].

WhenAvailable

Help Center

Required fields:

- A title for the event
- Your name
- The date the poll closes

Event name *
Lunch Meeting

Your name *
Ann

RSVP by *
Tue Mar 03 2026

Event time zone
GMT-08:00 America/Los_Ang..

Event location (optional)
Restaurant TBD

Event details (optional)
Same agenda as last time. I'll reserve a table once I know which date works for y'all.

Create Poll

Vote options:

- ☐ Yes/Maybe/No
- ☒ Yes/No
- ☐ Yes

☐ Hide guests from other guests?

☐ Hide poll numbers from guests?

Εικόνα 2.3: Δημιουργία poll και προσθήκη ημερομηνιών στο WhenAvailable

[\[https://whenavailable.com/help_center/article_how_to_create_a_poll\]](https://whenavailable.com/help_center/article_how_to_create_a_poll)

Στην Εικόνα 2.3 μπορεί να χρησιμοποιηθεί ένα από τα επίσημα screenshots του Help Center, όπου εμφανίζονται τα στάδια Enter Event Details, Add Dates ή Add Times. Η συγκεκριμένη σελίδα είναι προτιμότερη από ένα τυχαίο screenshot, επειδή αποτελεί επίσημη τεκμηρίωση της τρέχουσας ροής δημιουργίας [9].

Μετά τη δημιουργία, ο σύνδεσμος μπορεί να διαμοιραστεί σε προσκεκλημένους και οι απαντήσεις συγκεντρώνονται στην ίδια poll. Η υπηρεσία είναι σχεδιασμένη ώστε η συμμετοχή να είναι γρήγορη και να μη χρειάζεται ο κάθε προσκεκλημένος να διαχειριστεί έναν πλήρη λογαριασμό για να δηλώσει διαθεσιμότητα. Έτσι μπορεί να χρησιμοποιηθεί εύκολα και σε συναντήσεις όπου συμμετέχουν άτομα εκτός της βασικής ομάδας.

Το WhenAvailable δεν περιορίζεται στην απλή επιλογή ημερομηνίας. Διαθέτει επιπλέον στοιχεία συνεργασίας, ενώ σε ανώτερα πλάνα παρέχονται λειτουργίες που σχετίζονται με reminders, μεγαλύτερες ομάδες, calendar sync και export δεδομένων. Η διαφοροποίηση των λειτουργιών ανά πρόγραμμα φαίνεται και στη σελίδα τιμολόγησης της υπηρεσίας [10].

Committee Meeting
Organizer: Steph

It's time to schedule our next meeting. I'll post the agenda a week before the meeting. Brunch will be provided. Please add to the comments any food preferences.

Main conference room, 2nd level, Central bldg

Last day to RSVP is Tue Apr 21, 2026

Message

Please include some GF options

Carol
1 min ago
Coffee ☹️, Coffee ☹️, COFFEE ☹️ ☹️ ☹️

Steph
1 min ago
If I can't eat, can we have bagels ☹️ and coffee ☹️ ☹️ ☹️

Leave a message...

Check the box to mark your preference for each date.
The options are: yes ☒, maybe ☐, and no ☒.
Time zone: "America/Los_Angeles".

Views: Calendar Table

Compact View

	WED Apr 29 9:00 AM - 11:00 AM	THU Apr 30 9:00 AM - 11:00 AM	TUE May 5 9:00 AM - 11:00 AM	WED May 6 9:00 AM - 11:00 AM	THU May 14 9:00 AM - 11:00 AM
	8 ☒	12 ☒	7 ☒	11 ☒	10 ☒
Rick rick@whenavailable.com	☒ ☐ ☒	☒ ☐ ☒	☒ ☐ ☒	☒ ☐ ☒	☒ ☐ ☒
Ashley	☒ ☐ ☒	☒ ☐ ☒	☒ ☐ ☒	☒ ☐ ☒	☒ ☐ ☒
Carol	☒ ☐ ☒	☒ ☐ ☒	☒ ☐ ☒	☒ ☐ ☒	☒ ☐ ☒
Danny	☒ ☐ ☒	☒ ☐ ☒	☒ ☐ ☒	☒ ☐ ☒	☒ ☐ ☒
Finn	☒ ☐ ☒	☒ ☐ ☒	☒ ☐ ☒	☒ ☐ ☒	☒ ☐ ☒
Hans	☒ ☐ ☒	☒ ☐ ☒	☒ ☐ ☒	☒ ☐ ☒	☒ ☐ ☒
Kiera	☒ ☐ ☒	☒ ☐ ☒	☒ ☐ ☒	☒ ☐ ☒	☒ ☐ ☒
Linda	☒ ☐ ☒	☒ ☐ ☒	☒ ☐ ☒	☒ ☐ ☒	☒ ☐ ☒
Rob	☒ ☐ ☒	☒ ☐ ☒	☒ ☐ ☒	☒ ☐ ☒	☒ ☐ ☒

Εικόνα 2.4: Παράδειγμα scheduling poll στο WhenAvailable

[<https://whenavailable.com/>]

Στην Εικόνα 2.4 μπορεί να χρησιμοποιηθεί στιγμιότυπο από την επίσημη αρχική σελίδα ή από ενεργό demo της υπηρεσίας. Το στιγμιότυπο είναι χρήσιμο για να φανεί η λογική της συγκεντρωτικής παρουσίασης επιλογών και συμμετοχής, σε αντιπαράθεση με τη φόρμα δημιουργίας της προηγούμενης εικόνας.

Το WhenAvailable συνδυάζει σχετικά απλή συμμετοχή με αρκετές πρόσθετες λειτουργίες διαχείρισης μιας ψηφοφορίας. Η χρήση deadline, η δυνατότητα οπτικού calendar selection και οι premium λειτουργίες για μεγαλύτερα σενάρια δείχνουν ότι η υπηρεσία στοχεύει τόσο σε απλές προσωπικές συναντήσεις όσο και σε επαναλαμβανόμενη επαγγελματική χρήση [9] [10].

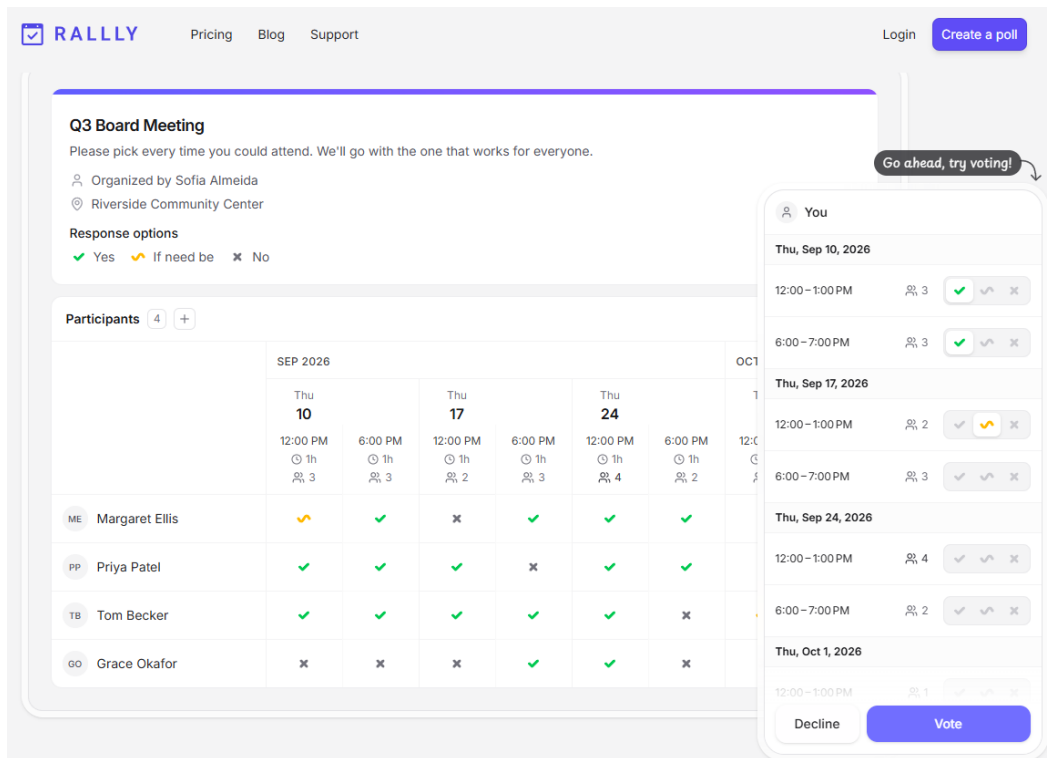
Από το WhenAvailable αξιοποιήθηκαν ως χρήσιμα παραδείγματα η ύπαρξη σαφούς προθεσμίας ψηφοφορίας, η άμεση προβολή των διαθέσιμων ωρών και η σύνδεση της ψηφοφορίας με υπενθυμίσεις και λειτουργίες ημερολογίου. Η πτυχιακή υλοποίηση υιοθετεί αντίστοιχη κατεύθυνση, αλλά διατηρεί τα δεδομένα και τις διαχειριστικές λειτουργίες στο ίδιο πληροφοριακό σύστημα.

2.5 Rallly

Το Rallly αποτελεί διαφορετική περίπτωση, επειδή συνδυάζει πολύ απλή εμπειρία scheduling poll με open-source προσανατολισμό. Η επίσημη ιστοσελίδα συνοψίζει τη διαδικασία σε τρία βήματα: Create a poll, Share the link και Vote and decide. Τονίζει επίσης ότι η βασική δημιουργία και η συμμετοχή μπορούν να πραγματοποιηθούν χωρίς λογαριασμό [11].

Στην αρχική σελίδα του Rallly εμφανίζεται απευθείας παράδειγμα ψηφοφορίας, ώστε ο επισκέπτης να καταλαβαίνει αμέσως τον τρόπο λειτουργίας της υπηρεσίας. Ο χρήστης βλέπει

προτεινόμενες ημερομηνίες, συμμετέχοντες και επιλογές Yes, If need be και No. Έτσι μπορεί να κατανοήσει τη λογική της υπηρεσίας χωρίς να προηγηθεί μεγάλη διαδικασία εκμάθησης [11].



Εικόνα 2.5: Παράδειγμα scheduling poll του Rallyly

[<https://rallyly.co/>]

Στην Εικόνα 2.5 προτείνεται να χρησιμοποιηθεί το interactive poll που εμφανίζεται απευθείας στην επίσημη αρχική σελίδα. Το συγκεκριμένο στιγμιότυπο δείχνει μαζί τις χρονικές επιλογές, τους συμμετέχοντες και το μοντέλο απαντήσεων, επομένως είναι κατάλληλο για τη συνοπτική παρουσίαση της υπηρεσίας [11].

Ένα σημαντικό χαρακτηριστικό του Rallyly είναι ότι ο διοργανωτής μπορεί να ξεκινήσει μια απλή poll χωρίς υποχρεωτική δημιουργία λογαριασμού. Αντίστοιχα, οι συμμετέχοντες λαμβάνουν τον σύνδεσμο και ψηφίζουν χωρίς sign-up. Με αυτόν τον τρόπο μειώνονται τα βήματα που απαιτούνται πριν από την ψηφοφορία, κάτι που βοηθά ιδιαίτερα σε έκτακτες συναντήσεις ή σε ομάδες με εξωτερικούς συμμετέχοντες. [11].

Η υπηρεσία διατηρεί επίσης σαφή διάκριση ανάμεσα στη δωρεάν και στην Pro χρήση. Η σελίδα pricing αναφέρει ότι το δωρεάν Hobby plan παρέχει unlimited scheduling polls και unlimited participants αλλά διατηρεί τις polls για 30 ημέρες, ενώ το Pro προσθέτει finalize date, extended poll lifetime, duplicate polls, advanced settings και άλλες επαγγελματικές δυνατότητες [12].

Σύγκριση προγραμμάτων		
Δείτε τι περιλαμβάνεται σε κάθε πρόγραμμα.		
	Χόμπι	Επαγγελματίας
Προγραμματισμός δημοσκοπήσεων	Απεριόριστος	Απεριόριστος
Συμμετέχοντες	Απεριόριστος	Απεριόριστος
Διατήρηση δημοσκοπήσεων	30 ημέρες	Αόριστος
Ημερομηνία οριστικοποίησης	×	⊕
Διπλότυπες δημοσκοπήσεις	×	⊕
Ρυθμίσεις δημοσκόπησης για προχωρημένους	×	⊕
Προσαρμοσμένη δημιουργία επωνυμίας	×	⊕
Κατάργηση αναφοράς	×	⊕
Ομαδική συνεργασία	×	⊕
Υποστήριξη προτεραιότητας	×	⊕

Εικόνα 2.6: Σύγκριση βασικών δυνατοτήτων των πλάνων Rallly

Στην Εικόνα 2.6 μπορεί να ενσωματωθεί το τμήμα Compare plans από την επίσημη σελίδα pricing. Η εικόνα βοηθά να φανεί ότι ακόμη και σε μια υπηρεσία με απλό και open-source προσανατολισμό, ορισμένες οργανωτικές λειτουργίες μεταφέρονται σε επαγγελματικό πλάνο [12].

Το γεγονός ότι το Rallly είναι open source είναι ενδιαφέρον και για ένα ακαδημαϊκό περιβάλλον, επειδή επιτρέπει τη μελέτη του τρόπου υλοποίησης και τη δυνατότητα αυτοφιλοξενίας. Η δυνατότητα εξέτασης του πηγαιού κώδικα ενισχύει τη διαφάνεια και δείχνει ότι ένα scheduling system μπορεί να υλοποιηθεί και να φιλοξενηθεί με μεγαλύτερο έλεγχο από τον οργανισμό που το χρησιμοποιεί. Παρόμοια λογική ακολουθήθηκε και στο UniMeet, το οποίο σχεδιάστηκε ώστε η εφαρμογή και τα δεδομένα της να μπορούν να ελέγχονται από τον ίδιο τον φορέα που τη χρησιμοποιεί.

Επιπλέον, η απλότητα της διεπαφής του Rallly αποτελεί χρήσιμο σχεδιαστικό παράδειγμα. Η μεγάλη ποσότητα λειτουργιών δεν πρέπει να μεταφέρεται απαραίτητα στην πρώτη οθόνη του συμμετέχοντα. Το UniMeet ακολουθεί παρόμοια αρχή στη δημόσια ψηφοφορία, όπου η βασική ενέργεια παραμένει η γρήγορη δήλωση Ναι ή Όχι.

2.6 Συγκριτική αξιολόγηση των συστημάτων

Τα τρία συστήματα μοιράζονται τον ίδιο λειτουργικό πυρήνα, αλλά διαφέρουν στη φιλοσοφία. Το Doodle διαθέτει ώριμο εμπορικό οικοσύστημα και ισχυρές integrations [7] [8]. Το WhenAvailable εστιάζει στην οπτική δημιουργία availability polls και σε λειτουργίες που επεκτείνονται ανά πλάνο [9] [10]. Το Rallly δίνει έμφαση στην απλότητα, στη γρήγορη συμμετοχή και σε open-source λογική [11] [12].

Χαρακτηριστικό	Doodle	WhenAvailable	Rallly
Δημιουργία scheduling poll	Ναι	Ναι	Ναι
Συμμετοχή χωρίς λογαριασμό	Ναι	Ναι / απλή συμμετοχή	Ναι
Κλίμακα απάντησης	Yes / If need be / No	Availability επιλογές	Yes / If need be / No

Calendar λειτουργίες	Ισχυρές integrations	Import/sync ανά λειτουργία ή πλάνο	Πιο περιορισμένη βασική ροή
Reminders / advanced settings	Μερικά σε Professional	Ανά premium πλάνο	Advanced settings στο Pro
Open source	Όχι	Όχι	Ναι
Κύριο πλεονέκτημα	Ωριμο οικοσύστημα	Ευέλικτη δημιουργία polls	Απλότητα και έλεγχος

Πίνακας 2.1: Συγκριτική παρουσίαση Doodle, WhenAvailable και Rally

Η σύγκριση δείχνει ότι και τα τρία συστήματα βασίζονται σε έναν εύκολο σύνδεσμο συμμετοχής, στην άμεση παρουσίαση των αποτελεσμάτων και στη δυνατότητα συνέχισης της διαδικασίας μετά την ψηφοφορία. Το UniMeet υιοθετεί τη λογική του εύκολου public link, αλλά τη συνδυάζει με αυτόματη δημιουργία λογαριασμού στην πρώτη guest ψήφο και με εσωτερικό ιστορικό [27] [29].

Ένα δεύτερο συμπέρασμα είναι ότι οι προηγμένες λειτουργίες συχνά διαφοροποιούν τα δωρεάν από τα επαγγελματικά πλάνα. Reminders, μεγαλύτερη διάρκεια ζωής poll, advanced settings ή ειδική διαχείριση συμμετεχόντων μπορεί να απαιτούν συνδρομή [7] [10] [12]. Στο UniMeet οι αντίστοιχες λειτουργίες αποτελούν μέρος της ίδιας ακαδημαϊκής υλοποίησης.

Ένα ακόμη θέμα που προκύπτει είναι το πόσο εύκολα μπορεί να συμμετέχει κάποιος χωρίς να χάνεται η δυνατότητα ταυτοποίησής του. Η απουσία υποχρεωτικής προεγγραφής μειώνει την τριβή, αλλά ένα ακαδημαϊκό σύστημα συχνά χρειάζεται ιστορικό, δυνατότητα ειδοποιήσεων και σαφή ταυτοποίηση. Η guest auto-registration του UniMeet σχεδιάστηκε ακριβώς για να καλύψει αυτή την ενδιάμεση ανάγκη [27] [28].

2.7 Η ανάγκη για ένα σύστημα προσαρμοσμένο στο ακαδημαϊκό Τμήμα

Ένα πανεπιστημιακό Τμήμα περιλαμβάνει πολλούς διαφορετικούς τύπους χρηστών και μεγάλο αριθμό επαναλαμβανόμενων διαδικασιών συντονισμού: συνελεύσεις, επιτροπές, παρουσιάσεις πτυχιακών και διπλωματικών εργασιών, project meetings, συναντήσεις με εξωτερικούς συνεργάτες και έκτακτες συσκέψεις. Σε μεγαλύτερες ομάδες η εύρεση απόλυτα κοινής ώρας γίνεται αντικειμενικά δυσκολότερη [1].

Η αποκλειστική χρήση email δημιουργεί κατακερματισμένες απαντήσεις και μεταφέρει το βάρος σύγκρισης στον διοργανωτή. Ένα κεντρικό poll διατηρεί κοινή εικόνα για όλους και επιτρέπει να εντοπιστεί όχι μόνο η απόλυτη ομοφωνία αλλά και η καλύτερη συμβιβαστική επιλογή, στοιχείο που συνδέεται με τη λογική των scheduling polls ως συλλογικών μηχανισμών απόφασης [2].

Σε ακαδημαϊκές συναντήσεις είναι επίσης συχνό να συμμετέχουν εξωτερικά μέλη, τα οποία δεν διαθέτουν ιδρυματικό λογαριασμό. Η υποχρεωτική προεγγραφή θα μπορούσε να καθυστερεί τη διαδικασία, ενώ η πλήρως ανώνυμη συμμετοχή δυσκολεύει την παρακολούθηση. Το UniMeet αντιμετωπίζει το πρόβλημα με guest identity και αυτόματη δημιουργία λογαριασμού κατά την πρώτη ψήφο [27] [28].

Ένα σύστημα που βρίσκεται υπό τον έλεγχο του Τμήματος μπορεί επιπλέον να προσαρμοστεί σε εσωτερικές κατηγορίες, ειδικούς ρόλους, audit log, στατιστικά και κανόνες διατήρησης δεδομένων. Αυτές οι λειτουργίες δεν είναι απαραίτητα κεντρικές σε μία γενικού σκοπού εμπορική υπηρεσία, αλλά έχουν αξία όταν η εφαρμογή εντάσσεται σε οργανωτική διαδικασία.

