ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ
ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΕΠΙΚΟΙΝΩΝΙΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Τίτλος Πτυχιακής Εργασίας	ΕΞΥΠΝΗ ΕΦΑΡΜΟΓΗ ΠΡΟΣΑΡΜΟΣΤΙΚΟΥ ΗΜΕΡΟΛΟΓΙΟΥ ΔΡΑΣΤΗΡΙΟΤΗΤΩΝ SMART APPLICATION FOR PERSONALISED ACTIVITY CALENDAR
Ονοματεπώνυμο Φοιτητή	ΤΟΥΡΚΟΧΩΡΙΤΗΣ ΠΑΝΑΓΙΩΤΗΣ
Πατρώνυμο	ΔΗΜΗΤΡΙΟΣ
ΑΡΙΘΜΟΣ ΜΗΤΡΩΟΥ	Π17145
Επιβλέπων	Δρ. ΧΡΥΣΑΦΙΑΔΗ ΚΩΝΣΤΑΝΤΙΝΑ, ΕΠΙΚΟΥΡΟΣ ΚΑΘΗΓΗΤΡΙΑ

Σεπτέμβριος 2024

1 Copyright ©

Απαγορεύεται η αντιγραφή, αποθήκευση και διανομή της παρούσας εργασίας, εξ ολοκλήρου ή τμήματος αυτής, για εμπορικό σκοπό. Επιτρέπεται η ανατύπωση, αποθήκευση και διανομή για σκοπό μη κερδοσκοπικό, εκπαιδευτικής ή ερευνητικής φύσης, υπό την προϋπόθεση να αναφέρεται η πηγή προέλευσης και να διατηρείται το παρόν μήνυμα.

Οι απόψεις και τα συμπεράσματα που περιέχονται σε αυτό το έγγραφο εκφράζουν αποκλειστικά τον συγγραφέα και δεν αντιπροσωπεύουν τις επίσημες θέσεις του Πανεπιστημίου Πειραιώς. Ως συγγραφέας της παρούσας εργασίας δηλώνω πως η παρούσα εργασία δεν αποτελεί προϊόν λογοκλοπής και δεν περιέχει υλικό από μη αναφερόμενες πηγές.

2 Περίληψη

Η παρακάτω πτυχιακή εργασία αφορά την δημιουργία ενός προσωποποιημένου ημερολογίου που δίνει την δυνατότητα στον χρήστη να επιλύει αυτόματα τυχόν χρονολογικές συμπτώσεις μεταξύ εγγραφών. Το My_Planner αποτελεί μια παραθυρική εφαρμογή ημερολογίου μέσω τις οποίας ο χρήστης μπορεί να εισάγει τις εγγραφές του μαζί με έναν βαθμό σημαντικότητας οποιαδήποτε ημέρα και ώρα θέλει. Βασικό χαρακτηριστικό της της εφαρμογής είναι ότι βάση της σημαντικότητας κάθε εγγραφής το πρόγραμμα αυτόματα επιλύει τυχόν συμπτώσεις μεταφέροντας χρονολογικά πιο μπροστά ή πιο πίσω τις λιγότερο σημαντικές εγγραφές, εφόσον ο χρήστης συναινέσει με αυτές.

3 Abstract

The following thesis concerns the creation of a personalized calendar that allows the user to automatically resolve any chronological conflicts between entries. My_Planner is a desktop calendar application through which the user can input their entries along with a degree of importance for any desired day and time. A key feature of the application is that based on the importance of each entry, the program automatically resolves any conflicts by moving less important entries forward or backward chronologically, provided the user consents to these changes.

4 Πίνακας Περιεχόμενων

5 Κατάλογος Εικόνων

6 Κατάλογος Διαγραμμάτων

7 Στόχοι Εργασίας

Στόχος της εργασίας είναι η δημιουργία και η ανάπτυξη ενός προσωποποιημένου ημερολογίου που θα μπορεί να επιλύει αυτόματα τυχόν χρονολογικές συμπτώσεις μεταξύ εγγραφών.

Μεγάλη έμφαση έχει δοθεί στην δημιουργία ενός ημερολογίου όπου ο κάθε χρήστης μπορεί να συνδεθεί στο προφίλ του και έτσι να έχει πρόσβαση σε ένα προσωπικό ημερολόγιο στο οποίο μπορεί να προσθέτει, να αφαιρεί και να τροποποιεί διάφορες εγγραφές σε οποιαδήποτε ημέρα και ώρα θέλει.

Τέλος, ένας από τους βασικότερους στόχους της εφαρμογής είναι η δημιουργία μιας αποτελεσματικής διαδικασίας μέσω της οποίας οποιαδήποτε χρονολογική σύμπτωση μεταξύ των εγγραφών θα επιλύεται αυτόματα από το πρόγραμμα βάση της σημαντικότητας της εγγραφής.

8 Εισαγωγή στα Προσαρμοστικά Συστήματα

8.1 Τι είναι η προσαρμοστικότητα / προσαρμοστικά συστήματα (adaptivity/ adaptive systems)?

Η προσαρμοστικότητα αναφέρεται στην ικανότητα ενός συστήματος ή τεχνολογίας να προσαρμόζει τις λειτουργίες του ως απόκριση σε αλλαγές στο περιβάλλον του ή στις αλληλεπιδράσεις των χρηστών.