2.8 Λειτουργικές απαιτήσεις που προκύπτουν από τη σύγκριση

- Εύκολος δημόσιος σύνδεσμος ώστε ο διοργανωτής να μπορεί να διαμοιράζει μία ψηφοφορία χωρίς πολύπλοκη πρόσκληση.
- Συμμετοχή χωρίς υποχρεωτική προηγούμενη εγγραφή, αλλά με σαφή ταυτοποίηση του συμμετέχοντα.
- Πολλαπλές προτεινόμενες ημερομηνίες και σαφής αποθήκευση της επιλογής του χρήστη.
- Live αποτελέσματα και εύκολη σύγκριση των επιλογών κατά τη διάρκεια της ψηφοφορίας.
- Owner management για edit, close, reopen, delete και reminders.
- Σύνδεση του αποτελέσματος με calendar export και εξωτερικά ημερολόγια.
- Ιστορικό, analytics, audit και notifications για οργανωτική χρήση.
- Δυνατότητα επέκτασης χωρίς εξάρτηση από εμπορικό πλάνο τρίτου παρόχου.

Οι παραπάνω απαιτήσεις προκύπτουν τόσο από την παρατήρηση των λειτουργιών των Doodle, WhenAvailable και Rallly όσο και από τις ανάγκες του ακαδημαϊκού σεναρίου [7] [9] [11]. Στο Κεφάλαιο 4 παρουσιάζεται ο τρόπος με τον οποίο μεταφέρθηκαν στην τελική υλοποίηση του UniMeet [27] [28] [29].

2.9 Διαφοροποίηση του UniMeet

Στόχος του UniMeet δεν είναι να αντικαταστήσει υπηρεσίες όπως το Doodle ή το Rallly, αλλά να προσαρμόσει τη βασική λογική τους στις ανάγκες της συγκεκριμένης εργασίας και ενός ακαδημαϊκού Τμήματος. Βασίζεται στην προσαρμογή της ίδιας βασικής ιδέας σε ένα σύστημα που μπορεί να ελεγχθεί και να επεκταθεί στο πλαίσιο της πτυχιακής και ενός ακαδημαϊκού Τμήματος.

Η πρώτη διαφοροποίηση είναι η guest auto-registration. Ο μη συνδεδεμένος συμμετέχων εισάγει όνομα, επώνυμο και email κατά την πρώτη ψήφο. Εφόσον το email είναι νέο, δημιουργείται λογαριασμός και προσωρινός κωδικός, χωρίς να απαιτείται εγκατάλειψη της poll για ξεχωριστή διαδικασία εγγραφής [27] [28].

Η δεύτερη διαφοροποίηση είναι η ενσωμάτωση recommendation score, audit log, προσωπικών/admin analytics, templates, recurrence, in-app notifications και email outbox στην ίδια εφαρμογή [27] [28] [29]. Με τις παραπάνω λειτουργίες, η εφαρμογή δεν περιορίζεται μόνο στην επιλογή ημερομηνίας αλλά υποστηρίζει και τη διαχείριση της συνάντησης πριν και μετά την ψηφοφορία.

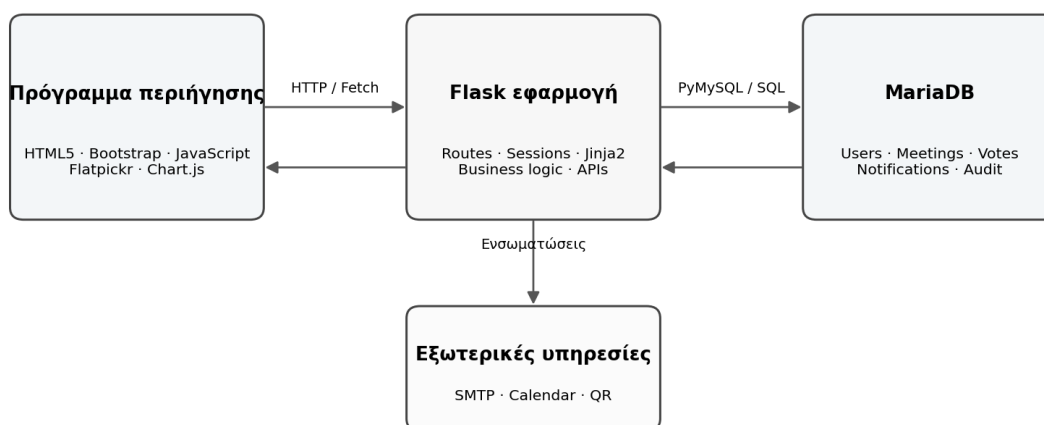
Το UniMeet αξιοποιεί ανοικτά πρότυπα όπως iCalendar και QR Code για διαλειτουργικότητα, ενώ η υλοποίηση παραμένει βασισμένη σε τεχνολογίες ανοικτού κώδικα. Έτσι το σύστημα μπορεί να εξελιχθεί μελλοντικά προς ιδρυματικό SSO, βαθύτερη calendar integration και παραγωγική εγκατάσταση.

Κεφάλαιο 3ο: Τεχνολογία και εργαλεία

3.1 Εισαγωγή

Η ανάπτυξη μιας σύγχρονης διαδικτυακής εφαρμογής προγραμματισμού συναντήσεων απαιτεί τεχνολογίες για server-side λογική, αποθήκευση δεδομένων, παρουσίαση και επικοινωνία. Για το UniMeet επιλέχθηκαν Python, Flask, MariaDB, PyMySQL, HTML5, CSS, Bootstrap 5 και JavaScript, ενώ Flatpickr και Chart.js καλύπτουν ειδικές ανάγκες. Επιπλέον αξιοποιούνται Fetch API, SMTP, QR Codes και iCalendar. [13] [14] [16] [18] [21]

Λογική αρχιτεκτονική της εφαρμογής UniMeet



Εικόνα 3.1: Λογική αρχιτεκτονική της εφαρμογής UniMeet

Στην Εικόνα 3.1 παρουσιάζεται η βασική λογική αρχιτεκτονική. Ο browser αποτελεί το επίπεδο παρουσίασης, το Flask εφαρμόζει τη λογική και προσπελαίνει τη MariaDB μέσω PyMySQL, ενώ SMTP, Calendar και QR λειτουργούν ως εξωτερικές ενσωματώσεις.

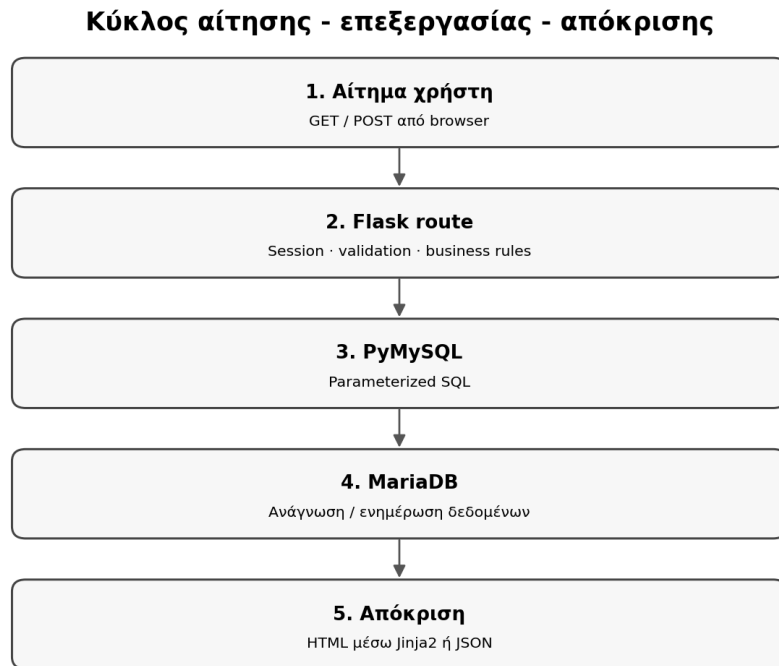
3.2 Η γλώσσα προγραμματισμού Python

Η Python είναι γλώσσα υψηλού επιπέδου και γενικού σκοπού με δυναμική τυποποίηση και σαφή σύνταξη. Η αναγνωσιμότητα και το μεγάλο οικοσύστημα βιβλιοθηκών την καθιστούν κατάλληλη για εκπαιδευτικές και web εφαρμογές. Στο UniMeet χρησιμοποιείται για users/sessions, meetings, votes, authorization, notifications, recommendation scoring, iCalendar, QR και JSON API endpoints. [13]

3.3 Το Flask ως web framework

Το Flask είναι ελαφρύ web framework για Python. Συνδέει URL routes με Python functions, υποστηρίζει requests/responses, sessions και templates. Η προσέγγιση ταιριάζει στο UniMeet επειδή

κάθε λειτουργία αντιστοιχίζεται σε σαφές endpoint και η ροή παραμένει εύκολα ανιχνεύσιμη. Για production deployment απαιτείται κατάλληλος WSGI server αντί του development server. [14]



Εικόνα 3.2: Κύκλος επεξεργασίας ενός HTTP αιτήματος στην εφαρμογή

Στην Εικόνα 3.2 φαίνεται ένα τυπικό request: το route παραλαμβάνει το αίτημα, εφαρμόζει session/validation, εκτελεί SQL και επιστρέφει HTML μέσω Jinja ή JSON.

3.4 MariaDB και σχεσιακή αποθήκευση δεδομένων

Η MariaDB αποτελεί σχεσιακό σύστημα διαχείρισης βάσεων δεδομένων ανοικτού κώδικα. Οι οντότητες users, meetings, meeting_options, votes, notifications, comments, tags και audit events έχουν σαφείς σχέσεις που εκφράζονται με primary/foreign keys. Unique constraints και indexes ενισχύουν την ακεραιότητα και την απόδοση. [16]

3.5 Η βιβλιοθήκη PyMySQL

Η PyMySQL λειτουργεί ως επίπεδο σύνδεσης Flask-MariaDB. Η εφαρμογή χρησιμοποιεί parameterized queries για να αποφεύγει απευθείας συνένωση τιμών χρήστη σε SQL και DictCursor ώστε οι στήλες να προσπελούνται με ονομαστικά keys. [17]

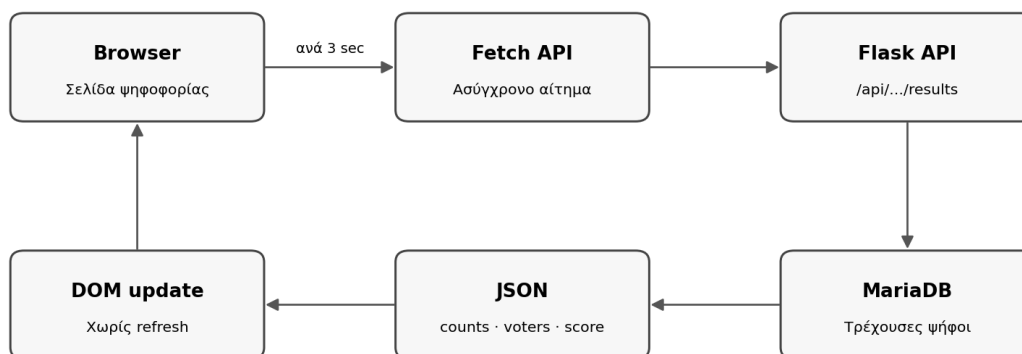
3.6 HTML5, CSS3 και Bootstrap 5

Το HTML5 ορίζει τη δομή των σελίδων και των forms, το CSS την οπτική παρουσίαση και το Bootstrap 5 προσφέρει mobile-first responsive grid, utilities και components. Η responsive σχεδίαση είναι απαραίτητη, καθώς ένας συμμετέχων μπορεί να ανοίξει το public poll από κινητό, ιδιαίτερα όταν χρησιμοποιεί το QR Code. [18] [19] [20]

3.7 JavaScript και Fetch API

Η JavaScript χρησιμοποιείται για αυτόματη αποθήκευση ψήφου, δυναμική προσθήκη επιλογών, αντιγραφή links, dark mode, countdown προσωρινού κωδικού και live ανανέωση. Το Fetch API επιτρέπει ασύγχρονα POST/GET requests με Promise/async-await. [21]

Live ενημέρωση αποτελεσμάτων με Fetch API



Εικόνα 3.3: Μηχανισμός live ενημέρωσης αποτελεσμάτων

Στην Εικόνα 3.3 παρουσιάζεται το polling: ο browser καλεί περιοδικά το API, το Flask διαβάζει τις ψήφους και επιστρέφει JSON, και η JavaScript ενημερώνει τα αντίστοιχα DOM στοιχεία.

3.8 Flatpickr για ημερομηνία και ώρα

Το Flatpickr είναι ελαφριά βιβλιοθήκη date/time picker με υποστήριξη 24ωρης ώρας και ελληνικού localization. Μειώνει σφάλματα εισαγωγής ημερομηνιών και χρησιμοποιείται στις φόρμες δημιουργίας και επεξεργασίας meeting. [22]

3.9 Chart.js και απεικόνιση αναλυτικών στοιχείων

Το Chart.js δημιουργεί γραφήματα σε HTML canvas. Στο UniMeet χρησιμοποιείται στα προσωπικά και διαχειριστικά analytics ώστε συγκεντρωτικά SQL δεδομένα να μετατρέπονται σε γρήγορα αναγνώσιμες οπτικοποιήσεις. [23]

3.10 SMTP και ειδοποιήσεις ηλεκτρονικού ταχυδρομείου

Το SMTP χρησιμοποιείται για αποστολή email invitations και reminders. Η εφαρμογή διαθέτει outbox fallback: όταν δεν υπάρχει διαθέσιμο SMTP configuration, το μήνυμα καταγράφεται ώστε να μπορεί να ελεγχθεί από το admin περιβάλλον. [4] [24]

3.11 QR Codes

Τα QR Codes κωδικοποιούν τον public URL μιας ψηφοφορίας και διευκολύνουν τη μεταφορά του συνδέσμου σε κινητό. Η παραγωγή γίνεται server-side με το πακέτο qrcode. [6] [25]

3.12 iCalendar, αρχεία ICS και ημερολόγια

Το iCalendar (RFC 5545) αποτελεί ανοικτό πρότυπο ανταλλαγής συμβάντων. Το UniMeet παράγει αρχείο .ics και επιπλέον links για Google και Outlook Calendar, ώστε η τελική συνάντηση να μεταφέρεται εύκολα σε εφαρμογές ημερολογίου. [5]



Εικόνα 3.4: Εξωτερικές ενσωματώσεις της εφαρμογής

Στην Εικόνα 3.4 συνοψίζονται οι εξωτερικές ενσωματώσεις για email, QR και calendars.

3.13 Αρχιτεκτονική και διαχωρισμός επιπέδων

Παρότι ο backend κώδικας της τελικής έκδοσης βρίσκεται σε κεντρικό app.py, οι ευθύνες διαχωρίζονται λογικά: presentation στον browser/Jinja, business rules στα Flask routes/helpers και persistence στη MariaDB. Ο διαχωρισμός επιτρέπει να αναλυθεί η εφαρμογή ως πολυεπίπεδο σύστημα.

3.14 Πλεονεκτήματα και περιορισμοί του τεχνολογικού συνόλου



Εικόνα 3.5: Πλεονεκτήματα και τεχνικά σημεία προσοχής του επιλεγμένου stack

Το stack είναι ελαφρύ, ανοικτού κώδικα, κατανοητό και κατάλληλο για ακαδημαϊκό prototype. Σημεία προσοχής είναι η ανάγκη production WSGI server, η σωστή διαχείριση database connections, το polling σε μεγάλη κλίμακα και η εξάρτηση από CDN αν τα frontend assets δεν φιλοξενούνται τοπικά.

3.15 Συνοπτική αντιστοίχιση τεχνολογιών και λειτουργιών

Τεχνολογία	Επίπεδο	Χρήση στο UniMeet	Κύριο όφελος
Python	Backend	Business logic	Αναγνωσιμότητα
Flask	Web framework	Routes/sessions/templates	Ελαφρύ framework
MariaDB	Database	Μόνιμα δεδομένα	Σχεσιακή ακεραιότητα
PyMySQL	Data access	Parameterized SQL	Άμεσος έλεγχος SQL
Bootstrap 5	Frontend	Responsive UI	Γρήγορη ομοιόμορφη διεπαφή
JavaScript/Fetch	Frontend/API	Votes/live results	Ασύγχρονη λειτουργία
Flatpickr	UI	Date/time	Μείωση λαθών
Chart.js	Analytics	Charts	Οπτικοποίηση
SMTP	Integration	Emails	Ειδοποιήσεις
QR Code	Integration	Public link	Πρόσβαση από κινητό
iCalendar	Integration	ICS export	Διαλειτουργικότητα

Πίνακας 3.1: Αντιστοίχιση τεχνολογιών και λειτουργιών

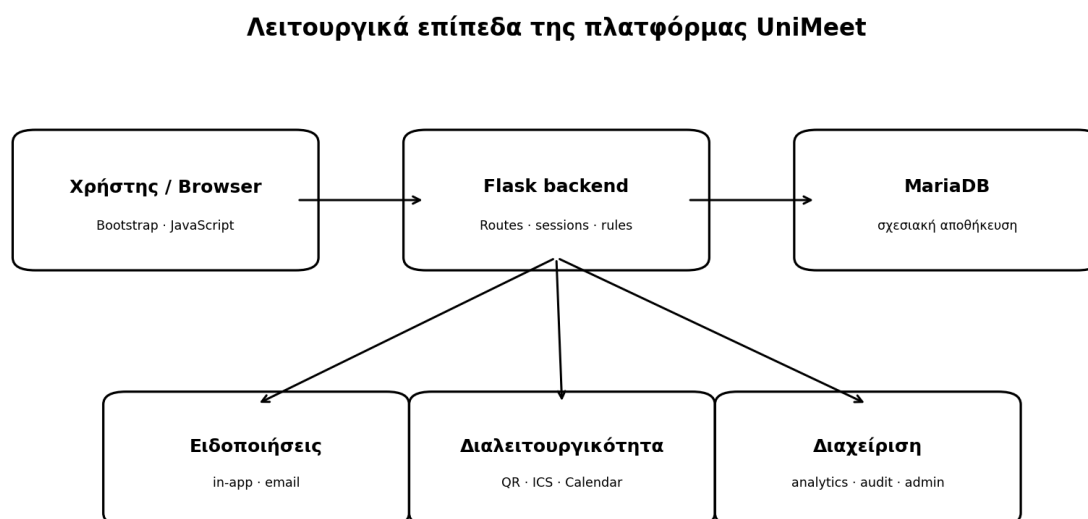
Κεφάλαιο 4ο: Η πλατφόρμα UniMeet - Παρουσίαση και ανάλυση

4.1 Εισαγωγή

Στο κεφάλαιο παρουσιάζεται αναλυτικά η πλατφόρμα UniMeet όπως υλοποιήθηκε στην τελική έκδοση v6.3. Το ενδιαφέρον μεταφέρεται στη λειτουργική δομή όπως ρόλοι, οντότητες βάσης, βασικές ροές και μηχανισμοί που συνδέουν τη δημιουργία ψηφοφορίας με τη συμμετοχή, την παρακολούθηση αποτελεσμάτων και την ολοκλήρωση της συνάντησης.