Τα προσαρμοστικά συστήματα είναι συστήματα που σχεδιάζονται για να προσαρμόζουν την συμπεριφορά τους ως δράση σε αλλαγές στο περιβάλλον ή την εσωτερική τους κατάσταση. Τα συστήματα αυτά χρησιμοποιούν διάφορους μηχανισμούς ανάδρασης για να παρακολουθούν την απόδοσή τους και να κάνουν τις απαραίτητες τροποποιήσεις για τη βελτίωση της αποδοτικότητας, της αποτελεσματικότητας ή της ανθεκτικότητας.[14]

Μερικά χαρακτηριστικά αυτών των συστημάτων είναι :

- Self-Monitoring: Τα συστήματα αυτά παρακολουθούν συνέχεια την απόδοσή τους και το περιβάλλον του συστήματος για να μπορούν να ανιχνεύουν αλλαγές ή/και ανωμαλίες.
- Feedback Loops:Χρησιμοποιούν feedback loops για να συγκρίνουν την απόδοση τους με τα επιθυμητά αποτελέσματα και έτσι να προσαρμόζονται έτσι όπως απαιτείτε σε κάθε περίπτωση.
- Flexibility: Παρέχουν μια ευελιξία έτσι ώστε να λειτουργούν υπό διαφορετικές συνθήκες χωρίς να είναι απαραίτητη κάποιο χειροκίνητη παρέμβαση.
- Resilience: Σχεδιάζονται με σκοπό την ανθεκτικότητα ώστε να αποδίδουν σε οποιαδήποτε απροσδόκητη αλλαγή ή διαταραχή.

Η προσαρμοστικότητα αποτελεί βασικό χαρακτηριστικό σε πολλές σύγχρονες τεχνολογικές εφαρμογές, επιτρέποντας στα συστήματα να ανταποκρίνονται αποτελεσματικά σε δυναμικά και απρόβλεπτα περιβάλλοντα. Οι εφαρμογές των προσαρμοστικών συστημάτων είναι ποικίλες και περιλαμβάνουν[23]:

- Βιομηχανικούς Ελέγχους: Χρήση σε συστήματα ελέγχου παραγωγής που προσαρμόζονται σε πραγματικό χρόνο για να βελτιστοποιήσουν την απόδοση. Για παράδειγμα, τα Προγραμματιζόμενα Λογικά Ελεγκτικά Συστήματα (PLCs) χρησιμοποιούνται για την αυτοματοποίηση και τον έλεγχο βιομηχανικών διαδικασιών, προσαρμοζόμενα σε αλλαγές στη διαδικασία ή στο περιβάλλον παραγωγής.
- Διαχείριση Δικτύων: Συστήματα που προσαρμόζονται για να αντιμετωπίσουν μεταβαλλόμενους όγκους δεδομένων και απειλές ασφαλείας. Στον τομέα της κυβερνοασφάλειας, τα προσαρμοστικά συστήματα μπορούν να αναγνωρίσουν και να αντιδράσουν σε νέες απειλές σε πραγματικό χρόνο, βελτιώνοντας την ασφάλεια των δικτύων.
- Αυτόνομα Οχήματα: Οχήματα που προσαρμόζονται στις συνθήκες κυκλοφορίας και του δρόμου για να βελτιώσουν την ασφάλεια και την αποδοτικότητα. Αυτά τα οχήματα χρησιμοποιούν αισθητήρες και αλγορίθμους μηχανικής μάθησης για να προσαρμόσουν τη συμπεριφορά τους σε πραγματικό χρόνο, εξασφαλίζοντας ασφαλή και αποδοτική οδήγηση.
- Έξυπνα Κτίρια: Συστήματα διαχείρισης κτιρίων που προσαρμόζονται στις συνθήκες των χώρων, όπως θερμοκρασία, φωτισμός και κατανάλωση ενέργειας, για να βελτιώσουν την άνεση των ενοίκων και να μειώσουν την ενεργειακή κατανάλωση.
- Υγεία και Ιατρική: Προσαρμοστικά συστήματα που χρησιμοποιούνται στην παρακολούθηση και διαχείριση της υγείας των ασθενών. Για παράδειγμα, οι φορητές συσκευές υγείας μπορούν να προσαρμόζουν τις ειδοποιήσεις και τις προτάσεις τους βάσει των δεδομένων υγείας που συλλέγουν σε πραγματικό χρόνο.

Η προσαρμοστικότητα είναι κρίσιμη για τη βελτίωση της αποδοτικότητας και της ανθεκτικότητας των συστημάτων σε διάφορους τομείς, από τη βιομηχανία μέχρι την τεχνολογία πληροφοριών και επικοινωνιών.[15]

8.2 Intelligent & adaptive system

Τα έξυπνα και προσαρμοστικά συστήματα μοιράζονται τα χαρακτηριστικά που περιγράφονται και πιο πάνω, προσθέτοντας όμως επιπλέον δυνατότητες στο περιβάλλον τους. Συνδυάζουν τις δυνατότητες της τεχνητής νοημοσύνης (AI) [33] με προσαρμοστικούς μηχανισμούς για να ανταποκρίνονται δυναμικά σε μεταβαλλόμενα περιβάλλοντα και συνθήκες. Αυτά τα συστήματα είναι σχεδιασμένα όχι μόνο να εκτελούν εργασίες αυτόνομα αλλά και να βελτιώνουν την απόδοσή τους με την πάροδο του χρόνου μέσω της μάθησης και της προσαρμογής [24].