4.2 Γενική εικόνα και λογική αρχιτεκτονική

Η πλατφόρμα ακολουθεί client-server αρχιτεκτονική. Οι περισσότερες σελίδες παράγονται server-side από Flask/Jinja, ενώ η αποθήκευση ψήφου και η live ανανέωση πραγματοποιούνται ασύγχρονα με JavaScript και JSON endpoints. Ο Flask server ελέγχει session, permissions και validation και εκτελεί SQL μέσω PyMySQL. [27] [28] [29]



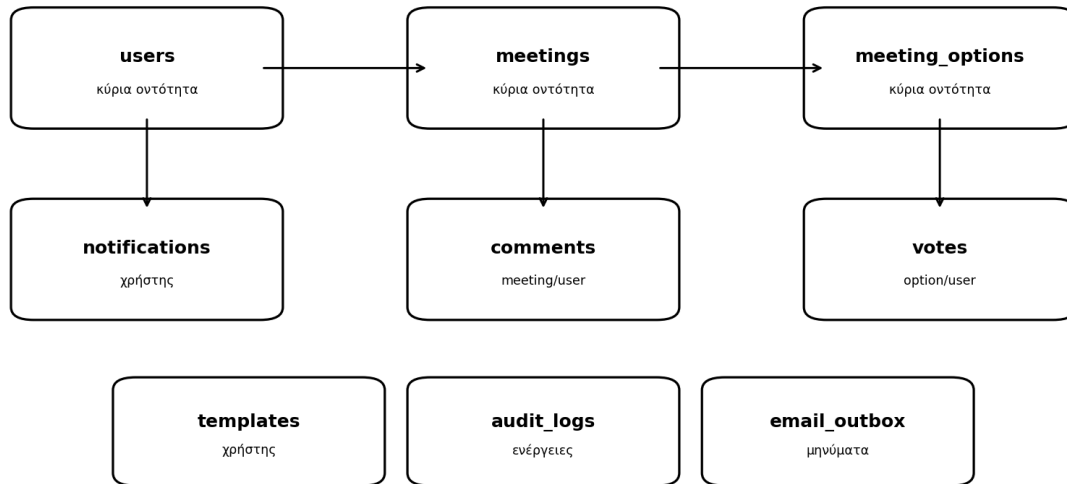
Εικόνα 4.1: Λειτουργικά επίπεδα της πλατφόρμας UniMeet

Στην Εικόνα 4.1 παρουσιάζεται η συνολική οργάνωση της εφαρμογής. Browser, Flask και MariaDB αποτελούν τον βασικό πυρήνα, ενώ notifications, interoperability και administration λειτουργούν οριζόντια.

4.3 Σχεδιασμός της βάσης δεδομένων

Η βάση unimeet_db χρησιμοποιεί utf8mb4/utf8mb4_unicode_ci και InnoDB. Περιλαμβάνει κύριες οντότητες users, meetings, meeting_options και votes και υποστηρικτικούς πίνακες για categories, tags, invitations, comments, notifications, audit logs, email outbox και templates. [28]

Απλοποιημένο διάγραμμα οντοτήτων της βάσης



Εικόνα 4.2: Απλοποιημένο διάγραμμα οντοτήτων της βάσης

Στην Εικόνα 4.2 απεικονίζονται οι βασικές σχέσεις. Ένας χρήστης δημιουργεί πολλά meetings, κάθε meeting περιλαμβάνει πολλές επιλογές και κάθε επιλογή συγκεντρώνει πολλές ψήφους.

4.3.1 Η βάση

4.3.1.1 Πίνακας users

Ο users αποθηκεύει username, email, first_name, last_name, password_hash, role και is_active. Για guest auto-registration υπάρχουν temporary_password_plain και temporary_password_visible_until. Η κανονική αυθεντικοποίηση βασίζεται στο password_hash.

4.3.1.2 Πίνακες meetings και meeting_options

Ο meetings περιγράφει την ψηφοφορία: owner, title, description, location, meeting_link, notes, visibility, voting window, public_token, status, final_option_id, category και recurrence metadata. Οι πιθανές ημερομηνίες αποθηκεύονται ως ανεξάρτητες εγγραφές meeting_options με title, starts_at, note και sort_order.

4.3.1.3 Πίνακας votes και μοναδικότητα ψήφου

Ο votes συνδέει option και user και αποθηκεύει choice yes/no μαζί με voter_name και voter_email. Unique constraints για option/user και option/email αποτρέπουν ανεπιθύμητες πολλαπλές ψήφους, ενώ η εφαρμογή ενημερώνει την υπάρχουσα ψήφο όταν ο χρήστης αλλάζει επιλογή.

4.3.1.4 Υποστηρικτικοί πίνακες

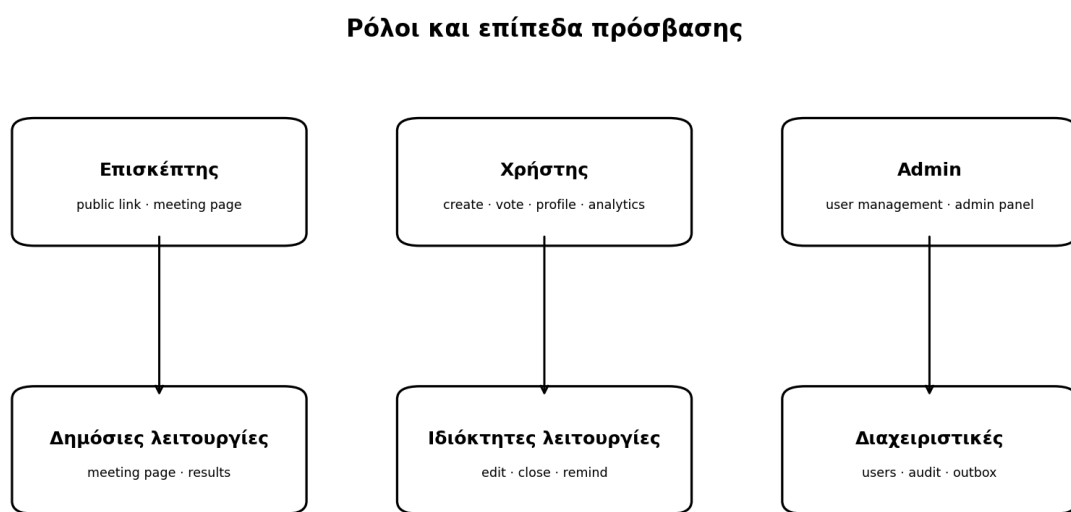
Πίνακας	Σκοπός	Κύρια συσχέτιση
categories	Κατηγορίες meeting	meetings.category_id
meeting_tags	Tags	meeting_id + tag
meeting_invites	Προσκλήσεις email	meeting_id + email
comments	Συζήτηση	meeting_id + user_id

notifications	Εσωτερικές ειδοποιήσεις	user_id
audit_logs	Ιχνηλασιμότητα	user/entity
email_outbox	Εξερχόμενα email	to_email
meeting_templates	Πρότυπα	user_id
meeting_template_options	Επιλογές template	template_id

Πίνακας 4.1: Υποστηρικτικοί πίνακες της βάσης δεδομένων

4.4 Ρόλοι χρηστών και έλεγχος πρόσβασης

Το σύστημα διακρίνει επισκέπτη, συνδεδεμένο χρήστη και admin. Decorators login_required και admin_required προστατεύουν τις αντίστοιχες λειτουργίες, ενώ edit/close/reopen/delete/remind/save-template ελέγχουν ownership ή admin role.



Εικόνα 4.3: Ρόλοι και επίπεδα πρόσβασης

Στην Εικόνα 4.3 παρουσιάζονται τα επίπεδα πρόσβασης. Οι public λειτουργίες είναι περιορισμένες, οι ιδιόκτητες απαιτούν authentication/ownership και οι διαχειριστικές απαιτούν admin.

4.4.1 Εγγραφή, σύνδεση και διαχείριση λογαριασμού

Η τυπική εγγραφή γίνεται από /register με έλεγχο μοναδικού username/email και password hashing. Η σύνδεση από /login επαληθεύει hash και ενεργό λογαριασμό και αποθηκεύει user_id/role στο session. Υποστηρίζονται forgot password, reset token και change password.

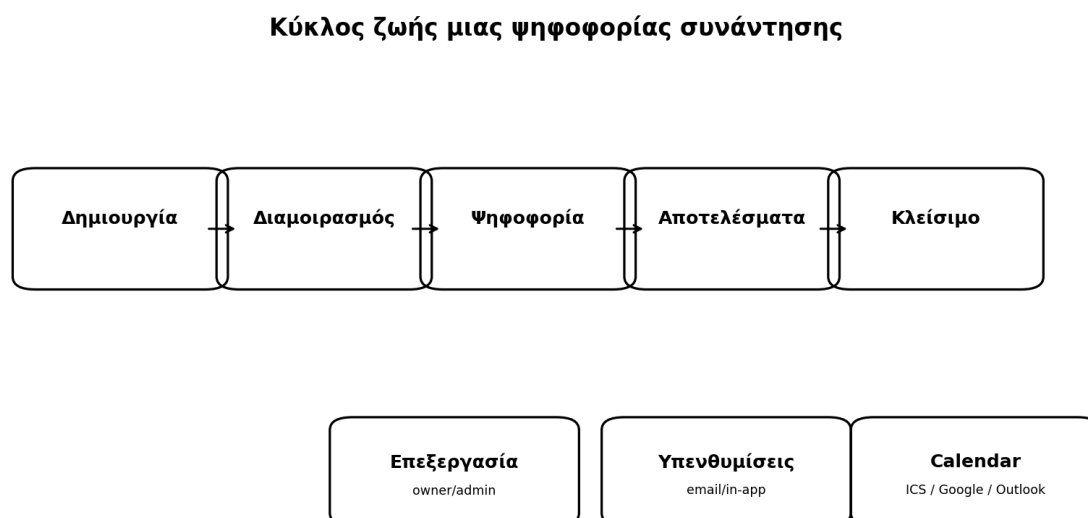
4.4.2 Dashboard και προσωπική πλοήγηση

Μετά τη σύνδεση, το dashboard συγκεντρώνει meetings που δημιούργησε ο χρήστης και meetings στα οποία συμμετείχε. Οι σελίδες /my/created και /my/voted υποστηρίζουν search, filters και pagination. Το profile εμφανίζει στοιχεία λογαριασμού και τον προσωρινό κωδικό όσο παραμένει εντός του χρονικού ορίου εμφάνισης.

4.4.3 Δημιουργία νέας ψηφοφορίας

Το /meeting/new συλλέγει title, description, location, meeting link, notes, category, tags, visibility, voting start/end και προτεινόμενες επιλογές. Κάθε επιλογή απαιτεί τίτλο και ημερομηνία/ώρα. Ο server κάνει validation και, σε σφάλμα, επαναφέρει τα ήδη συμπληρωμένα δεδομένα. Με επιτυχία παράγεται public_token και αποθηκεύονται meeting, options, tags και invites. Υποστηρίζεται εβδομαδιαία recurrence με parent_meeting_id και recurrence_index.

4.4.4 Κύκλος ζωής meeting



Εικόνα 4.4: Κύκλος ζωής μιας ψηφοφορίας συνάντησης

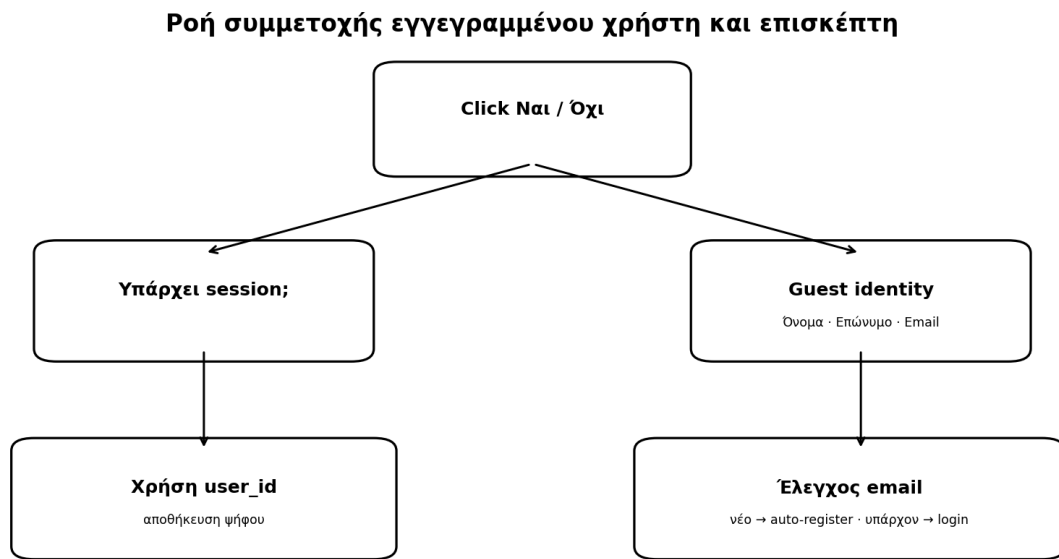
Στην Εικόνα 4.4 φαίνεται η πορεία από δημιουργία και διαμοιρασμό μέχρι ψηφοφορία, αποτελέσματα και κλείσιμο. Το status μπορεί να είναι open ή closed και υπάρχει δυνατότητα reopen. Τα voting_starts_at/voting_ends_at λειτουργούν ανεξάρτητα από το status.

4.4.5 Δημόσια σελίδα meeting

Κάθε meeting διαθέτει μοναδικό /m/<token>. Το token είναι μη προβλέψιμο και ανεξάρτητο από το numeric primary key. Η public page εμφανίζει τις επιλογές, counts, grouped voters, comments και recommendation πληροφορίες, ενώ owner-only στοιχεία όπως sharing/QR εμφανίζονται μόνο στον ιδιοκτήτη.

4.4.6 Συμμετοχή επισκέπτη και αυτόματη εγγραφή

Αν ο χρήστης δεν είναι συνδεδεμένος, πριν από την πρώτη ψήφο εισάγει όνομα, επώνυμο και email. Αν το email είναι νέο, παράγεται τυχαίος τετραψήφιος προσωρινός κωδικός, δημιουργείται λογαριασμός, ο χρήστης συνδέεται αυτόματα και η ψήφος αποθηκεύεται με το νέο user_id. Αν το email υπάρχει ήδη, ζητείται login ή password reset ώστε να αποτρέπεται πλαστοπροσωπία.



Εικόνα 4.5: Ροή συμμετοχής εγγεγραμμένου χρήστη και επισκέπτη

Στην Εικόνα 4.5 παρουσιάζεται ο έλεγχος session και email που καθορίζει αν η ψήφος θα συνδεθεί με υπάρχον user_id ή θα προηγηθεί auto-registration.

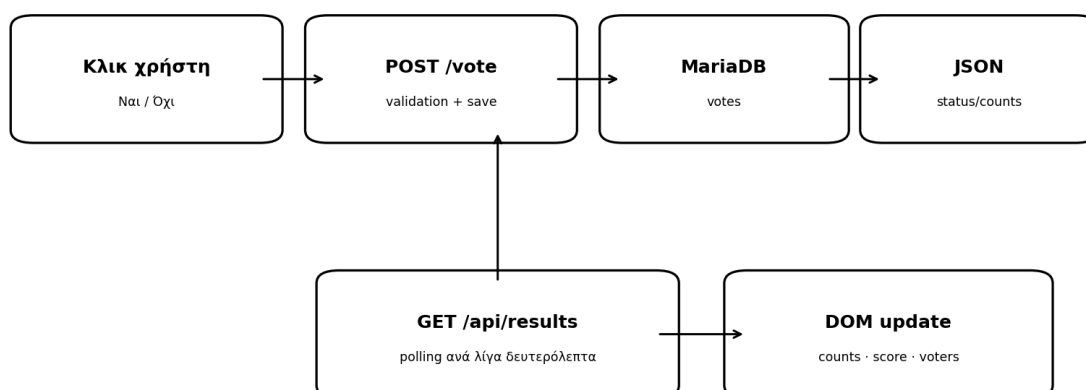
4.4.7 Αυτόματη αποθήκευση ψήφου

Η επιλογή Ναι/Όχι αποθηκεύεται με κάθε click χωρίς ξεχωριστό κουμπί οριστικοποίησης. Η JavaScript αποστέλλει POST request και ο server εισάγει ή ενημερώνει την εγγραφή votes. Με αυτό τον τρόπο μειώνονται τα βήματα και η πιθανότητα ο χρήστης να ξεχάσει submit.

4.4.8 Live αποτελέσματα

Το endpoint /api/m/<token>/results επιστρέφει JSON με yes/no counts, total, score, suitability, rank και grouped voter lists. Η public page καλεί περιοδικά το endpoint και ενημερώνει μόνο τα απαραίτητα DOM στοιχεία. [27] [29]

Αποθήκευση ψήφου και ζωντανή ενημέρωση αποτελεσμάτων



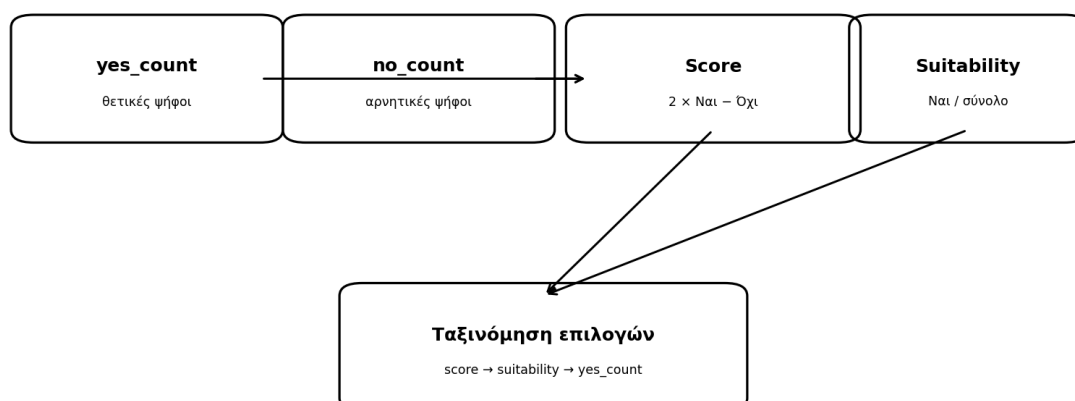
Εικόνα 4.6: Αποθήκευση ψήφου και ζωντανή ενημέρωση αποτελεσμάτων

Στην Εικόνα 4.6 παρουσιάζεται η αλληλεπίδραση vote endpoint, database και results API. Το polling διατηρεί την υποδομή απλή για την αναμενόμενη κλίμακα μιας ακαδημαϊκής ομάδας.

4.4.9 Μηχανισμός recommendation

Η helper recommendation_for_options υπολογίζει για κάθε επιλογή $\text{score} = 2 \times \text{yes_count} - \text{no_count}$ και $\text{suitability} = \text{yes} / \text{total} \times 100$. Οι επιλογές ταξινομούνται κατά score, suitability και αριθμό θετικών ψήφων. Η βαθμολογία λειτουργεί μόνο ως βοήθημα για τον διοργανωτή, ενώ οι πραγματικές ψήφοι παραμένουν ορατές ώστε να μπορεί να αξιολογήσει ο ίδιος το αποτέλεσμα.

Λογική κατάταξης προτεινόμενων ημερομηνιών



Εικόνα 4.7: Λογική κατάταξης προτεινόμενων ημερομηνιών

Στην Εικόνα 4.7 απεικονίζεται ο πυρήνας της κατάταξης και η μετατροπή των ψήφων σε αριθμητική βαθμολογία.

4.4.10 Επεξεργασία, κλείσιμο, επανάνοιγμα και διαγραφή

Ο ιδιοκτήτης ή admin μπορεί να επεξεργαστεί meeting και επιλογές, ενώ close, reopen και delete εκτίθενται ως POST actions. Η διαγραφή βασίζεται και στα cascading foreign keys ώστε να απομακρύνονται εξαρτώμενες εγγραφές.

4.4.11 Υπενθυμίσεις και email outbox

Το remind action συγκεντρώνει προσκεκλημένους και συμμετέχοντες και δημιουργεί email reminders. Όταν υπάρχει SMTP configuration γίνεται προσπάθεια αποστολής, διαφορετικά τα μηνύματα καταγράφονται στο email_outbox. Το /outbox επιτρέπει στον admin να βλέπει queued, sent ή failed entries.

4.4.12 Εσωτερικές ειδοποιήσεις

Ο πίνακας notifications και η helper notify δημιουργούν ειδοποιήσεις με title, message, link, kind και is_read. Οι in-app notifications παραμένουν διαθέσιμες ακόμη και όταν ένα email αποτύχει ή αγνοηθεί.

4.4.13 Σχόλια και συζήτηση

Η δημόσια σελίδα υποστηρίζει comments που συνδέονται με meeting και user. Τα σχόλια επιτρέπουν στους συμμετέχοντες να προσθέτουν διευκρινίσεις, για παράδειγμα σχετικά με την ώρα ή την τοποθεσία, χωρίς οι πληροφορίες αυτές να μεταβάλλουν τη βαθμολογία των επιλογών.

4.4.14 Templates και επαναχρησιμοποίηση

Ένα υπάρχον meeting μπορεί να αποθηκευτεί ως template. Αντιγράφονται title, description, location, link, notes, category και οι τίτλοι/σημειώσεις των επιλογών, ενώ οι ακριβείς ημερομηνίες επανακαθορίζονται σε κάθε νέο meeting. Η σελίδα /templates εμφανίζει τα templates του τρέχοντος χρήστη.

4.4.15 QR Code και δημόσιος σύνδεσμος

Το /m/<token>/qr.png παράγει δυναμικό QR Code με τον public URL. Η λειτουργία είναι διαθέσιμη στον owner για εύκολο διαμοιρασμό, ιδιαίτερα όταν οι συμμετέχοντες χρησιμοποιούν κινητές συσκευές.

4.4.16 Calendar integration

Το /calendar.ics δημιουργεί συμβάν iCalendar για την τελική επιλογή, ενώ calendar-links παράγει links προς Google και Outlook. Η υλοποίηση δεν πραγματοποιεί ακόμη αμφίδρομο συγχρονισμό με προσωπικά ημερολόγια, αλλά προσφέρει διαλειτουργική εξαγωγή.

4.4.17 Κατηγορίες, tags και επαναλαμβανόμενα meetings

Οι categories παρέχουν προκαθορισμένη ομαδοποίηση, τα meeting_tags ελεύθερες ετικέτες και η weekly recurrence δημιουργεί συσχετισμένα meetings σε επόμενες εβδομάδες. Οι κατηγορίες, τα tags και οι επαναλήψεις γίνονται περισσότερο χρήσιμα όταν ο χρήστης έχει να διαχειριστεί μεγάλο αριθμό ψηφοφοριών.

4.4.18 Personal analytics

Το /analytics παρουσιάζει συγκεντρωτικά στοιχεία χρήσης του τρέχοντος χρήστη. SQL aggregations παράγουν counts και χρονικές κατανομές, οι οποίες προβάλλονται με Chart.js.

4.4.19 Admin panel, analytics και audit log

Το admin panel παρέχει συνολική εικόνα χρηστών και δραστηριότητας. Το audit log καταγράφει ενέργειες όπως δημιουργία meeting, αποθήκευση ψήφου και διαχειριστικές μεταβολές. Το Audit Log δεν αποτελεί πλήρες σύστημα παρακολούθησης ασφάλειας, αλλά επιτρέπει στον διαχειριστή να βλέπει πότε και από ποιον πραγματοποιήθηκαν σημαντικές ενέργειες.

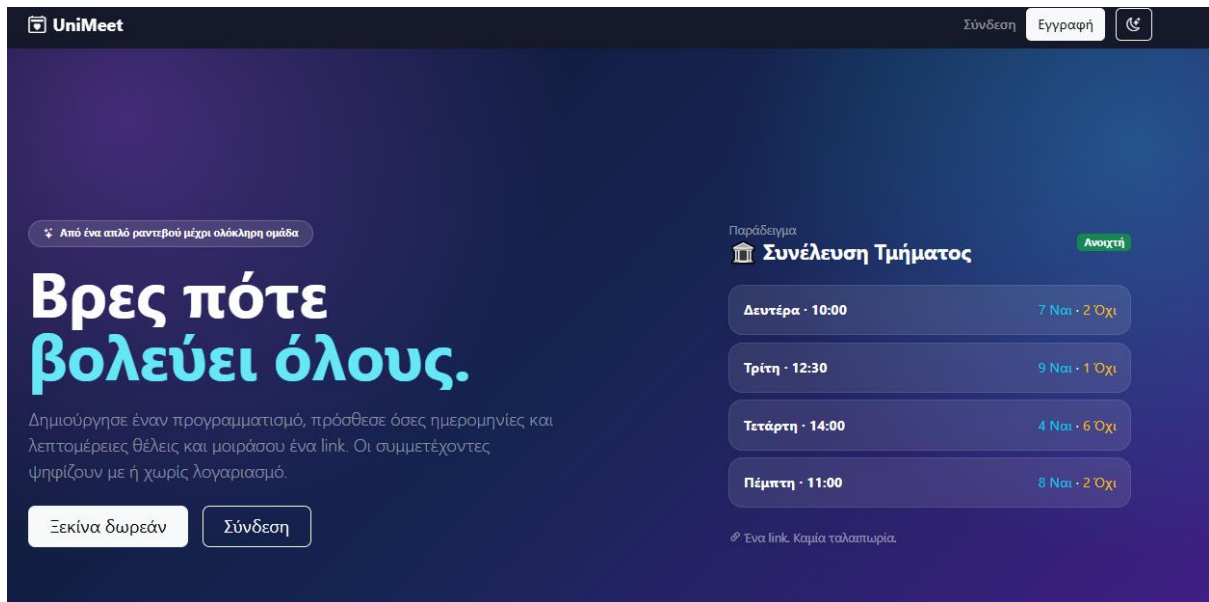
4.4.20 Αναζήτηση, φίλτρα και pagination

Οι λίστες meetings υποστηρίζουν search/filter/pagination, ώστε η εφαρμογή να παραμένει πρακτική όταν αυξάνεται ο αριθμός εγγραφών. Τα φίλτρα μπορούν να βασίζονται σε status, category ή λέξεις-κλειδιά.

4.4.21 Ασφάλεια και validation

Η ασφάλεια βασίζεται σε password hashing, parameterized SQL, session-based authentication, decorators και owner/admin authorization. Τα public tokens δεν εκθέτουν numeric IDs. Η validation λογική ελέγχει required fields, datetime σειρά, valid choices και ownership. Για production χρήση προτείνονται επιπλέον CSRF protection, rate limiting, HTTPS και αυστηρότερα security headers.

4.5 Παρουσίαση Πλατφόρμας



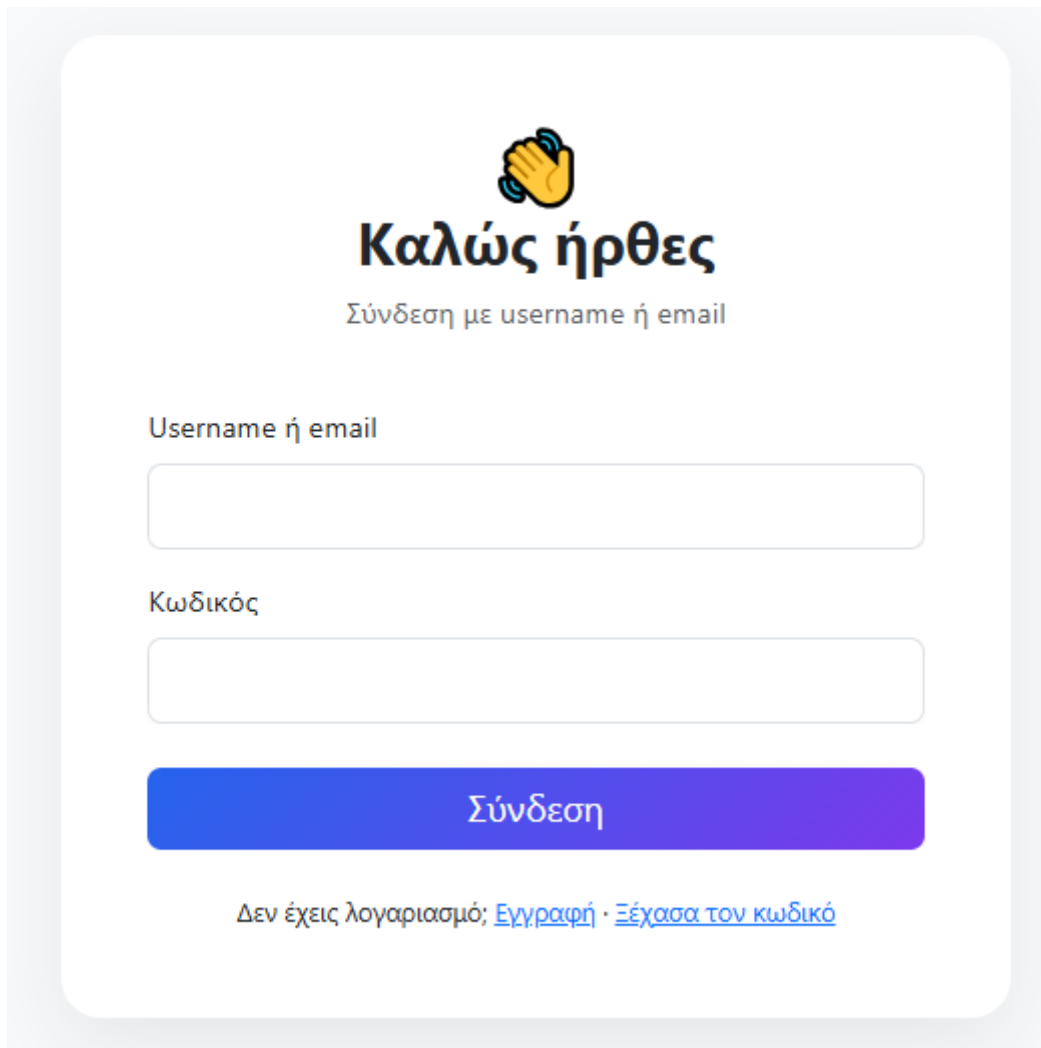
Εικόνα 4.8: Αρχική σελίδα της πλατφόρμας UniMeet

Στην Εικόνα 4.8 παρουσιάζεται η αρχική σελίδα της πλατφόρμας UniMeet, η οποία αποτελεί το πρώτο σημείο επαφής του χρήστη με το σύστημα. Στο επάνω μέρος εμφανίζεται η βασική γραμμή πλοήγησης με τις επιλογές σύνδεσης και εγγραφής, ενώ στο κεντρικό τμήμα προβάλλεται ο βασικός σκοπός της εφαρμογής, δηλαδή η εύκολη εύρεση κοινής ημερομηνίας για μια συνάντηση. Παράλληλα, παρουσιάζεται ενδεικτικό παράδειγμα ψηφοφορίας με διαφορετικές ημέρες και ώρες, ώστε ο επισκέπτης να αντιλαμβάνεται άμεσα τον τρόπο λειτουργίας της πλατφόρμας. Από την ίδια σελίδα ο χρήστης μπορεί να προχωρήσει είτε στη σύνδεση είτε στη δημιουργία νέου λογαριασμού.

Εικόνα 4.9: Σελίδα δημιουργίας νέου λογαριασμού

Στην Εικόνα 4.9 φαίνεται η σελίδα εγγραφής νέου χρήστη. Η φόρμα ζητά όνομα, επώνυμο, username, email και κωδικό πρόσβασης, ώστε να δημιουργηθεί ένας πλήρης λογαριασμός στο

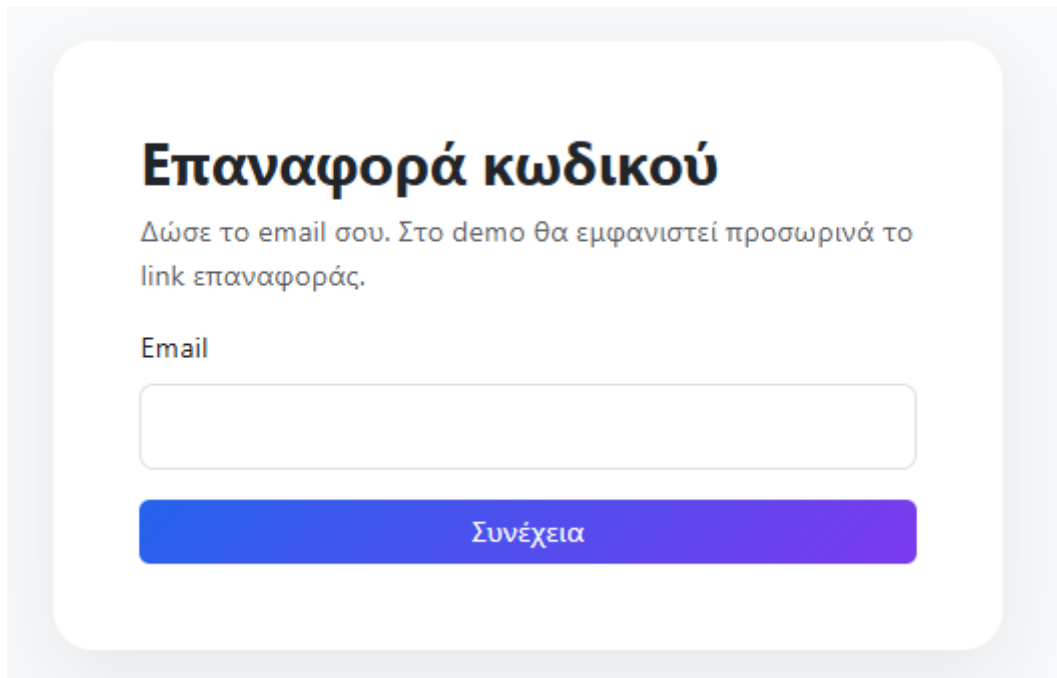
UniMeet. Η διάταξη παραμένει απλή και συγκεντρώνει σε ένα σημείο μόνο τα απαραίτητα στοιχεία για την ολοκλήρωση της εγγραφής.



The image shows a login page for the UniMeet platform. At the top, there is a yellow hand icon with blue lines, followed by the text "Καλώς ήρθες" (Welcome) in a large, bold, black font. Below this, the text "Σύνδεση με username ή email" (Login with username or email) is displayed in a smaller, black font. There are two input fields: the first is labeled "Username ή email" and the second is labeled "Κωδικός" (Password). Below the input fields is a large blue button with the text "Σύνδεση" (Login). At the bottom, there is a link that says "Δεν έχεις λογαριασμό; Εγγραφή · Ξέχασα τον κωδικό" (Don't have an account? Register · Forgot the password).

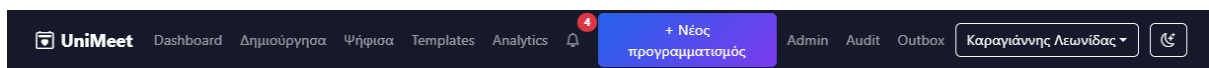
Εικόνα 4.10: Σελίδα σύνδεσης χρήστη στην πλατφόρμα

Η Εικόνα 4.10 απεικονίζει τη φόρμα σύνδεσης στην πλατφόρμα. Ο χρήστης μπορεί να εισαγάγει username ή email μαζί με τον κωδικό πρόσβασης και να μεταβεί στο προσωπικό του περιβάλλον. Στο κάτω μέρος υπάρχουν επίσης σύνδεσμοι για εγγραφή νέου λογαριασμού και για επαναφορά κωδικού, ώστε οι βασικές ενέργειες πρόσβασης να είναι άμεσα διαθέσιμες.



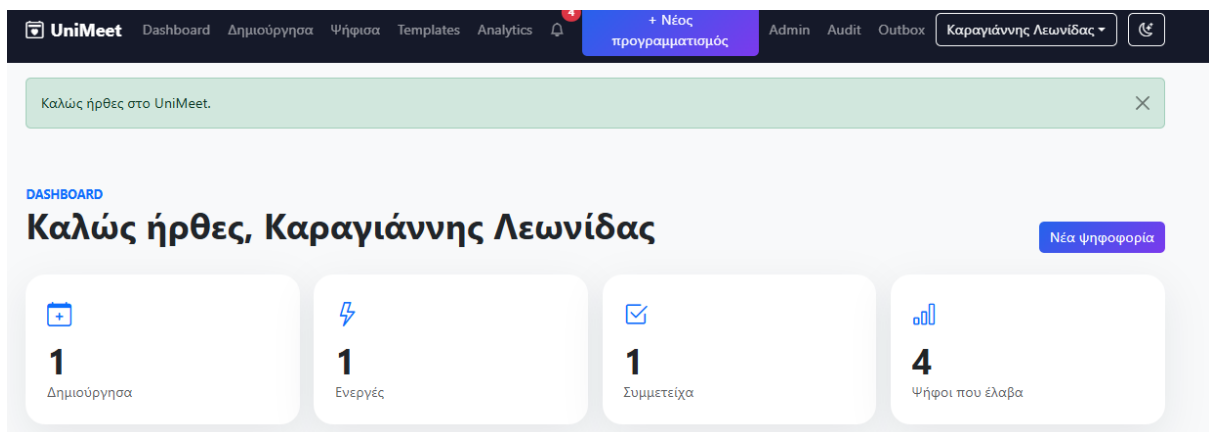
Εικόνα 4.11: Σελίδα επαναφοράς κωδικού πρόσβασης

Μέσω της Εικόνας 4.11 παρουσιάζεται η διαδικασία έναρξης επαναφοράς κωδικού πρόσβασης. Ο χρήστης εισάγει το email του και το σύστημα προχωρά στη δημιουργία της αντίστοιχης διαδικασίας reset. Η οθόνη είναι σκόπιμα λιτή, καθώς απαιτείται μόνο το στοιχείο που είναι απαραίτητο για την αναζήτηση του λογαριασμού.



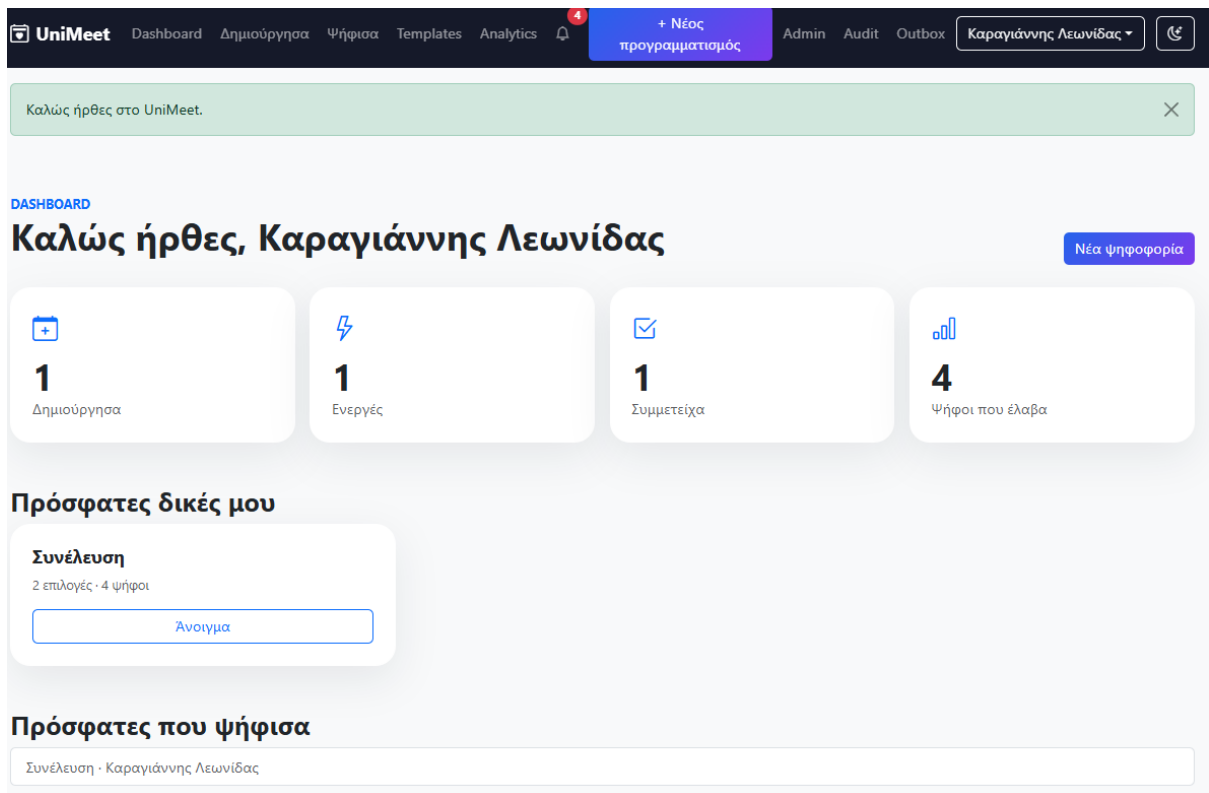
Εικόνα 4.12: Γραμμή πλοήγησης συνδεδεμένου χρήστη

Στην Εικόνα 4.12 διακρίνεται η κεντρική γραμμή πλοήγησης μετά τη σύνδεση ενός χρήστη. Από αυτή παρέχεται πρόσβαση στο Dashboard, στις ψηφοφορίες, στα templates, στα analytics και, όταν ο χρήστης είναι διαχειριστής, στις αντίστοιχες διαχειριστικές λειτουργίες. Ξεχωρίζει επίσης το κουμπί δημιουργίας νέου προγραμματισμού, το οποίο επιτρέπει γρήγορη μετάβαση στη φόρμα νέας ψηφοφορίας.