Κύρια Χαρακτηριστικά:

- Αυτονομία και Αυτοβελτίωση: Τα έξυπνα και προσαρμοστικά συστήματα μπορούν να λειτουργούν αυτόνομα, λαμβάνοντας αποφάσεις και εκτελώντας εργασίες χωρίς ανθρώπινη παρέμβαση. Επιπλέον, χρησιμοποιούν τεχνικές μηχανικής μάθησης για να βελτιώνουν την απόδοσή τους με την πάροδο του χρόνου, μαθαίνοντας από τις εμπειρίες και τα δεδομένα που συλλέγουν.
- Δυναμική Ανταπόκριση: Είναι σε θέση να αντιδρούν σε πραγματικό χρόνο σε αλλαγές στο περιβάλλον τους, προσαρμόζοντας τις ενέργειές τους για να επιτύχουν τους στόχους τους. Αυτό περιλαμβάνει την προσαρμογή σε νέες συνθήκες λειτουργίας, όπως αλλαγές στις απαιτήσεις των χρηστών ή στον όγκο των δεδομένων.
- Συνεργασία και Επικοινωνία: Μπορούν να συνεργάζονται με άλλα συστήματα και να ανταλλάσσουν πληροφορίες για να βελτιώσουν τη συνολική απόδοση του δικτύου ή του συστήματος στο οποίο ανήκουν. Αυτό είναι ιδιαίτερα σημαντικό σε περιβάλλοντα όπου απαιτείται συντονισμένη δράση, όπως σε δίκτυα κυβερνοασφάλειας ή σε αυτόνομα οχήματα.
- Προγνωστική Ανάλυση: Χρησιμοποιούν τεχνικές ανάλυσης δεδομένων για να προβλέπουν μελλοντικές τάσεις και να προσαρμόζουν τη λειτουργία τους ανάλογα. Αυτό επιτρέπει την πρόληψη προβλημάτων πριν αυτά εμφανιστούν και τη βελτιστοποίηση της απόδοσης.

Εφαρμογές:

- Έξυπνα Δίκτυα Ενέργειας: Προσαρμόζονται δυναμικά στη ζήτηση και την προσφορά ενέργειας, βελτιώνοντας την απόδοση και μειώνοντας τις απώλειες.
- Προηγμένα Συστήματα Υγείας: Παρακολουθούν συνεχώς τους ασθενείς και προσαρμόζουν τις θεραπείες με βάση τα δεδομένα υγείας σε πραγματικό χρόνο.
- Αυτόνομα Ρομπότ: Χρησιμοποιούνται σε βιομηχανίες για την εκτέλεση εργασιών που απαιτούν υψηλή ακρίβεια και προσαρμόζονται σε αλλαγές στο περιβάλλον εργασίας.
- Έξυπνα Συστήματα Μεταφορών: Βελτιώνουν τη ροή της κυκλοφορίας και την ασφάλεια στους δρόμους,

προσαρμόζοντας τις ενέργειές τους βάσει των δεδομένων από αισθητήρες και άλλες πηγές.

Τα έξυπνα και προσαρμοστικά συστήματα αποτελούν το μέλλον της τεχνολογίας, προσφέροντας λύσεις που είναι όχι μόνο αποδοτικές αλλά και ευέλικτες, ικανές να ανταποκριθούν στις συνεχώς μεταβαλλόμενες ανάγκες και προκλήσεις του σύγχρονου κόσμου.[11]

8.3 Σχέση Τεχνητής Νοημοσύνης και προσαρμοστικών συστημάτων.

Η σχέση μεταξύ τεχνητής νοημοσύνης και προσαρμοστικών συστημάτων είναι εξαιρετικά στενή, καθώς η τεχνητή νοημοσύνη αποτελεί βασικό στοιχείο για την ανάπτυξη και τη λειτουργία αποτελεσματικών προσαρμοστικών συστημάτων [26]. Τα προσαρμοστικά συστήματα χρησιμοποιούν την τεχνητή νοημοσύνη για να προσαρμόζονται και να βελτιώνονται συνεχώς σε δυναμικά περιβάλλοντα, βασιζόμενα σε δεδομένα και εμπειρίες.[10,35]

Κύριες πτυχές της σχέσης [34]:

Ανάλυση Δεδομένων:

- Η τεχνητή νοημοσύνη παρέχει ισχυρά εργαλεία για την ανάλυση μεγάλου όγκου δεδομένων σε πραγματικό χρόνο. Μέσω αυτής της ανάλυσης, τα προσαρμοστικά συστήματα μπορούν να εντοπίζουν μοτίβα και τάσεις που δεν είναι εύκολα αντιληπτά από τους ανθρώπους [27].

Λήψη Αποφάσεων Χωρίς Παρέμβαση:

- Η τεχνητή νοημοσύνη επιτρέπει στα προσαρμοστικά συστήματα να λαμβάνουν αποφάσεις αυτόνομα, χωρίς την ανάγκη για ανθρώπινη παρέμβαση. Αυτό επιταχύνει τη διαδικασία λήψης αποφάσεων και αυξάνει την αποδοτικότητα.