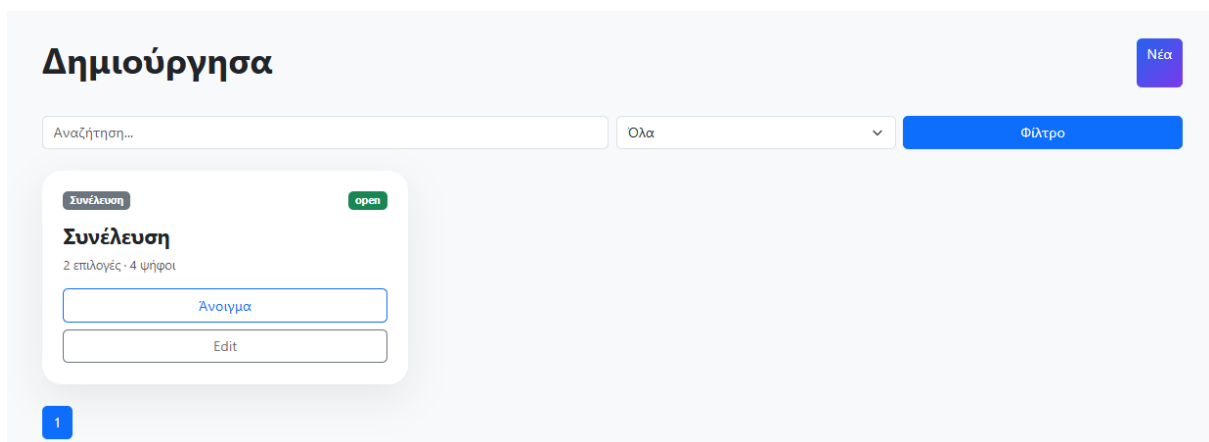
Εικόνα 4.13: Συνοπτική προβολή του προσωπικού dashboard

Η Εικόνα 4.13 παρουσιάζει τη συνοπτική περιοχή του προσωπικού dashboard. Εμφανίζεται προσωποποιημένο μήνυμα καλωσορίσματος και βασικοί αριθμητικοί δείκτες, όπως πόσες ψηφοφορίες έχει δημιουργήσει ο χρήστης, πόσες είναι ενεργές, πόσες έχουν ολοκληρωθεί και σε πόσες έχει ψηφίσει. Με αυτόν τον τρόπο ο χρήστης αποκτά άμεσα συνολική εικόνα της δραστηριότητάς του.



Εικόνα 4.14: Αναλυτική προβολή του dashboard χρήστη

Στην Εικόνα 4.14 φαίνεται η πιο αναλυτική μορφή του dashboard, όπου οι συνοπτικοί δείκτες συνδυάζονται με λίστες πρόσφατης δραστηριότητας. Ο χρήστης μπορεί να δει πρόσφατες ψηφοφορίες που δημιούργησε, να ανοίξει την αντίστοιχη σελίδα και να ενημερωθεί για συναντήσεις στις οποίες έχει ήδη συμμετάσχει. Η οθόνη λειτουργεί ως κεντρικό σημείο πλοήγησης στις προσωπικές εργασίες του χρήστη.



Εικόνα 4.15: Σελίδα διαχείρισης ψηφοφοριών που δημιούργησε ο χρήστης

Στην Εικόνα 4.15 αποτυπώνεται η σελίδα με τις ψηφοφορίες που έχει δημιουργήσει ο συνδεδεμένος χρήστης. Στο επάνω μέρος υπάρχουν πεδίο αναζήτησης και φίλτρο κατάστασης, ενώ κάθε ψηφοφορία εμφανίζεται σε ξεχωριστή κάρτα με βασικά στοιχεία και διαθέσιμες ενέργειες. Η συγκεκριμένη οργάνωση διευκολύνει τη διαχείριση όταν ο αριθμός των meetings αυξάνεται.

ΝΕΟΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ

Δημιουργία ψηφοφορίας

Τίτλος, έναρξη/λήξη ψηφοφορίας και τουλάχιστον μία επιλογή με ημερομηνία & ώρα είναι υποχρεωτικά.

Γενικά στοιχεία

Τίτλος *

Περιγραφή

Κατηγορία

Χωρίς κατηγορία

Tags

Τοποθεσία

Link τηλεδιάσκεψης

Σημειώσεις

Ορατότητα

Μόνο με link

Emails πρόσκλησης

Προθεσμία ψηφοφορίας

Έναρξη *

Λήξη *

Επιλογές

Σε κάθε επιλογή απαιτούνται Τίτλος και Ημερομηνία & ώρα.

+ Επιλογή

Επιλογή 1

×

Τίτλος *

Ημερομηνία & ώρα *

Σημείωση

Επιλογή 2

×

Τίτλος *

Ημερομηνία & ώρα *

Σημείωση

Επιλογή 3

×

Τίτλος *

Ημερομηνία & ώρα *

Σημείωση

Επιλογή 4

×

Τίτλος *

Ημερομηνία & ώρα *

Επανάληψη

Χωρίς επανάληψη

1

Δημιουργία

Εικόνα 4.16: Φόρμα δημιουργίας νέας ψηφοφορίας

Η Εικόνα 4.16 δείχνει τη φόρμα δημιουργίας νέας ψηφοφορίας. Στο πρώτο τμήμα καταχωρίζονται τα γενικά στοιχεία, όπως τίτλος, περιγραφή, κατηγορία, tags, τοποθεσία, σύνδεσμος τηλεδιάσκεψης, σημειώσεις, ορατότητα και emails προσκεκλημένων, ενώ ακολουθεί η προθεσμία ψηφοφορίας. Στη συνέχεια προστίθενται μία ή περισσότερες επιλογές με τίτλο, ημερομηνία και ώρα και προαιρετική σημείωση, καθώς και ρυθμίσεις επανάληψης πριν από την τελική δημιουργία.

Συνέλευση

Γενικά στοιχεία

Τίτλος *

Συνέλευση

Περιγραφή

Περιγραφή συνέλευσης

Κατηγορία

Συνέλευση

Tags

Τοποθεσία

Link τηλεδιάσκεψης

Σημειώσεις

Προθεσμία ψηφοφορίας

Εναρξη *

2026-08-29 12:00:00

Λήξη *

2026-08-30 15:00:00

Επιλογές

Τίτλος και ημερομηνία & ώρα υποχρεωτικά.

[+ Επιλογή](#)

Επιλογή 1

Τίτλος *

Τρίτη

Ημερομηνία & ώρα *

2026-09-01 12:00:00

Σημείωση

Επιλογή 2

Τίτλος *

Τετάρτη

Ημερομηνία & ώρα *

2026-09-02 12:00:00

Σημείωση

Αποθήκευση αλλαγών

Εικόνα 4.17: Φόρμα επεξεργασίας υπάρχουσας ψηφοφορίας

Στην Εικόνα 4.17 παρουσιάζεται η φόρμα επεξεργασίας μιας ήδη υπάρχουσας ψηφοφορίας. Τα αποθηκευμένα γενικά στοιχεία, η περίοδος ψηφοφορίας και οι προτεινόμενες ημερομηνίες εμφανίζονται προσυμπληρωμένα, ώστε ο ιδιοκτήτης να μπορεί να πραγματοποιήσει αλλαγές χωρίς να δημιουργήσει νέο meeting. Η ενέργεια ολοκληρώνεται με την αποθήκευση των αλλαγών.

The screenshot shows a web interface for a public voting session. On the left, the meeting details are displayed: 'Συνέλευση' (Meeting), created by 'Καραγιάννης Λεωνίδας', with a description 'Περιγραφή συνέλευσης'. The voting period is from 29/08/2026 12:00 to 30/08/2026 15:00. Below this, a blue bar indicates the user is logged in as 'Καραγιάννης Λεωνίδας' and that automatic saving is enabled. Two voting options are shown: 'Τρίτη' (Tuesday, 01/09/2026 12:00) with a '★ Προτεινόμενη' (Recommended) tag, and 'Τετάρτη' (Wednesday, 02/09/2026 12:00). Each option has 'Ναι' (Yes) and 'Όχι' (No) buttons and a progress bar showing 2 votes for Yes and 0 for No, with a score of 4 (100%) for Tuesday and -2 (0%) for Wednesday. On the right, there is a 'Κοινή χρήση' (Share) section with a URL and a QR code, and a 'Πρόταση συστήματος' (System suggestion) section for the 'Τρίτη' option, showing a score of 4 and 100% suitability, with buttons to download the .ics file and integrate with Google/Outlook.

Εικόνα 4.18: Δημόσια σελίδα ψηφοφορίας με QR Code και πρόταση συστήματος

Στην Εικόνα 4.18 φαίνεται η δημόσια σελίδα μιας ενεργής ψηφοφορίας. Οι προτεινόμενες ημερομηνίες εμφανίζονται με κουμπιά Ναι και Όχι, ενώ στη δεξιά πλευρά παρέχονται ο κοινόχρηστος σύνδεσμος, το QR Code και η πρόταση του συστήματος για την καταλληλότερη επιλογή. Επιπλέον, υπάρχουν επιλογές εξαγωγής ή μεταφοράς της συνάντησης σε ημερολόγιο, ώστε η τελική απόφαση να μπορεί να αξιοποιηθεί άμεσα.

ορατό

Διαχείριση

Συνέλευση

Δημοσιεύς: Καραγιάννης Λεωνίδας

Περιγραφή συνέλευσης

Ψηφοφορία: 29/08/2026 12:00 → 30/08/2026 15:00

Ψηφίζεις ως **Καραγιάννης Λεωνίδας**. Αποθήκευση αυτόματα.

Τρίτη

★ Προτεινόμενη

01/09/2026 12:00

Ναι

Όχι

2 φέγγα

3 Ναι

0 Όχι

Score 4 : 100%

Τετάρτη

02/09/2026 12:00

Ναι

Όχι

2 φέγγα

0 Ναι

2 Όχι

Score -2 : 0%

Κοινή χρήση

http://127.0.0.1:5000/m/1E2ORjNNV-οζ

Πρόταση συστήματος

Τρίτη

Score: 4 · Καταλληλότητα 100%

Λήψη .ics

Google / Outlook

Live αποτελέσματα

Τρίτη 2

Ναι (2)

- Καραγιάννης Λεωνίδας
- Παπαδόπουλος Νίκος

Όχι (0)

- Κανείς

Τετάρτη 2

Ναι (0)

- Κανείς

Όχι (2)

- Καραγιάννης Λεωνίδας
- Παπαδόπουλος Νίκος

Συζήτηση

Γράψε σχόλιο...

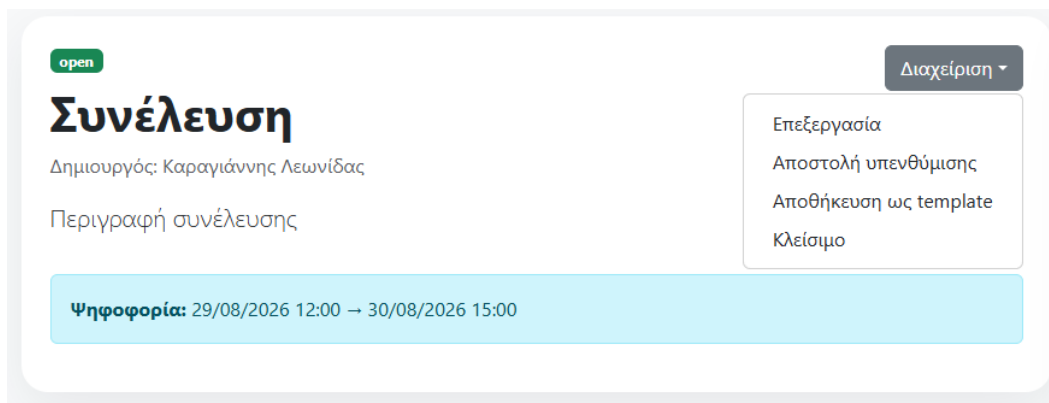
Δημοσίευση

Δεν υπάρχουν σχόλια.

Εικόνα 4.19: Δημόσια σελίδα με live αποτελέσματα και συζήτηση

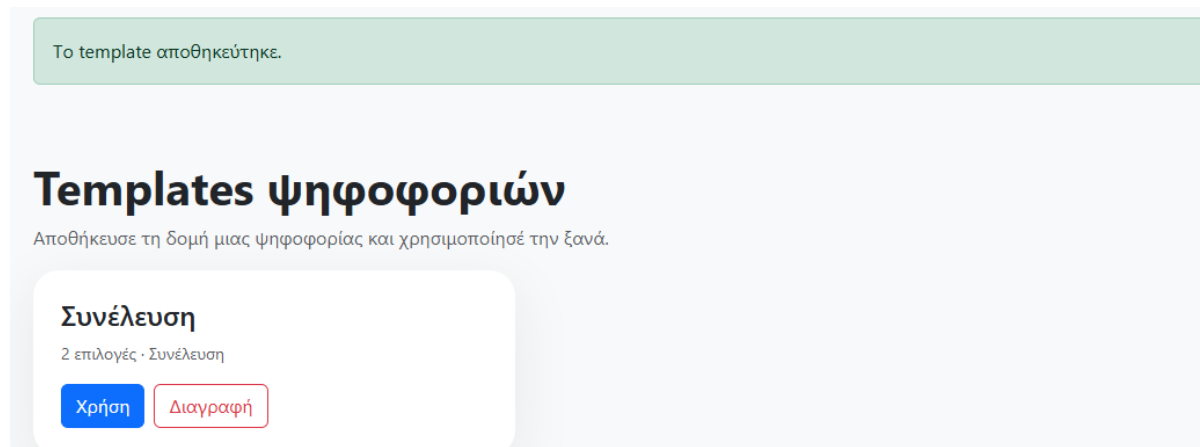
Η Εικόνα 4.19 απεικονίζει την εκτεταμένη δημόσια προβολή της ψηφοφορίας μετά την καταχώριση ψήφων. Εκτός από τις επιλογές και το recommendation, εμφανίζονται live αποτελέσματα με τους συμμετέχοντες που ψήφισαν Ναι ή Όχι για κάθε ημερομηνία. Στο κάτω μέρος υπάρχει επίσης περιοχή συζήτησης, όπου οι χρήστες μπορούν να προσθέσουν σχόλια σχετικά με τη συνάντηση.

39



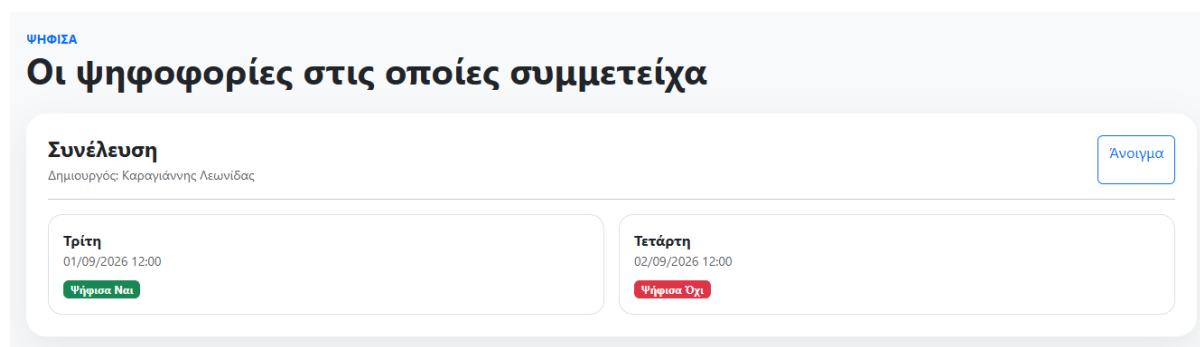
Εικόνα 4.20: Μενού διαχείρισης ψηφοφορίας από τον ιδιοκτήτη

Στην Εικόνα 4.20 παρουσιάζεται το μενού διαχείρισης που είναι διαθέσιμο στον ιδιοκτήτη της ψηφοφορίας. Από το συγκεκριμένο dropdown μπορούν να εκτελεστούν ενέργειες όπως επεξεργασία, αποστολή υπενθύμισης, αποθήκευση ως template και κλείσιμο της ψηφοφορίας. Έτσι οι βασικές λειτουργίες διαχείρισης συγκεντρώνονται σε ένα σαφές και περιορισμένο σημείο της διεπαφής.



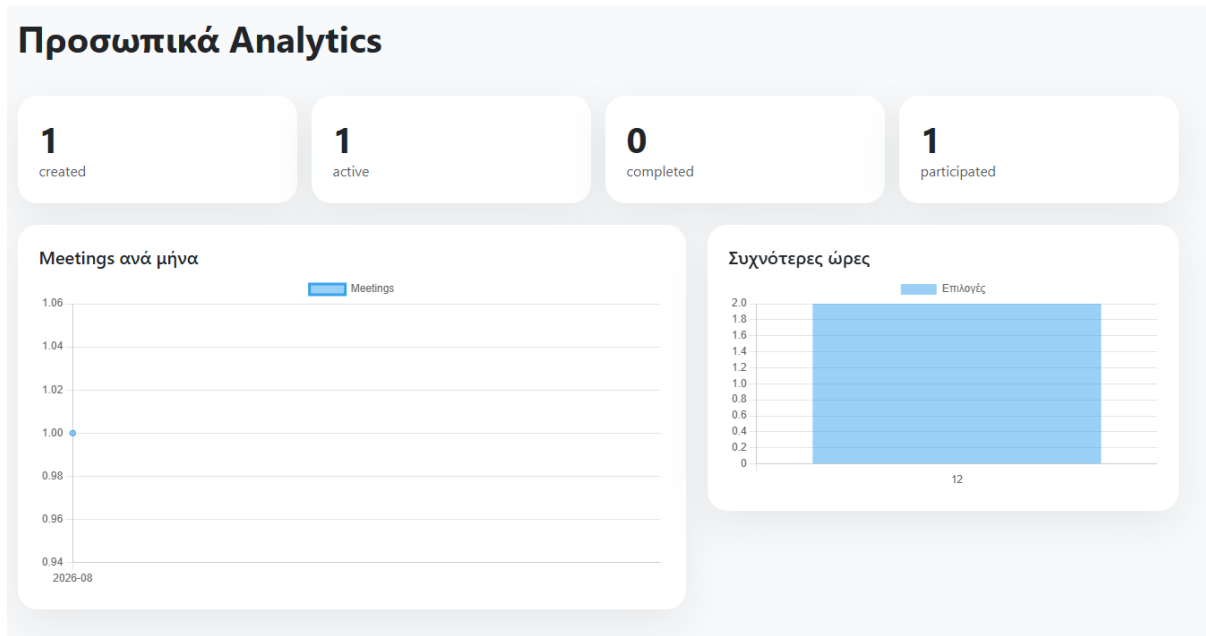
Εικόνα 4.21: Σελίδα αποθηκευμένων templates ψηφοφοριών

Μέσα από την Εικόνα 4.21 φαίνεται η σελίδα διαχείρισης των αποθηκευμένων templates. Κάθε πρότυπο εμφανίζεται με το όνομά του και παρέχει επιλογές για χρήση ή διαγραφή. Η λειτουργία επιτρέπει την επαναχρησιμοποίηση συχνών ρυθμίσεων και μειώνει τον χρόνο που απαιτείται για τη δημιουργία παρόμοιων ψηφοφοριών.