Αυτοματοποίηση και Αυτονομία:

- Μέσω της τεχνητής νοημοσύνης, τα προσαρμοστικά συστήματα μπορούν να αυτοματοποιήσουν πολλές διαδικασίες, μειώνοντας το ανθρώπινο λάθος και απελευθερώνοντας πόρους για πιο σύνθετες εργασίες. Η αυτονομία αυτή είναι κρίσιμη για την αποτελεσματική λειτουργία σε περιβάλλοντα όπου οι συνθήκες αλλάζουν συνεχώς.

Προγνωστική Ανάλυση:

- Η προγνωστική ανάλυση, που βασίζεται σε αλγορίθμους τεχνητής νοημοσύνης, επιτρέπει στα προσαρμοστικά συστήματα να προβλέπουν μελλοντικές τάσεις και να προετοιμάζονται ανάλογα. Αυτό είναι ιδιαίτερα χρήσιμο για την αντιμετώπιση απρόβλεπτων αλλαγών και προβλημάτων [25].

Βελτιστοποίηση της Απόδοσης:

- Η τεχνητή νοημοσύνη βοηθά στη συνεχή βελτιστοποίηση της απόδοσης των προσαρμοστικών συστημάτων, εξασφαλίζοντας ότι λειτουργούν πάντα με τον πιο αποδοτικό τρόπο. Μέσω της συνεχούς αξιολόγησης και αναπροσαρμογής, τα συστήματα μπορούν να επιτυγχάνουν καλύτερα αποτελέσματα με λιγότερους πόρους.

Δυναμική Εκμάθηση:

- Η δυναμική εκμάθηση, η οποία αποτελεί βασικό χαρακτηριστικό της τεχνητής νοημοσύνης, επιτρέπει στα προσαρμοστικά συστήματα να μαθαίνουν και να βελτιώνονται συνεχώς. Καθώς τα συστήματα εκτίθενται σε νέα δεδομένα και

εμπειρίες, μπορούν να αναπροσαρμόζουν τις στρατηγικές και τις λειτουργίες τους για να επιτυγχάνουν καλύτερα αποτελέσματα.[16, 22]

Συνοψίζοντας, η τεχνητή νοημοσύνη όχι μόνο ενισχύει τις δυνατότητες των προσαρμοστικών συστημάτων αλλά και αποτελεί το θεμέλιο για την ανάπτυξή τους [20]. Οι τεχνολογίες της τεχνητής νοημοσύνης προσφέρουν τα μέσα για την αποτελεσματική ανάλυση δεδομένων, τη λήψη αποφάσεων, την αυτοματοποίηση και τη συνεχή βελτίωση, καθιστώντας τα προσαρμοστικά συστήματα πιο αποδοτικά και ικανά να ανταποκρίνονται στις προκλήσεις των σύγχρονων δυναμικών περιβαλλόντων.[20]

8.4 Γιατί είναι χρήσιμα; Τι προσφέρουν;

Τα προσαρμοστικά συστήματα είναι πολύτιμα για διάφορους λόγους, καθώς προσφέρουν μια σειρά από πλεονεκτήματα που ενισχύουν την αποδοτικότητα, την εμπειρία του χρήστη και την ανθεκτικότητα των συστημάτων.[11]

Αναλυτικότερα:

Εξατομίκευση της Εμπειρίας των Χρηστών:

- Τα προσαρμοστικά συστήματα έχουν τη δυνατότητα να προσαρμόζουν τις υπηρεσίες και το περιεχόμενο σύμφωνα με τις ατομικές προτιμήσεις και ανάγκες κάθε χρήστη. Αυτό οδηγεί σε μια πιο προσωπική και ικανοποιητική εμπειρία, αυξάνοντας την ικανοποίηση και την εμπιστοσύνη των χρηστών.

Αύξηση της Αποδοτικότητας:

- Με τη χρήση προηγμένων αλγορίθμων και αναλυτικών εργαλείων, τα προσαρμοστικά συστήματα βελτιστοποιούν τη διαχείριση πόρων και τη λειτουργία των διαδικασιών. Αυτό έχει ως αποτέλεσμα τη μείωση του κόστους και την αύξηση της παραγωγικότητας.

Κλιμακούμενη Διαχείριση Μεγάλων Συνόλων Δεδομένων:

- Τα προσαρμοστικά συστήματα μπορούν να διαχειρίζονται τεράστια ποσά δεδομένων, αξιοποιώντας τις πληροφορίες αυτές για τη λήψη αποφάσεων και τη βελτίωση των υπηρεσιών. Η δυνατότητα κλιμάκωσης είναι κρίσιμη σε περιβάλλοντα με συνεχή αύξηση του όγκου των δεδομένων.

Βελτίωση της Λήψης Αποφάσεων μέσω Ανάλυσης Δεδομένων:

- Χρησιμοποιώντας τεχνολογίες τεχνητής νοημοσύνης και αναλυτικής, τα προσαρμοστικά συστήματα προσφέρουν ακριβείς και έγκαιρες πληροφορίες που διευκολύνουν τη λήψη στρατηγικών αποφάσεων. Αυτό επιτρέπει την πρόβλεψη και την αποτροπή προβλημάτων πριν αυτά εμφανιστούν.

Ανθεκτικότητα και Προσαρμοστικότητα:

- Τα προσαρμοστικά συστήματα είναι σχεδιασμένα να ανταποκρίνονται δυναμικά στις αλλαγές του περιβάλλοντος ή στη συμπεριφορά των χρηστών. Αυτό τα καθιστά εξαιρετικά ανθεκτικά σε απρόβλεπτες συνθήκες, εξασφαλίζοντας την συνεχή και ομαλή λειτουργία τους ακόμη και σε περιπτώσεις κρίσεων.

Αυτονομία και Αυτοβελτίωση:

- Με τη δυνατότητα να μαθαίνουν από τις εμπειρίες και τα δεδομένα, τα προσαρμοστικά συστήματα βελτιώνονται συνεχώς, καθιστώντας τα πιο αποτελεσματικά και ακριβή με την πάροδο του χρόνου. Αυτή η συνεχής αυτοβελτίωση είναι κρίσιμη για την επίτευξη μακροπρόθεσμης επιτυχίας και βιωσιμότητας.

Εφαρμογές σε Ποικίλους Τομείς:

- Τα προσαρμοστικά συστήματα βρίσκουν εφαρμογές σε πληθώρα τομέων, όπως η υγεία, η εκπαίδευση, η βιομηχανία, οι υπηρεσίες και το λιανεμπόριο. Σε κάθε τομέα, συμβάλλουν στη βελτίωση της αποδοτικότητας, της ποιότητας των υπηρεσιών και της ικανοποίησης των χρηστών.

Συνοψίζοντας, τα προσαρμοστικά συστήματα προσφέρουν σημαντικά πλεονεκτήματα που συμβάλλουν στη βελτίωση της συνολικής απόδοσης και της εμπειρίας των χρηστών, καθιστώντας τα απαραίτητα εργαλεία για την αντιμετώπιση των προκλήσεων των σύγχρονων δυναμικών περιβαλλόντων.[18]

8.5 Πώς επηρεάζουν το user experience;

Τα προσαρμοστικά συστήματα βελτιώνουν σημαντικά την εμπειρία των χρηστών με διάφορους τρόπους. Εξατομικεύουν τις αλληλεπιδράσεις προσαρμόζοντας το περιεχόμενο και τις λειτουργίες στις ατομικές προτιμήσεις και συμπεριφορές κάθε χρήστη, προσφέροντας έτσι μια πιο προσωπική και ικανοποιητική εμπειρία. Προσαρμόζουν τις διαδικασίες ώστε να κάνουν τις εργασίες ευκολότερες και ταχύτερες, μειώνοντας την προσπάθεια που απαιτείται από τον χρήστη και αυξάνοντας την αποδοτικότητα.[19]

Τα προσαρμοστικά συστήματα παρέχουν ένα user interface που προσαρμόζεται στις ανάγκες και τις προτιμήσεις του εκάστοτε χρήστη, κάνοντας την πλοήγηση και τη χρήση των συστημάτων πιο ευχάριστη. Αυτή η προσαρμογή βελτιώνει τη συνολική εμπειρία χρήστη και καθιστά τη χρήση των συστημάτων πιο φιλική [32].

Επιπλέον, τα προσαρμοστικά συστήματα μαθαίνουν συνεχώς από τις αλληλεπιδράσεις με τους χρήστες και αναπροσαρμόζουν τις λειτουργίες και τις προτάσεις τους, βελτιώνοντας συνεχώς την εμπειρία χρήστη. Αυτή η δυναμική εκμάθηση εξασφαλίζει ότι το σύστημα παραμένει σχετικό και αποδοτικό, ανταποκρινόμενο στις αλλαγές των προτιμήσεων και των αναγκών των χρηστών.

Τέλος, προσφέροντας εξατομικευμένες εμπειρίες και ευκολία στη χρήση, τα προσαρμοστικά συστήματα ενισχύουν την ικανοποίηση των χρηστών. Οι χρήστες αισθάνονται ότι το σύστημα κατανοεί και ανταποκρίνεται στις ανάγκες τους, αυξάνοντας έτσι την εμπιστοσύνη και την αφοσίωσή τους.[17, 31]

8.6 Προσαρμοστικότητα και ημερολόγια δραστηριοτήτων

Η συνεργασία μεταξύ προσαρμοστικών συστημάτων και ημερολογίων δραστηριοτήτων μπορεί να έχει σημαντικά οφέλη για τους χρήστες. Τα προσαρμοστικά συστήματα λειτουργούν αναλύοντας τον τρόπο που οι χρήστες διαχειρίζονται τις δραστηριότητές τους και τις προτιμήσεις τους στον προγραμματισμό του χρόνου τους. Αυτή η ανάλυση επιτρέπει στα προσαρμοστικά συστήματα να προτείνουν ιδανικούς χρόνους για τις διάφορες δραστηριότητες, λαμβάνοντας υπόψη τη συχνότητα και την προτεραιότητα των εργασιών. [12]

Πέραν αυτού, τα προσαρμοστικά συστήματα μπορούν να παρέχουν προσαρμοσμένες υπενθυμίσεις και ειδοποιήσεις που ενσωματώνουν τις επιταγές της καθημερινής ζωής του χρήστη [21]. Αυτό σημαίνει ότι μπορούν να υποδεικνύουν στον χρήστη πότε είναι η κατάλληλη στιγμή για κάθε είδους δραστηριότητα, βοηθώντας τον να οργανώσει την ημέρα του αποτελεσματικότερα και να αποφύγει την υπερφόρτωση ή την αναβολή εργασιών [29].