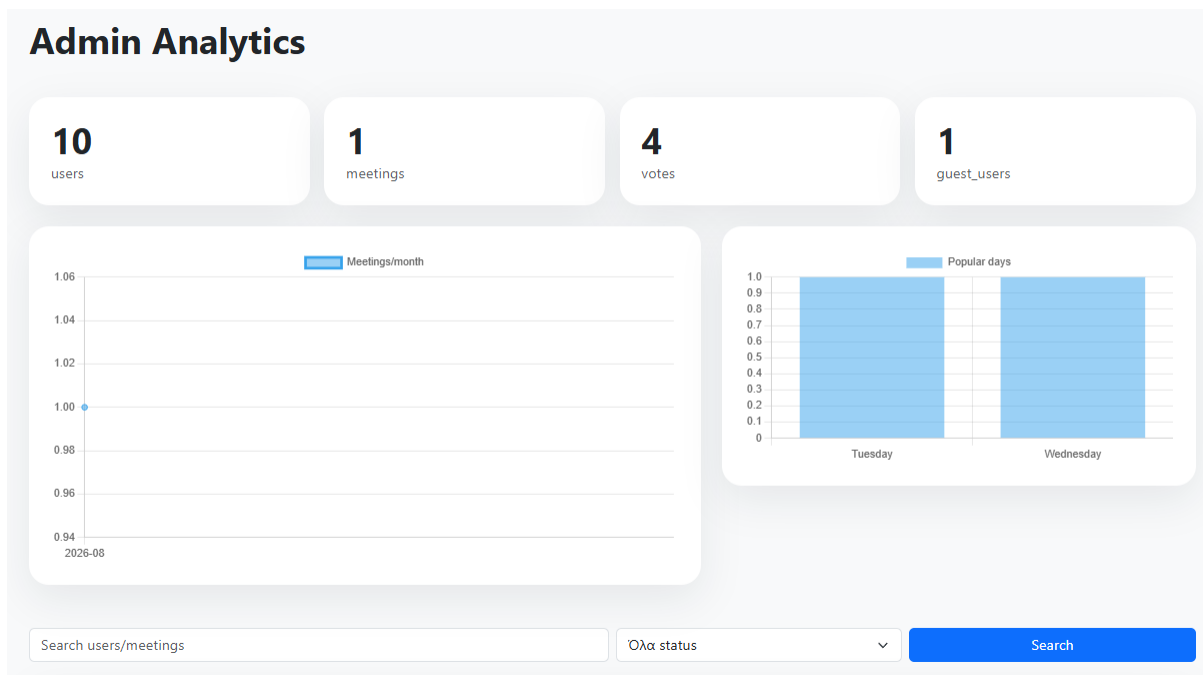
Εικόνα 4.22: Σελίδα ψηφοφοριών στις οποίες συμμετείχε ο χρήστης

Η Εικόνα 4.22 παρουσιάζει τη σελίδα με τις ψηφοφορίες στις οποίες έχει συμμετάσχει ο χρήστης. Για κάθε meeting εμφανίζονται οι διαθέσιμες επιλογές και η ψήφος που έχει καταχωριστεί, όπως Ναι ή Όχι. Με αυτόν τον τρόπο ο χρήστης μπορεί να ανατρέχει στο ιστορικό συμμετοχής του και να ανοίγει ξανά την αντίστοιχη ψηφοφορία.



Εικόνα 4.23: Σελίδα προσωπικών αναλυτικών στοιχείων

Στην Εικόνα 4.23 προβάλλονται τα προσωπικά analytics του συνδεδεμένου χρήστη. Στο επάνω μέρος εμφανίζονται συνοπτικά counts για δημιουργημένες, ενεργές, ολοκληρωμένες και συμμετοχικές ψηφοφορίες, ενώ τα γραφήματα παρουσιάζουν χρονικές κατανομές και συχνότερες ώρες. Η οπτικοποίηση βοηθά τον χρήστη να αντιλαμβάνεται γρήγορα τα βασικά πρότυπα της δραστηριότητάς του.



Εικόνα 4.24: Σελίδα διαχειριστικών αναλυτικών στοιχείων

Η Εικόνα 4.24 δείχνει το περιβάλλον Admin Analytics, το οποίο συγκεντρώνει στατιστικά για ολόκληρο το σύστημα. Παρουσιάζονται συνολικοί αριθμοί χρηστών, meetings, ψήφων και guest users, καθώς και γραφήματα δραστηριότητας. Στο κάτω μέρος υπάρχουν εργαλεία αναζήτησης και φιλτραρίσματος που υποστηρίζουν τη διαχειριστική εποπτεία.

Search users/meetings

Όλα status

Search

Χρήστες

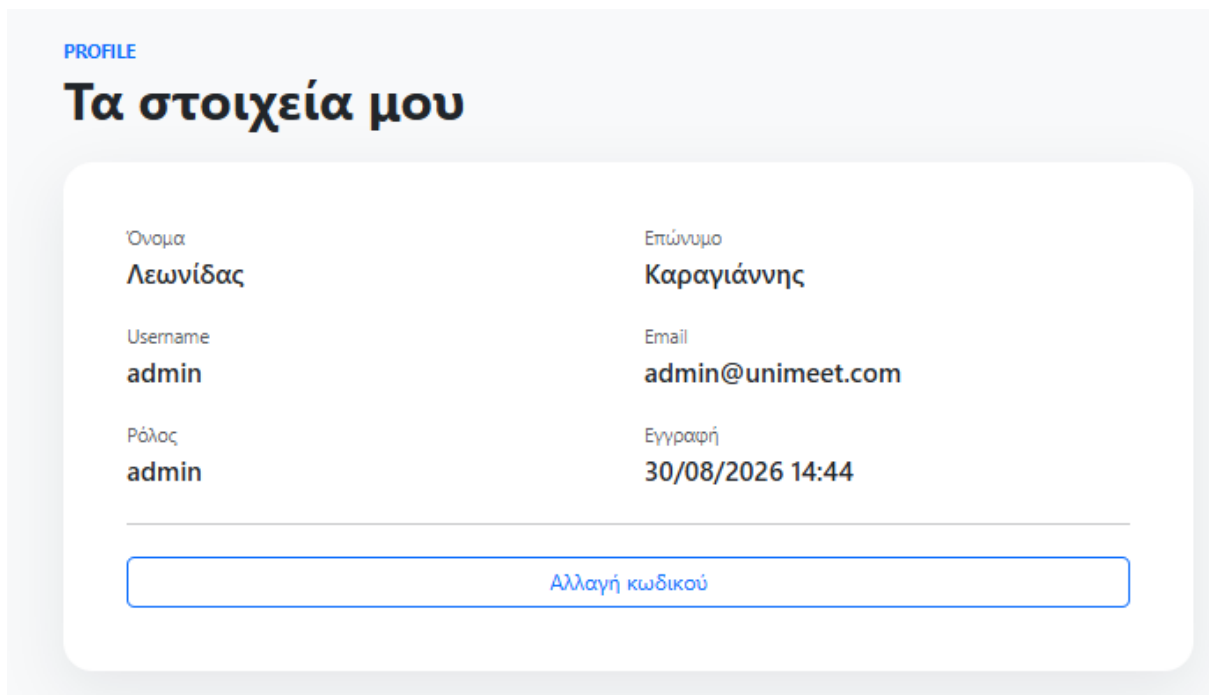
User	Email	Role	Meetings	Status
Παπαδόπουλος Νίκος pnik	pnik@meet.gr	user	0	Active
Καραγιάννης Λεωνίδας admin	admin@unimeet.com	admin	1	Active
Παπαδόπουλος Πέτρος petros	petros@unimeet.com	user	0	Active
Ιωάννου Μαρία maria	maria@unimeet.com	user	0	Active
Γεωργίου Νίκος nikos	nikos@unimeet.com	user	0	Active
Δημητρίου Ελένη eleni	eleni@unimeet.com	user	0	Active
Κωνσταντίνου Γιώργος giorgos	giorgos@unimeet.com	user	0	Active
Νικολάου Άννα anna	anna@unimeet.com	user	0	Active
Αντωνίου Κώστας kostas	kostas@unimeet.com	user	0	Active
Χριστοδούλου Σοφία sofia	sofia@unimeet.com	user	0	Active

Meetings

Τίτλος	Owner	Status	Options
Συνέλευση	Καραγιάννης Λεωνίδας	open	2

Εικόνα 4.25: Διαχειριστική προβολή χρηστών και συναντήσεων

Στην Εικόνα 4.25 φαίνεται η αναλυτική διαχειριστική προβολή των χρηστών και των meetings. Ο πίνακας χρηστών περιλαμβάνει username, email, ρόλο, αριθμό meetings και κατάσταση λογαριασμού, ενώ ξεχωριστή ενότητα εμφανίζει τις υπάρχουσες συναντήσεις με τον ιδιοκτήτη και την κατάστασή τους. Η οθόνη παρέχει στον administrator συγκεντρωτική εικόνα των βασικών οντοτήτων του συστήματος.



Εικόνα 4.26: Σελίδα προφίλ χρήστη

Η Εικόνα 4.26 απεικονίζει τη σελίδα προφίλ του χρήστη. Παρουσιάζονται τα βασικά στοιχεία λογαριασμού, όπως όνομα, επώνυμο, username, email, ρόλος και ημερομηνία εγγραφής. Από την ίδια σελίδα παρέχεται και η δυνατότητα αλλαγής κωδικού πρόσβασης.

Audit Log

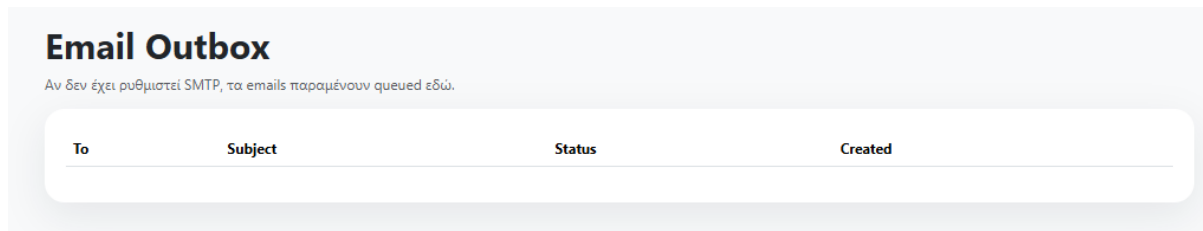
Αναζήτηση action/email...

Χρόνος	User	Action	Entity	Details
2026-09-05 18:14:47	Καραγιάννης Λεωνίδας admin@unimeet.com	template_saved	meeting_template #1	{"meeting_id": 1}
2026-08-30 14:52:26	Καραγιάννης Λεωνίδας admin@unimeet.com	meeting_edited	meeting #1	{"title": "Συνέλευση"}
2026-08-30 14:51:12	Παπαδόπουλος Νίκος pnik@meet.gr	vote_saved	meeting #1	{"option_id": 2, "choice": "no"}
2026-08-30 14:51:11	Παπαδόπουλος Νίκος pnik@meet.gr	vote_saved	meeting #1	{"option_id": 2, "choice": "yes"}
2026-08-30 14:50:23	Παπαδόπουλος Νίκος pnik@meet.gr	vote_saved	meeting #1	{"option_id": 2, "choice": "no"}
2026-08-30 14:50:18	Παπαδόπουλος Νίκος pnik@meet.gr	vote_saved	meeting #1	{"option_id": 1, "choice": "yes"}
2026-08-30 14:49:43	Καραγιάννης Λεωνίδας admin@unimeet.com	vote_saved	meeting #1	{"option_id": 2, "choice": "no"}
2026-08-30 14:49:37	Καραγιάννης Λεωνίδας admin@unimeet.com	vote_saved	meeting #1	{"option_id": 1, "choice": "yes"}
2026-08-30 14:49:31	Καραγιάννης Λεωνίδας admin@unimeet.com	meeting_edited	meeting #1	{"title": "Συνέλευση"}
2026-08-30 14:49:09	Καραγιάννης Λεωνίδας admin@unimeet.com	meeting_created	meeting #1	{"title": "aa"}

Εικόνα 4.27: Σελίδα καταγραφής ενεργειών Audit Log

Στην Εικόνα 4.27 παρουσιάζεται το Audit Log της εφαρμογής. Κάθε εγγραφή περιλαμβάνει χρόνο, χρήστη, ενέργεια, σχετική οντότητα και πρόσθετες λεπτομέρειες, επιτρέποντας την παρακολούθηση

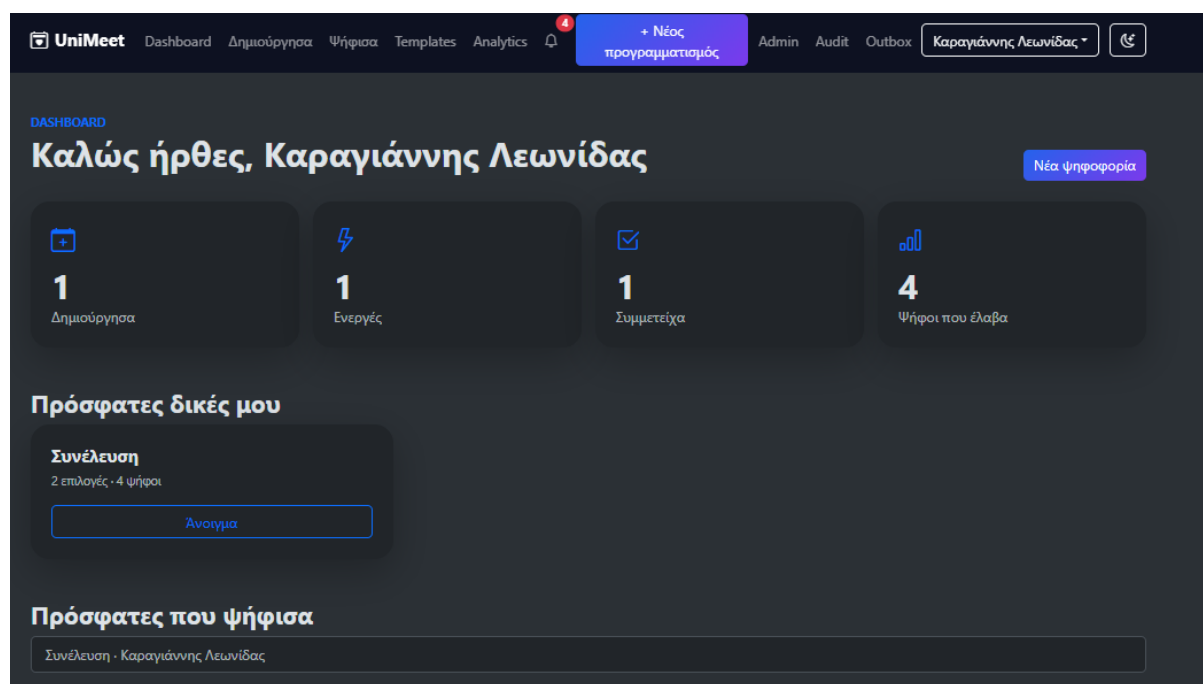
γεγονότων όπως δημιουργία ή επεξεργασία meeting και αποθήκευση ψήφου. Η λειτουργία ενισχύει την ιχνηλασιμότητα και προσφέρει στον διαχειριστή ιστορικό σημαντικών ενεργειών.



Αν δεν έχει ρυθμιστεί SMTP, τα emails παραμένουν queued εδώ.			
To	Subject	Status	Created

Εικόνα 4.28: Σελίδα Email Outbox

Η Εικόνα 4.28 δείχνει τη σελίδα Email Outbox, στην οποία συγκεντρώνονται τα εξερχόμενα μηνύματα όταν χρησιμοποιείται ο αντίστοιχος μηχανισμός της εφαρμογής. Ο πίνακας προβλέπει στοιχεία όπως παραλήπτης, θέμα, κατάσταση και χρόνος δημιουργίας. Στο συγκεκριμένο στιγμιότυπο δεν υπάρχουν καταχωρισμένα μηνύματα, γεγονός που αποτυπώνει την κενή κατάσταση της ουράς.



Εικόνα 4.29: Dashboard χρήστη σε σκοτεινή λειτουργία

Τέλος, στην Εικόνα 4.29 παρουσιάζεται το dashboard του UniMeet με ενεργοποιημένο το dark mode. Η δομή και οι λειτουργίες παραμένουν ίδιες με τη φωτεινή έκδοση, αλλά εφαρμόζεται σκοτεινή χρωματική παλέτα σε navigation, κάρτες και φόντο. Η επιλογή αυτή επιτρέπει στον χρήστη να προσαρμόξει την εμφάνιση της εφαρμογής στις προσωπικές του προτιμήσεις ή στις συνθήκες φωτισμού.

Κεφάλαιο 5ο: Συμπεράσματα και μελλοντικές βελτιώσεις

Η εργασία είχε ως αντικείμενο τη μελέτη και ανάπτυξη διαδικτυακής πλατφόρμας για αποτελεσματικό προγραμματισμό συναντήσεων μέσω ψηφοφορίας.

5.1 Αποτίμηση της εργασίας

Η ανάπτυξη οδήγησε σε πλήρως λειτουργική web εφαρμογή στην οποία υλοποιήθηκε ο κύκλος ζωής μιας scheduling poll. Η τελική έκδοση επεκτάθηκε με accounts, roles, guest auto-registration, notifications, reminders, comments, templates, categories, tags, recurrence, QR, iCalendar, calendar links, analytics, audit, search/filter/pagination.

Κατά την ανάπτυξη έγινε προσπάθεια οι διαφορετικές λειτουργίες να συνεργάζονται μεταξύ τους και να αποτελούν μέρος της ίδιας διαδικασίας χρήσης. Η βάση, το Flask backend, τα templates και η JavaScript συνεργάζονται ώστε ο χρήστης να μεταβαίνει από τη δημιουργία στη συμμετοχή και στη λήψη απόφασης χωρίς εξωτερική επεξεργασία.

Η τελική εφαρμογή καλύπτει τον βασικό στόχο που τέθηκε στην αρχή της εργασίας, καθώς επιτρέπει τη δημιουργία ψηφοφορίας και τη συγκέντρωση της διαθεσιμότητας όλων των συμμετεχόντων σε ένα σημείο. Η εύκολη συμμετοχή καλύπτεται με registered και guest flows, η live ενημέρωση με Fetch polling και η υποστήριξη απόφασης με score/suitability. Οι notifications, reminders, QR και calendar export καλύπτουν τη συνέχεια μετά τη συλλογή ψήφων.

5.1.1 Πλεονεκτήματα της πλατφόρμας UniMeet

- Απλή βασική ροή και άμεση αποθήκευση Ναι/Όχι.
- Συμμετοχή χωρίς προηγούμενη χειροκίνητη εγγραφή.
- Ζωντανή παρουσίαση counts και ψηφοφόρων.
- Ενσωμάτωση templates, tags, comments και analytics.
- QR Code και ανοικτό iCalendar για διαλειτουργικότητα.
- Έλεγχος ιδιοκτήτη και ξεχωριστό admin περιβάλλον.

5.1.2 Περιορισμοί της παρούσας υλοποίησης

Η εφαρμογή αναπτύχθηκε ως ακαδημαϊκό prototype. Για δημόσια παραγωγική εγκατάσταση απαιτούνται production WSGI server, reverse proxy, HTTPS, ασφαλές configuration και backups. Το live update βασίζεται σε polling, το calendar integration δεν είναι αμφίδρομο και οι email λειτουργίες θα ωφελούνταν από background queue/retries. Επιπλέον χρειάζεται περαιτέρω security hardening.

5.2 Προτεινόμενες μελλοντικές βελτιώσεις

5.2.1 Σύνδεση με ιδρυματικούς λογαριασμούς

SSO με OAuth 2.0/OpenID Connect ή ιδρυματική υπηρεσία θα περιορίζε ξεχωριστούς κωδικούς και θα επέτρεπε ασφαλέστερη αντιστοίχιση ακαδημαϊκής ταυτότητας και ρόλων.

5.2.2 Πλήρης συγχρονισμός με ημερολόγια

Με APIs Google/Microsoft, το σύστημα θα μπορούσε, με συγκατάθεση, να ανακτά busy intervals και να προτείνει αυτόματα κοινή διαθεσιμότητα, καθώς και να δημιουργεί συμβάντα μετά την οριστικοποίηση.

5.2.3 WebSockets ή Server-Sent Events

Σε μελλοντική έκδοση το polling θα μπορούσε να αντικατασταθεί από WebSockets ή Server-Sent Events, ώστε ο server να στέλνει νέα δεδομένα μόνο όταν υπάρχει κάποια αλλαγή.

5.2.4 Εφαρμογή για κινητές συσκευές και PWA

Μια PWA έκδοση με manifest/service worker και push notifications θα προσέφερε πιο εφαρμογοκεντρική εμπειρία σε κινητά.

5.2.5 Βελτίωση του αλγορίθμου πρότασης

Ο recommendation μηχανισμός θα μπορούσε να λάβει βάρη για υποχρεωτικούς συμμετέχοντες, time zones και κανόνες καταλληλότητας, διατηρώντας όμως επεξηγήσιμη τη λογική της πρότασης.

Μέσα από την ανάπτυξη της εφαρμογής φάνηκε ότι ένα σύστημα προγραμματισμού συναντήσεων μπορεί να υλοποιηθεί με σχετικά ελαφρύ τεχνολογικό σύνολο και να καλύπτει ολόκληρη τη διαδικασία, από την πρόσκληση μέχρι την οργάνωση της τελικής συνάντησης. Το UniMeet μετατρέπει διάσπαρτες προτιμήσεις σε δομημένα δεδομένα και συνδυάζει public participation, auto-registration, live results, recommendation, analytics, notifications, QR και calendar integration. Οι βασικοί περιορισμοί της παρούσας έκδοσης σχετίζονται με την παραγωγική εγκατάσταση, την πληρέστερη σύνδεση με εξωτερικές υπηρεσίες και την ενίσχυση της ασφάλειας. Τα σημεία αυτά μπορούν να αντιμετωπιστούν σε επόμενη έκδοση του συστήματος. Διευκρινίζεται ότι για τον συντακτικό και λεξιλογικό έλεγχο της εργασίας χρησιμοποιήθηκε πλατφόρμα τεχνητής νοημοσύνης.

ΒΙΒΛΙΟΓΡΑΦΙΑ

- [1] Brown, K., Mathur, H., Narayan, O. (2024). “Scheduling meetings: are the odds in your favor?”, European Physical Journal B, 97, 120. <https://link.springer.com/article/10.1140/epjb/s10051-024-00742-z>
- [2] Bredereck, R., Chen, J., Niedermeier, R., Obraztsova, S., Talmon, N. (2017). “Teams in Online Scheduling Polls: Game-Theoretic Aspects”, AAAI. <https://doi.org/10.1609/aaai.v31i1.10589>
- [3] Song, J., Ashktorab, Z., Malone, T. W. (2025). “Togedule: Scheduling Meetings with Large Language Models and Adaptive Representations of Group Availability”, PACM HCI.
- [4] Klensin, J. (2008). RFC 5321: Simple Mail Transfer Protocol. RFC Editor. <https://www.rfc-editor.org/info/rfc5321/>
- [5] Desruisseaux, B. (Ed.) (2009). RFC 5545: Internet Calendaring and Scheduling Core Object Specification. <https://www.rfc-editor.org/info/rfc5545/>
- [6] ISO/IEC 18004:2024. QR code bar code symbology specification. <https://www.iso.org/standard/83389.html>
- [7] Doodle. Group Poll. <https://doodle.com/en/product/polls/>
- [8] Doodle Help Center. Participation in Group Polls. <https://help.doodle.com/>
- [9] WhenAvailable. Free Group Scheduling Tool. <https://whenavailable.com/>
- [10] WhenAvailable. Pricing. <https://whenavailable.com/premium>
- [11] Rallly. Free Group Meeting Scheduling Tool. <https://rallly.co/>
- [12] Rallly. Pricing. <https://landing.rallly.co/pricing>
- [13] Python Software Foundation. Python Documentation. <https://docs.python.org/3/>
- [14] Pallets Projects. Flask Documentation - Quickstart. <https://flask.palletsprojects.com/en/stable/quickstart/>
- [15] Pallets Projects. Jinja Documentation. <https://jinja.palletsprojects.com/>
- [16] MariaDB. MariaDB Server Documentation. <https://mariadb.com/docs/server/>
- [17] PyMySQL Documentation. User Guide. <https://pymysql.readthedocs.io/en/latest/user/>
- [18] Bootstrap. Bootstrap 5.3 Documentation. <https://getbootstrap.com/docs/5.3/>
- [19] WHATWG. HTML Living Standard. <https://html.spec.whatwg.org/>
- [20] MDN Web Docs. CSS. <https://developer.mozilla.org/en-US/docs/Web/CSS>
- [21] MDN Web Docs. Fetch API. https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API
- [22] Flatpickr Documentation. <https://flatpickr.js.org/>
- [23] Chart.js. Getting Started. <https://www.chartjs.org/docs/latest/getting-started/>
- [24] Python Software Foundation. smtplib. <https://docs.python.org/3/library/smtplib.html>
- [25] PyPI. qrcode. <https://pypi.org/project/qrcode/>
- [26] Pallets Projects. Werkzeug Security Helpers. <https://werkzeug.palletsprojects.com/>
- [27] UniMeet v6.3. app.py - τελική έκδοση Flask backend.
- [28] UniMeet v6.3. database.sql - σχεσιακό σχήμα MariaDB.
- [29] UniMeet v6.3. templates/ - Jinja2/HTML templates.

ΠΑΡΑΡΤΗΜΑ

Το παράρτημα συγκεντρώνει επιλεγμένο τεχνικό υλικό από την τελική έκδοση UniMeet v6.3. Στόχος δεν είναι η αναπαραγωγή ολόκληρου του πηγαίου κώδικα, αλλά η παρουσίαση αντιπροσωπευτικών αποσπασμάτων που τεκμηριώνουν βασικές λειτουργίες, τη δομή της βάσης και τη διαδικασία εγκατάστασης.

A.1 Δομή του έργου

```
unimeet_v6_3_final/  
├─ app.py  
├─ database.sql  
├─ requirements.txt  
├─ README.md  
└─ templates/  
    ├─ base.html  
    ├─ welcome.html  
    ├─ login.html  
    ├─ register.html  
    ├─ dashboard.html  
    ├─ new_meeting.html  
    ├─ edit_meeting.html  
    ├─ public_meeting.html  
    ├─ profile.html  
    ├─ notifications.html  
    ├─ analytics.html  
    ├─ admin.html  
    ├─ audit.html  
    └─ outbox.html
```

A.2 Εξαρτήσεις και εκτέλεση

```
Flask==3.1.2  
PyMySQL==1.1.2  
Werkzeug==3.1.3  
qrcode[pil]==8.2
```

Η εφαρμογή αναπτύχθηκε για Windows με XAMPP/MariaDB. Η βασική διαδικασία περιλαμβάνει εισαγωγή database.sql, δημιουργία virtual environment, pip install -r requirements.txt και εκκίνηση με py app.py.

A.3 Μηχανισμός πρότασης καλύτερης επιλογής

```
def recommendation_for_options(options):  
    ranked = []  
    for o in options:  
        yes = int(o.get("yes_count") or 0)  
        no = int(o.get("no_count") or 0)  
        total = yes + no  
        score = yes * 2 - no  
        suitability = round((yes / total) * 100, 1) if total else 0  
        item = dict(o)  
        item["score"] = score  
        item["suitability"] = suitability  
        ranked.append(item)  
    ranked.sort(key=lambda x: (x["score"], x["suitability"], int(x.get("yes_count") or 0)), reverse=True)  
    return ranked
```

A.4 Δημιουργία meeting και validation

```
@app.route("/meeting/new", methods=["GET", "POST"])  
@login_required  
def new_meeting():  
    # συλλογή φόρμας, validation, public token  
    public_token = secrets.token_urlsafe(18)  
    # INSERT meeting και meeting_options
```

```
# tags, invites, audit και recurrence
...
```

A.5 Δημόσια ψήφος και αυτόματη εγγραφή επισκέπτη

```
@app.route("/m/<token>/vote", methods=["POST"])
def public_vote(token):
    # έλεγχος open/deadline και valid option
    # connected user -> session user_id
    # guest -> first_name, last_name, email
    # νέο email -> τυχαίο 4ψήφιο PIN + auto login
    # existing vote -> UPDATE, διαφορετικά INSERT
    ...
```

A.6 Live αποτελέσματα και API

```
@app.route("/api/m/<token>/results")
def api_meeting_results(token):
    # load options/voters
    # yes/no counts, score, suitability, rank
    return {"ok": True, "options": grouped}
```

A.7 Αυτόματη αποθήκευση ψήφου στο frontend

```
async function saveVote(id, choice) {
    const d = new FormData();
    d.append('option_id', id);
    d.append('choice', choice);
    const r = await fetch(VOTE_URL, { method: 'POST', body: d });
    if (r.ok) refreshResults();
}

async function refreshResults() {
    const r = await fetch(RESULTS_URL, { cache: 'no-store' });
    // ενημέρωση counts, score και voters
}
setInterval(refreshResults, 3000);
```

A.8 Βασικά routes της εφαρμογής

Endpoint	Λειτουργία
/register	Εγγραφή
/login	Σύνδεση
/dashboard	Dashboard
/meeting/new	Δημιουργία
/meeting/<id>/edit	Επεξεργασία
/m/<token>	Public meeting
/m/<token>/vote	Αποθήκευση ψήφου
/api/m/<token>/results	Live JSON results
/meeting/<id>/close	Κλείσιμο
/meeting/<id>/reopen	Επανάνοιγμα
/notifications	Ειδοποιήσεις
/m/<token>/qr.png	QR Code
/m/<token>/calendar.ics	iCalendar
/meeting/<id>/remind	Reminder
/templates	Templates
/analytics	Analytics

/admin	Admin
/audit	Audit log
/outbox	Email outbox

A.9 Ενδεικτικός πηγαίος κώδικας της εφαρμογής UniMeet

```
# """
```

```
UniMeet - Ενδεικτικός κώδικας παραρτήματος
```

```
=====
```

Το αρχείο αυτό αποτελεί συμπυκνωμένη και σχολιασμένη παρουσίαση της τελικής εφαρμογής UniMeet. Δεν αντικαθιστά το πλήρες app.py. Επιλέχθηκαν τα τμήματα που περιγράφουν καλύτερα την αρχιτεκτονική και τις βασικές λειτουργίες:

1. Ρύθμιση Flask, MariaDB και SMTP
2. Βοηθητικές συναρτήσεις βάσης δεδομένων
3. Authentication / Authorization
4. Δημιουργία και επεξεργασία ψηφοφορίας
5. Δημόσια σελίδα και guest auto-registration
6. Αυτόματη αποθήκευση ψήφου
7. Live αποτελέσματα και recommendation score
8. Notifications, comments και reminders
9. QR Code και iCalendar integration
10. Templates, analytics και admin λειτουργίες

Ο κώδικας βασίζεται στην τελική έκδοση UniMeet v6.3.

```
"""
```

```
# -----
```

```
from flask import Flask, render_template, request, redirect, url_for, flash, session, abort, make_response, send_file
import pymysql

from werkzeug.security import generate_password_hash, check_password_hash

from functools import wraps

import secrets
import io
import json
import math
import smtplib
from email.message import EmailMessage
```

```

from urllib.parse import urlencode, quote
from datetime import datetime, timedelta
import qrcode

app = Flask(__name__)
app.secret_key = "CHANGE-ME-UNIMEET-SECRET-KEY"

SMTP_CONFIG = {
    "host": "",      # π.χ. smtp.gmail.com
    "port": 587,
    "username": "",
    "password": "",
    "from_email": "noreply@unimeet.local",
    "use_tls": True,
}

DB_CONFIG = {
    "host": "127.0.0.1",
    "user": "root",
    "password": "",
    "database": "unimeet_db",
    "charset": "utf8mb4",
    "cursorclass": pymysql.cursors.DictCursor,
    "autocommit": False,
}

def get_db():
    return pymysql.connect(**DB_CONFIG)

def query_one(sql, params=()):
    conn = get_db()
    try:
        with conn.cursor() as cur:
            cur.execute(sql, params)
            return cur.fetchone()
    finally:
        conn.close()

def query_all(sql, params=()):
    conn = get_db()
    try:
        with conn.cursor() as cur:
            cur.execute(sql, params)

```

```

        return cur.fetchall()
    finally:
        conn.close()

def execute(sql, params=()):
    conn = get_db()
    try:
        with conn.cursor() as cur:
            cur.execute(sql, params)
            last_id = cur.lastrowid
        conn.commit()
        return last_id
    except Exception:
        conn.rollback()
        raise
    finally:
        conn.close()

def audit(action, entity_type=None, entity_id=None, details=None, user_id=None):
    try:
        execute(
            """INSERT INTO audit_logs(user_id,action,entity_type,entity_id,details,ip_address)
            VALUES(%s,%s,%s,%s,%s,%s)""",
            (
                user_id if user_id is not None else session.get("user_id"),
                action, entity_type, entity_id,
                json.dumps(details or {}, ensure_ascii=False),
                request.remote_addr if request else None,
            ),
        )
    except Exception:
        pass

def notify(user_id, title, message, link=None, kind="info"):
    if not user_id:
        return
    try:
        execute(
            """INSERT INTO notifications(user_id,title,message,link,kind)
            VALUES(%s,%s,%s,%s,%s)""",
            (user_id, title, message, link, kind),
        )
    except Exception:
        pass

```

```

def queue_email(to_email, subject, body):
    status = "queued"
    error_text = None
    try:
        if SMTP_CONFIG["host"] and SMTP_CONFIG["username"]:
            msg = EmailMessage()
            msg["From"] = SMTP_CONFIG["from_email"]
            msg["To"] = to_email
            msg["Subject"] = subject
            msg.set_content(body)
            with smtplib.SMTP(SMTP_CONFIG["host"], SMTP_CONFIG["port"], timeout=10) as smtp:
                if SMTP_CONFIG.get("use_tls"):
                    smtp.starttls()
                smtp.login(SMTP_CONFIG["username"], SMTP_CONFIG["password"])
                smtp.send_message(msg)
            status = "sent"
    except Exception as exc:
        status = "failed"
        error_text = str(exc)[:500]
    try:
        execute(
            """INSERT INTO email_outbox(to_email,subject,body,status,error_text,sent_at)
            VALUES(%s,%s,%s,%s,%s,IF(%s='sent',NOW(),NULL))""",
            (to_email, subject, body, status, error_text, status),
        )
    except Exception:
        pass
    return status

def meeting_participant_user_ids(meeting_id):
    rows = query_all(
        """SELECT DISTINCT v.user_id FROM votes v
        JOIN meeting_options o ON o.id=v.option_id
        WHERE o.meeting_id=%s AND v.user_id IS NOT NULL""",
        (meeting_id,),
    )
    return [r["user_id"] for r in rows]

def meeting_participant_emails(meeting_id):
    rows = query_all(
        """SELECT DISTINCT v.voter_email email FROM votes v
        JOIN meeting_options o ON o.id=v.option_id
        WHERE o.meeting_id=%s""",
        (meeting_id,),
    )

```

```

)
return [r["email"] for r in rows if r["email"]]

def normalize_datetime_value(value):
    """Normalize values from Flatpickr or native datetime-local inputs for MariaDB."""
    value = (value or "").strip()
    if not value:
        return ""
    value = value.replace("T", " ")
    if len(value) == 16:
        value += ":00"
    try:
        return datetime.strptime(value, "%Y-%m-%d %H:%M:%S").strftime("%Y-%m-%d %H:%M:%S")
    except ValueError:
        return ""

def recommendation_for_options(options):
    ranked = []
    for o in options:
        yes = int(o.get("yes_count") or 0)
        no = int(o.get("no_count") or 0)
        total = yes + no
        score = yes * 2 - no
        suitability = round((yes / total) * 100, 1) if total else 0
        item = dict(o)
        item["score"] = score
        item["suitability"] = suitability
        ranked.append(item)
    ranked.sort(key=lambda x: (x["score"], x["suitability"], int(x.get("yes_count") or 0)), reverse=True)
    return ranked

def paginated(query_sql, params, page, per_page=8):
    page = max(int(page or 1), 1)
    offset = (page - 1) * per_page
    rows = query_all(query_sql + " LIMIT %s OFFSET %s", tuple(params) + (per_page, offset))
    return rows, page

# -----

def clone_weekly_meetings(source_meeting_id, repeat_count):
    if repeat_count <= 1:
        return
    source = query_one("SELECT * FROM meetings WHERE id=%s", (source_meeting_id,))
    options = query_all("SELECT * FROM meeting_options WHERE meeting_id=%s ORDER BY sort_order,id",
        (source_meeting_id,))

```

```
tags = query_all("SELECT tag FROM meeting_tags WHERE meeting_id=%s", (source_meeting_id,))
invites = query_all("SELECT email FROM meeting_invites WHERE meeting_id=%s", (source_meeting_id,))
for idx in range(1, min(repeat_count, 8)):
    delta = timedelta(days=7*idx)
    token = secrets.token_urlsafesize(18)
    new_id = execute(
        """INSERT INTO meetings
            (owner_id,title,description,location,meeting_link,notes,visibility,
             voting_starts_at,voting_ends_at,public_token,status,category_id,
             recurrence_type,recurrence_count,parent_meeting_id,recurrence_index)
            VALUES(%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,'open',%s,'weekly',%s,%s,%s)""",
        (
            source["owner_id"], source["title"], source["description"], source["location"],
            source["meeting_link"], source["notes"], source["visibility"],
            source["voting_starts_at"] + delta, source["voting_ends_at"] + delta,
            token, source.get("category_id"), repeat_count, source_meeting_id, idx
        ),
    )
for o in options:
    execute(
        """INSERT INTO meeting_options(meeting_id,title,starts_at,ends_at,note,sort_order)
            VALUES(%s,%s,%s,%s,%s,%s)""",
        (
            new_id, o["title"],
            o["starts_at"] + delta if o["starts_at"] else None,
            o["ends_at"] + delta if o["ends_at"] else None,
            o["note"], o["sort_order"],
        ),
    )
for t in tags:
    execute("INSERT INTO meeting_tags(meeting_id,tag) VALUES(%s,%s)", (new_id, t["tag"]))
for inv in invites:
    execute("INSERT IGNORE INTO meeting_invites(meeting_id,email) VALUES(%s,%s)", (new_id,
inv["email"])))

def login_required(view):
    @wraps(view)
    def wrapped(*args, **kwargs):
        if "user_id" not in session:
            flash("Πρέπει να συνδεθείς για αυτή την ενέργεια.", "warning")
            return redirect(url_for("login"))
        return view(*args, **kwargs)
    return wrapped
```

```

def admin_required(view):
    @wraps(view)
    def wrapped(*args, **kwargs):
        if session.get("role") != "admin":
            abort(403)
        return view(*args, **kwargs)
    return wrapped

@app.context_processor
def inject_user():
    current_name = None
    if "user_id" in session:
        current_name = f'{session.get("last_name','')} {session.get("first_name','")}'.strip()
    return {
        "logged_in": "user_id" in session,
        "current_user_id": session.get("user_id"),
        "current_name": current_name,
        "current_first_name": session.get("first_name"),
        "current_last_name": session.get("last_name"),
        "current_role": session.get("role"),
        "current_email": session.get("email"),
        "current_username": session.get("username"),
        "unread_notifications": (
            query_one("SELECT COUNT(*) c FROM notifications WHERE user_id=%s AND is_read=0",
(session.get("user_id"),))["c"]
            if session.get("user_id") else 0
        ),
    }

# -----

@app.route("/")
def index():
    recent = query_all(
        """SELECT m.id,m.title,m.public_token,m.created_at,CONCAT(u.last_name,' ',u.first_name) owner_name
        FROM meetings m JOIN users u ON u.id=m.owner_id
        WHERE m.visibility='public'
        ORDER BY m.created_at DESC LIMIT 6"""
    )
    return render_template("welcome.html", recent=recent)

@app.route("/register", methods=["GET", "POST"])
def register():
    if request.method == "POST":
        username = request.form.get("username", "").strip()