Επιπλέον, η προσαρμοστικότητα του ημερολογίου μπορεί να ενισχύσει την αποτελεσματικότητα και την παραγωγικότητα του χρήστη, καθώς επιτρέπει στον χρήστη να επικεντρωθεί περισσότερο στις προτεραιότητές του και να διαχειρίζεται τον χρόνο του με πιο αποτελεσματικό τρόπο [30]. Κατ' αυτόν τον τρόπο, η συνεργασία αυτών των δύο τεχνολογιών

δημιουργεί μια ολοκληρωμένη εμπειρία χρήστη που είναι εξατομικευμένη, παρέχοντας ταυτόχρονα εργαλεία για βελτίωση της οργάνωσης και της παραγωγικότητας.[13,28]

9 Περιγραφή Προβλήματος

Στις μέρες μας τα πάντα κινούνται σε τόσο γρήγορους ρυθμούς που οι περισσότεροι άνθρωποι δεν μπορούν να συμβαδίσουν με τον καθημερινό φόρτο εργασιών που έχουν να διεκπεραιώσουν μέσα στην ημέρα τους.

Έτσι, πολλές φορές βρισκόμαστε σε μια σύγχυση όσον αφορά την οργάνωση των καθημερινών μας απαιτήσεων και εργασιών καθώς και ερχόμαστε πολύ συχνά σε διλλήματα που αφορούν την διεκπεραίωση εργασιών που συνάδουν χρονικά.

Για τον λόγο αυτό, το My_Planner καλείται να ξεπεράσει αυτές τις δυσκολίες και να δώσει την δυνατότητα στους χρήστες του, όχι μόνο να έχουν ένα προσωποποιημένο ημερολόγιο στο οποίο μπορούν να εισάγουν τις καθημερινές δουλείες που έχουν ανά ημέρα και ώρα αλλά και να τους λύνει τα χέρια επιλύοντας αυτόματα τυχόν χρονολογικές συμπτώσεις βάση ενός αρχικού συντελεστή σημαντικότητας που θέτει ο κάθε χρήστης για την συγκεκριμένη εγγραφή.

10 Αναφορά σε Παρόμοιες Εφαρμογές

Είναι γεγονός ότι το προαναφερθέν πρόβλημα είναι κάτι που βασανίζει την σημερινή κοινωνία για πολλά χρόνια. Αυτό έχει ως αποτέλεσμα την δημιουργία πολλών εφαρμογών που καλύπτουν αυτές τις ανάγκες καθώς και την ενσωμάτωση τους σε συσκευές που χρησιμοποιούμε στην καθημερινότητά μας όπως τα κινητά τηλέφωνα και οι υπολογιστές σαν προ εγκατεστημένες εφαρμογές.

Παρακάτω θα γίνει μια σύντομη αναφορά σε παρόμοιες εφαρμογές. Πιο συγκεκριμένα θα παρουσιαστούν τα εξής:

- Google calendar
- MS outlook Calendar
- Calendar (IOS application)
- Click up

Για όλες τις εφαρμογές δημιουργήθηκε ένας λογαριασμός χρήστη με ίδια χαρακτηριστικά. Σε κάθε μία δημιουργήθηκαν 2 διαφορετικές εγγραφές, με ίδια ακριβώς χαρακτηριστικά, οι οποίες συμπίπτουν χρονολογικά. Αυτό που θέλουμε να δούμε είναι πως αυτές οι εφαρμογές αντιδρούν, ποιες είναι οι δυνατότητες που προσφέρουν καθώς τι λύση προτείνουν σε περίπτωση που οι δύο μας εγγραφές συμπίπτουν.

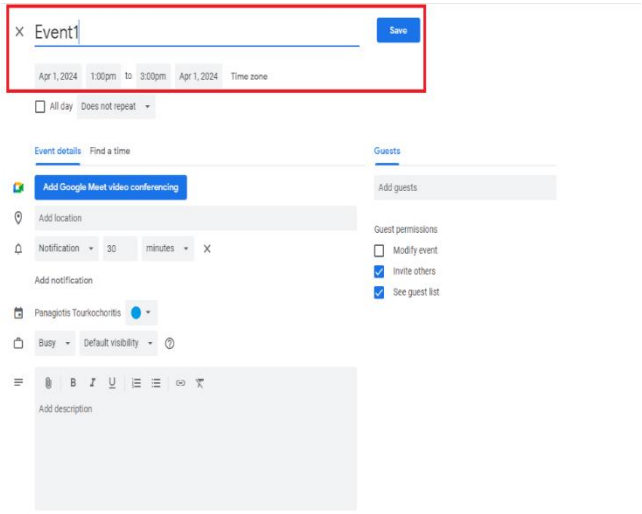
10.1 Google calendar

Το Google Calendar είναι μια υπηρεσία ημερολογίου διαχείρισης χρόνου και προγραμματισμού που αναπτύχθηκε από την Google. Έγινε διαθέσιμο σε έκδοση beta στις 13 Απριλίου 2006 και σε γενική κυκλοφορία τον Ιούλιο του 2009.

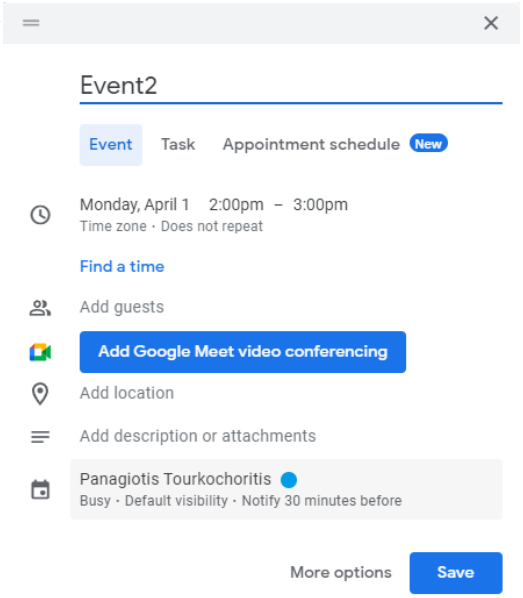
Αυτή η πλατφόρμα είναι χρήσιμη για τον προγραμματισμό, την παρακολούθηση και τη διαχείριση εκδηλώσεων και συσκέψεων. Επιπλέον, οι χρήστες μπορούν:

- Να δημιουργήσουν και να προσαρμόσουν πολλά ημερολόγια για προσωπικές, εργασιακές και άλλες δραστηριότητες.
- Να αποκτήσουν πρόσβαση σε υπενθυμίσεις και ειδοποιήσεις συμβάντων μέσω email ή ειδοποιήσεων μέσω κινητού τηλεφώνου.
- Να μοιραστούν ημερολόγια με άλλους χρήστες και να μένουν ενημερωμένοι για το πρόγραμμα και τη διαθεσιμότητα της ομάδας τους.
- Να το ενσωματώσουν με άλλες υπηρεσίες της Google όπως το Gmail, το Google Meet και το Google Tasks, διευκολύνοντας τη διαχείριση χρονοδιαγραμμάτων και συμβάντων.
- Να ρυθμίσουν επαναλαμβανόμενα συμβάντα για ημερήσιες, εβδομαδιαίες, μηνιαίες ή ετήσιες συσκέψεις.

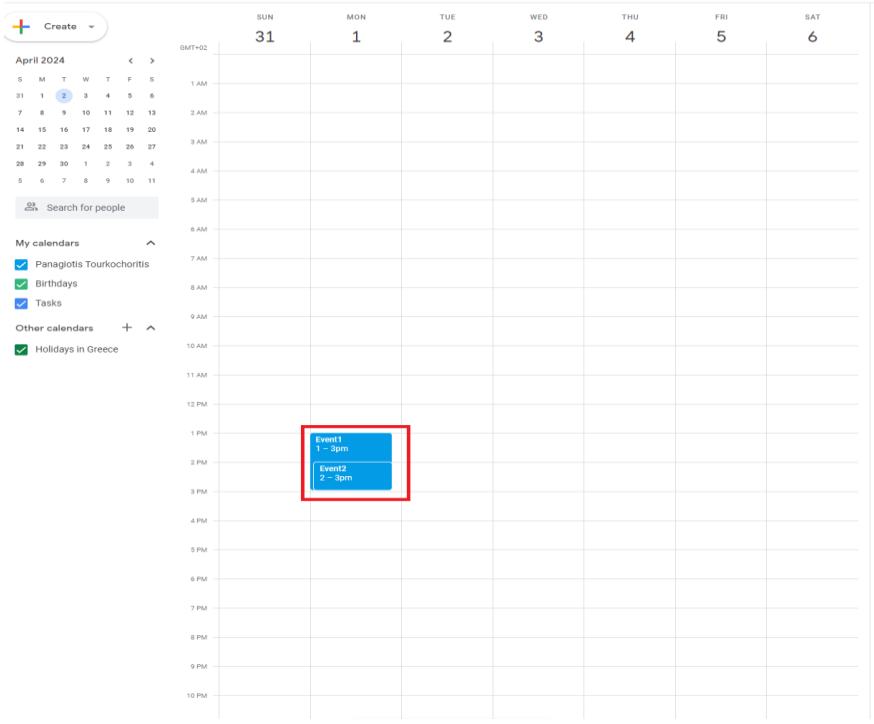
10.1.1 Σενάριο χρήσης



ΕΙΚΟΝΑ 1



ΕΙΚΟΝΑ 2



ΕΙΚΟΝΑ 3

Παρατηρούμε λοιπόν ότι κατά την δημιουργία των εγγραφών, παρόλο που το σύστημα μας προσφέρει αρκετές δυνατότητες, δεν έχουμε κάποια επιλογή να ορίσουμε κάποια σημαντικότητα για αυτές (Εικόνα 1&2). Καθ' όλη την διάρκεια της δημιουργίας το πρόγραμμα δεν

μας ενημέρωσε ότι οι εγγραφές μας συμπίπτουν χρονικά ο μόνος τρόπος να το δούμε είναι να εισέλθουμε στο ημερολόγιο που παρέχει η εφαρμογή και να το παρατηρήσουμε μόνοι μας (Εικόνα 3).