```

```

first_name = request.form.get("first_name", "").strip()
last_name = request.form.get("last_name", "").strip()
email = request.form.get("email", "").strip().lower()
password = request.form.get("password", "")

if not username or not first_name or not last_name or not email or len(password) < 4:
    flash("Συμπλήρωσε όλα τα πεδία. Ο κωδικός πρέπει να έχει τουλάχιστον 4 χαρακτήρες.", "danger")
    return render_template("register.html")

exists = query_one(
    "SELECT id FROM users WHERE username=%s OR email=%s",
    (username, email),
)
if exists:
    flash("Το username ή το email χρησιμοποιείται ήδη.", "warning")
    return render_template("register.html")

execute(
    """INSERT INTO users(username,email,first_name,last_name,password_hash,role,is_active)
    VALUES(%s,%s,%s,%s,%s,'user',1)""",
    (username, email, first_name, last_name, generate_password_hash(password)),
)
flash("Ο λογαριασμός δημιουργήθηκε.", "success")
return redirect(url_for("login"))

return render_template("register.html")

@app.route("/login", methods=["GET", "POST"])
def login():
    if request.method == "POST":
        identity = request.form.get("identity", "").strip().lower()
        password = request.form.get("password", "")
        user = query_one(
            """SELECT * FROM users
            WHERE (LOWER(email)=%s OR LOWER(username)=%s) AND is_active=1""",
            (identity, identity),
        )
        if user and check_password_hash(user["password_hash"], password):
            session.clear()
            session["user_id"] = user["id"]
            session["username"] = user["username"]
            session["first_name"] = user["first_name"]
            session["last_name"] = user["last_name"]
            session["email"] = user["email"]
            session["role"] = user["role"]

```

```

        flash("Καλώς ήρθες στο UniMeet.", "success")
        return redirect(url_for("dashboard"))
        flash("Λάθος username/email ή κωδικός.", "danger")
        return render_template("login.html")

@app.route("/logout")
def logout():
    session.clear()
    flash("Αποσυνδέθηκες.", "info")
    return redirect(url_for("index"))

@app.route("/forgot", methods=["GET", "POST"])
def forgot():
    if request.method == "POST":
        email = request.form.get("email", "").strip().lower()
        user = query_one("SELECT id,email FROM users WHERE LOWER(email)=%", (email,))
        if user:
            token = secrets.token_urlsafe(24)
            execute("DELETE FROM password_resets WHERE user_id=%s", (user["id"],))
            execute(
                """INSERT INTO password_resets(user_id,token,expires_at)
                VALUES(%s,%s,DATE_ADD(NOW(), INTERVAL 30 MINUTE))""",
                (user["id"], token),
            )
            reset_url = url_for("reset_password", token=token, _external=True)
            flash(f'Demo link επαναφοράς: {reset_url}', "info")
        else:
            flash("Αν το email υπάρχει, δημιουργήθηκε αίτημα επαναφοράς.", "info")
        return render_template("forgot.html")

@app.route("/reset/<token>", methods=["GET", "POST"])
def reset_password(token):
    reset = query_one(
        """SELECT pr.*,u.email
        FROM password_resets pr
        JOIN users u ON u.id=pr.user_id
        WHERE pr.token=%s AND pr.expires_at>NOW()""",
        (token,),
    )
    if not reset:
        flash("Ο σύνδεσμος επαναφοράς δεν είναι έγκυρος ή έχει λήξει.", "danger")
        return redirect(url_for("forgot"))

    if request.method == "POST":

```

```

password = request.form.get("password", "")
if len(password) < 4:
    flash("Ο κωδικός πρέπει να έχει τουλάχιστον 4 χαρακτήρες.", "danger")
else:
    execute(
        "UPDATE users SET password_hash=%s WHERE id=%s",
        (generate_password_hash(password), reset["user_id"]),
    )
    execute("DELETE FROM password_resets WHERE id=%s", (reset["id"],))
    flash("Ο κωδικός άλλαξε. Μπορείς να συνδεθείς.", "success")
    return redirect(url_for("login"))
return render_template("reset_password.html")

@app.route("/change-password", methods=["GET", "POST"])
@login_required
def change_password():
    if request.method == "POST":
        old_password = request.form.get("old_password", "")
        new_password = request.form.get("new_password", "")
        user = query_one("SELECT * FROM users WHERE id=%s", (session["user_id"],))

        if not check_password_hash(user["password_hash"], old_password):
            flash("Ο τρέχων κωδικός δεν είναι σωστός.", "danger")
        elif len(new_password) < 4:
            flash("Ο νέος κωδικός πρέπει να έχει τουλάχιστον 4 χαρακτήρες.", "danger")
        else:
            execute(
                """UPDATE users
                SET password_hash=%s,
                temporary_password_plain=NULL,
                temporary_password_visible_until=NULL
                WHERE id=%s""",
                (generate_password_hash(new_password), session["user_id"]),
            )
            flash("Ο κωδικός ενημερώθηκε.", "success")
            return redirect(url_for("dashboard"))
    return render_template("change_password.html")

@app.route("/dashboard")
@login_required
def dashboard():
    uid = session["user_id"]
    meetings = query_all(
        """SELECT m.*, (SELECT COUNT(*) FROM meeting_options o WHERE o.meeting_id=m.id) option_count,

```

```

        (SELECT COUNT(*) FROM votes v JOIN meeting_options o2 ON o2.id=v.option_id WHERE
o2.meeting_id=m.id) vote_count
        FROM meetings m WHERE m.owner_id=%s ORDER BY m.created_at DESC LIMIT 6""",
        (uid,),
    )
    participated = query_all(
        """SELECT DISTINCT m.id,m.title,m.public_token,m.status,CONCAT(u.last_name,' ',u.first_name) owner_name
        FROM meetings m JOIN meeting_options o ON o.meeting_id=m.id
        JOIN votes v ON v.option_id=o.id JOIN users u ON u.id=m.owner_id
        WHERE v.user_id=%s ORDER BY m.created_at DESC LIMIT 6""",
        (uid,),
    )
    stats = {
        "created": query_one("SELECT COUNT(*) c FROM meetings WHERE owner_id=%s", (uid,))["c"],
        "active": query_one("SELECT COUNT(*) c FROM meetings WHERE owner_id=%s AND status='open'",
(uid,))["c"],
        "participated": query_one("SELECT COUNT(DISTINCT o.meeting_id) c FROM votes v JOIN meeting_options o
ON o.id=v.option_id WHERE v.user_id=%s", (uid,))["c"],
        "votes_received": query_one("SELECT COUNT(*) c FROM votes v JOIN meeting_options o ON o.id=v.option_id
JOIN meetings m ON m.id=o.meeting_id WHERE m.owner_id=%s", (uid,))["c"],
    }
    return render_template("dashboard.html", meetings=meetings, participated=participated, stats=stats)

# -----

@app.route("/my/created")
@login_required
def my_created():
    q = request.args.get("q", "").strip()
    status = request.args.get("status", "").strip()
    page = request.args.get("page", 1, type=int)
    per_page = 9
    where = ["m.owner_id=%s"]
    params = [session["user_id"]]
    if q:
        where.append("(m.title LIKE %s OR m.description LIKE %s)")
        params.extend([f"%{q}%", f"%{q}%"])
    if status in ("open", "closed"):
        where.append("m.status=%s")
        params.append(status)
    total = query_one(f"SELECT COUNT(*) c FROM meetings m WHERE {' AND '.join(where)}", tuple(params))["c"]
    meetings = query_all(
        f"""SELECT m.*,c.name category_name,
        (SELECT COUNT(*) FROM meeting_options o WHERE o.meeting_id=m.id) option_count,

```

```

        (SELECT COUNT(*) FROM votes v JOIN meeting_options o2 ON o2.id=v.option_id WHERE
o2.meeting_id=m.id) vote_count
        FROM meetings m LEFT JOIN categories c ON c.id=m.category_id
        WHERE {' AND '.join(where)}
        ORDER BY m.created_at DESC LIMIT %s OFFSET %s""",
        tuple(params) + (per_page, (page-1)*per_page),
    )
    return render_template("my_created.html", meetings=meetings, q=q, status=status, page=page, pages=max(1,
math.ceil(total/per_page)))

@app.route("/my/voted")
@login_required
def my_voted():
    meetings = query_all(
        """SELECT DISTINCT m.*, CONCAT(u.last_name,',',u.first_name) owner_name
        FROM meetings m
        JOIN meeting_options o ON o.meeting_id=m.id
        JOIN votes v ON v.option_id=o.id
        JOIN users u ON u.id=m.owner_id
        WHERE v.user_id=%s
        ORDER BY m.created_at DESC""",
        (session["user_id"],),
    )

    vote_details = {}
    for meeting in meetings:
        rows = query_all(
            """SELECT o.id option_id,o.title,o.starts_at,o.ends_at,v.choice,v.updated_at
            FROM meeting_options o
            LEFT JOIN votes v
            ON v.option_id=o.id AND v.user_id=%s
            WHERE o.meeting_id=%s
            ORDER BY o.sort_order,o.id""",
            (session["user_id"], meeting["id"]),
        )
        vote_details[meeting["id"]] = rows

    return render_template("my_voted.html", meetings=meetings, vote_details=vote_details)

@app.route("/profile")
@login_required
def profile():
    user = query_one("SELECT * FROM users WHERE id=%s", (session["user_id"],))
    show_temp_password = False
    temp_password = None

```

```

if user and user.get("temporary_password_plain") and user.get("temporary_password_visible_until"):
    if datetime.now() <= user["temporary_password_visible_until"]:
        show_temp_password = True
        temp_password = user["temporary_password_plain"]
return render_template(
    "profile.html",
    user=user,
    show_temp_password=show_temp_password,
    temp_password=temp_password,
)

@app.route("/api/m/<token>/results")
def api_meeting_results(token):
    meeting, options, voters, tags, comments = load_public_meeting(token)
    ranked = recommendation_for_options(options)
    rank_map = {o["id"]: idx + 1 for idx, o in enumerate(ranked)}
    grouped = []
    for option in options:
        yes_voters = [{"name": v["voter_name"], "email": v["voter_email"]}
                       for v in voters if v["option_id"] == option["id"] and v["choice"] == "yes"]
        no_voters = [{"name": v["voter_name"], "email": v["voter_email"]}
                     for v in voters if v["option_id"] == option["id"] and v["choice"] == "no"]
        yes = int(option["yes_count"] or 0)
        no = int(option["no_count"] or 0)
        total = yes + no
        grouped.append({
            "id": option["id"], "title": option["title"],
            "yes_count": yes, "no_count": no, "total_count": total,
            "score": yes * 2 - no,
            "suitability": round(yes / total * 100, 1) if total else 0,
            "rank": rank_map.get(option["id"]),
            "recommended": rank_map.get(option["id"]) == 1 and total > 0,
            "yes_voters": yes_voters, "no_voters": no_voters,
        })
    return {"ok": True, "options": grouped}

# -----

@app.route("/meeting/new", methods=["GET", "POST"])
@login_required
def new_meeting():
    categories = query_all("SELECT * FROM categories ORDER BY name")
    templates = query_all("SELECT id,name FROM meeting_templates WHERE user_id=%s ORDER BY name",
(session["user_id"],))
    template_id = request.args.get("template_id", type=int)

```

```

template_data = None
template_options = []
if template_id:
    template_data = query_one("SELECT * FROM meeting_templates WHERE id=%s AND user_id=%s",
(template_id, session["user_id"]))
    if template_data:
        template_options = query_all("SELECT * FROM meeting_template_options WHERE template_id=%s ORDER
BY sort_order,id", (template_id,))

if request.method == "POST":
    title = request.form.get("title", "").strip()
    description = request.form.get("description", "").strip()
    location = request.form.get("location", "").strip()
    meeting_link = request.form.get("meeting_link", "").strip()
    notes = request.form.get("notes", "").strip()
    visibility = request.form.get("visibility", "link")
    category_id = request.form.get("category_id", type=int)
    tags_text = request.form.get("tags", "").strip()
    invite_emails = request.form.get("invite_emails", "").strip()
    recurrence_type = request.form.get("recurrence_type", "none")
    recurrence_count = request.form.get("recurrence_count", type=int) or 1
    voting_starts_at = normalize_datetime_value(request.form.get("voting_starts_at", ""))
    voting_ends_at = normalize_datetime_value(request.form.get("voting_ends_at", ""))

    option_titles = request.form.getlist("option_title[]")
    option_datetimes = request.form.getlist("option_datetime[]")
    option_notes = request.form.getlist("option_note[]")
    option_rows = []
    options = []
    max_len = max(len(option_titles), len(option_datetimes), len(option_notes), 0)
    for i in range(max_len):
        ot = option_titles[i].strip() if i < len(option_titles) else ""
        odt_raw = option_datetimes[i].strip() if i < len(option_datetimes) else ""
        odt = normalize_datetime_value(odt_raw)
        on = option_notes[i].strip() if i < len(option_notes) else ""
        option_rows.append((ot, odt_raw, on))
        if not ot and not odt and not on:
            continue
        if not ot:
            flash("Ο τίτλος είναι υποχρεωτικός σε κάθε επιλογή.", "danger")
            return render_template("new_meeting.html", categories=categories, templates=templates,
                                template_data=template_data, template_options=template_options,
                                form_data=request.form, option_rows=option_rows)
        if not odt:
            flash(f"Η ημερομηνία και ώρα είναι υποχρεωτικές στην επιλογή «{ot}».", "danger")

```

```

return render_template("new_meeting.html", categories=categories, templates=templates,
                        template_data=template_data, template_options=template_options,
                        form_data=request.form, option_rows=option_rows)
options.append((ot, odt, on))

def rerender():
    return render_template("new_meeting.html", categories=categories, templates=templates,
                           template_data=template_data, template_options=template_options,
                           form_data=request.form, option_rows=option_rows)

if not title:
    flash("Ο τίτλος του προγραμματισμού είναι υποχρεωτικός.", "danger")
    return rerender()

if not voting_starts_at or not voting_ends_at:
    flash("Η έναρξη και η λήξη της προθεσμίας ψηφοφορίας είναι υποχρεωτικές.", "danger")
    return rerender()

if voting_ends_at <= voting_starts_at:
    flash("Η λήξη της ψηφοφορίας πρέπει να είναι μετά την έναρξη.", "danger")
    return rerender()

if not options:
    flash("Χρειάζεται τουλάχιστον μία επιλογή με τίτλο, ημερομηνία και ώρα.", "danger")
    return rerender()

if recurrence_type not in ("none", "weekly"):
    recurrence_type = "none"

recurrence_count = max(1, min(recurrence_count, 8))

public_token = secrets.token_urlsafe(18)
meeting_id = execute(
    """INSERT INTO meetings
(owner_id,title,description,location,meeting_link,notes,visibility,
voting_starts_at,voting_ends_at,public_token,status,category_id,
recurrence_type,recurrence_count)
VALUES(%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,'open',%s,%s,%s)""",
(session["user_id"], title, description, location, meeting_link, notes, visibility,
voting_starts_at, voting_ends_at, public_token, category_id, recurrence_type, recurrence_count),
)

for idx, (ot, odt, on) in enumerate(options, start=1):
    execute("""INSERT INTO meeting_options(meeting_id,title,starts_at,ends_at,note,sort_order)
VALUES(%s,%s,%s,NULL,%s,%s)""", (meeting_id, ot, odt, on, idx))

for tag in {t.strip() for t in tags_text.split(",") if t.strip()} :
    execute("INSERT IGNORE INTO meeting_tags(meeting_id,tag) VALUES(%s,%s)", (meeting_id, tag[:60]))

for email in {e.strip().lower() for e in invite_emails.replace(";", ",").split(",") if e.strip()} :
    execute("INSERT IGNORE INTO meeting_invites(meeting_id,email) VALUES(%s,%s)", (meeting_id, email))
queue_email(email, f"Πρόσκληση: {title}", f"Έχεις πρόσκληση για την ψηφοφορία
«{title}».\n\n{url for('public meeting', token=public_token, external=True)}"))
```

```

audit("meeting_created", "meeting", meeting_id, {"title": title})
if recurrence_type == "weekly" and recurrence_count > 1:
    clone_weekly_meetings(meeting_id, recurrence_count)
flash("Ο προγραμματισμός δημιουργήθηκε.", "success")
return redirect(url_for("public_meeting", token=public_token))

return render_template("new_meeting.html", categories=categories, templates=templates,
    template_data=template_data, template_options=template_options,
    form_data=None, option_rows=None)

# -----

@app.route("/meeting/<int:meeting_id>/edit", methods=["GET", "POST"])
@login_required
def edit_meeting(meeting_id):
    meeting = query_one("SELECT * FROM meetings WHERE id=%s", (meeting_id,))
    if not meeting:
        abort(404)
    if session.get("role") != "admin" and meeting["owner_id"] != session["user_id"]:
        abort(403)

    categories = query_all("SELECT * FROM categories ORDER BY name")
    existing_tags = query_all("SELECT tag FROM meeting_tags WHERE meeting_id=%s ORDER BY tag",
(meeting_id,))

    if request.method == "POST":
        title = request.form.get("title", "").strip()
        description = request.form.get("description", "").strip()
        location = request.form.get("location", "").strip()
        meeting_link = request.form.get("meeting_link", "").strip()
        notes = request.form.get("notes", "").strip()
        visibility = request.form.get("visibility", "link")
        category_id = request.form.get("category_id", type=int)
        tags_text = request.form.get("tags", "").strip()
        voting_starts_at = normalize_datetime_value(request.form.get("voting_starts_at", ""))
        voting_ends_at = normalize_datetime_value(request.form.get("voting_ends_at", ""))

        option_ids = request.form.getlist("option_id[]")
        option_titles = request.form.getlist("option_title[]")
        option_datetimes = request.form.getlist("option_datetime[]")
        option_notes = request.form.getlist("option_note[]")
        deleted_ids = [int(x) for x in request.form.getlist("deleted_option_ids[]") if x.isdigit()]

    edit_rows = []
    parsed = []

```

```

count = max(len(option_ids), len(option_titles), len(option_datetimes), len(option_notes), 0)
for i in range(count):
    oid = option_ids[i].strip() if i < len(option_ids) else ""
    ot = option_titles[i].strip() if i < len(option_titles) else ""
    odt_raw = option_datetimes[i].strip() if i < len(option_datetimes) else ""

# -----
# Οι υπόλοιπες admin λειτουργίες της πλήρους έκδοσης περιλαμβάνουν επιπλέον
# αναζήτηση/φίλτρα χρηστών και meetings, pagination και συγκεντρωτικά analytics.
# Στο παράρτημα παραλείπονται για να διατηρηθεί αντιπροσωπευτικό μέγεθος.
# -----

@app.errorhandler(403)
def e403(error):
    return render_template("error.html", code=403, message="Δεν έχεις δικαίωμα πρόσβασης."), 403

@app.errorhandler(404)
def e404(error):
    return render_template("error.html", code=404, message="Η σελίδα δεν βρέθηκε."), 404

if __name__ == "__main__":
    print("UniMeet -> http://127.0.0.1:5000")
    app.run(debug=True)

```