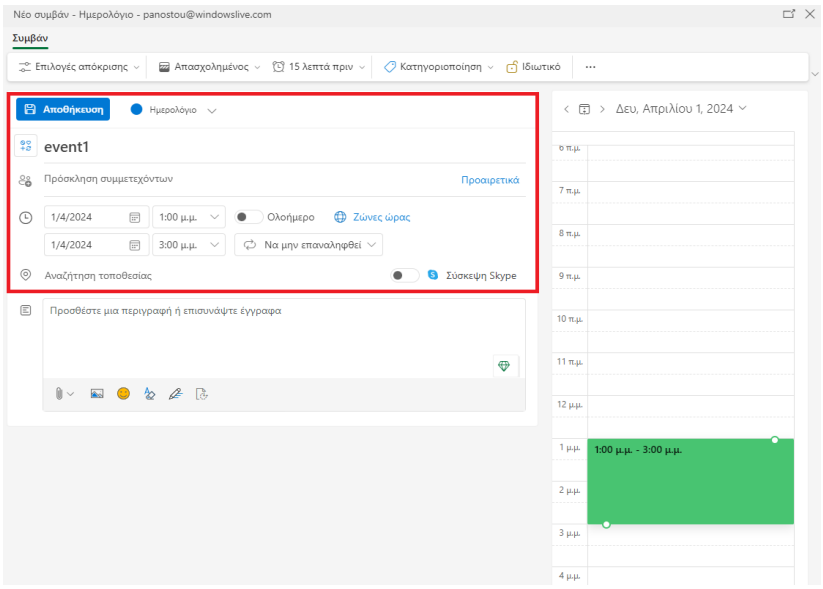
Αυτό δεν είναι πολύ βολικό καθώς στην περίπτωση ενός ημερολογίου γεμάτου από εγγραφές θα ήταν δύσκολο να παρατηρήσουμε αυτές που συμπίπτουν χρονολογικά. Η εφαρμογή μας παρέχει την δυνατότητα να επιλέξουμε εμείς διαφορετικά χρώματα για κάθε εγγραφή μας ώστε να μπορούμε να κωδικοποιήσουμε χρωματικά (color coding) τις εγγραφές. Αυτό όμως προσθέτει επιπλέον χειροκίνητη εργασία από τον χρήστη καθώς και προϋποθέτει ο χρήστης να θυμάται τα χρώματα που έχει επιλέξει.

10.2 MS Outlook Calendar

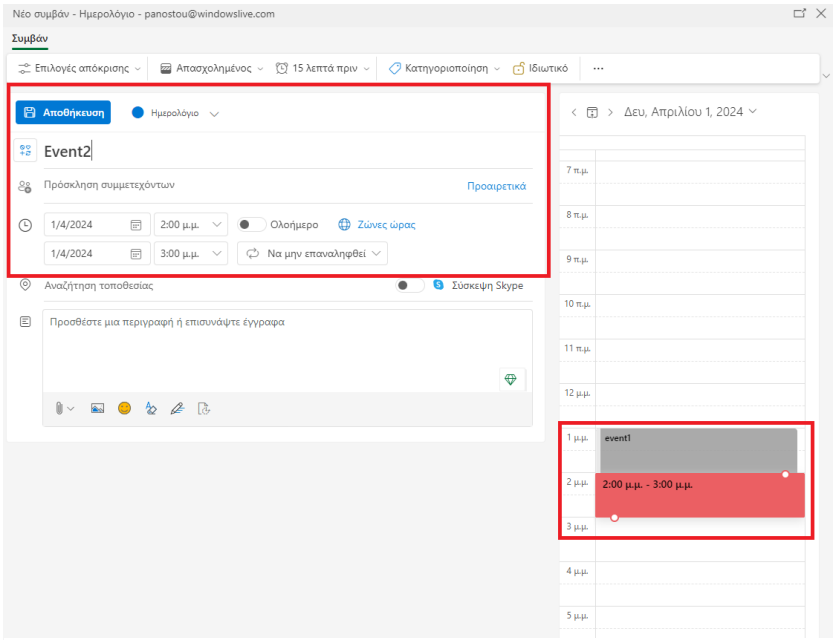
Το Microsoft Outlook είναι ένα ευρέως χρησιμοποιούμενο πρόγραμμα ηλεκτρονικού ταχυδρομείου που διαθέτει έναν ενσωματωμένο ημερολόγιο γνωστό ως Outlook Calendar. Το Outlook Calendar επιτρέπει στον χρήστη να διαχειρίζεται τα ραντεβού, τις συναντήσεις με έναν ομαλό και βολικό τρόπο.

Ένα από τα ξεχωριστά χαρακτηριστικά του είναι η δυνατότητα αποστολής προσκλήσεων σε συσκέψεις απευθείας από το ημερολόγιο, βελτιστοποιώντας τη συνεργασία με άλλους χρήστες. Επιπλέον, το Ημερολόγιο του Outlook υποστηρίζει την κοινή χρήση ημερολογίου καθώς και την προβολή πολλών ημερολογίων, ιδανικό για την εξισορρόπηση προσωπικών και επαγγελματικών υποχρεώσεων.

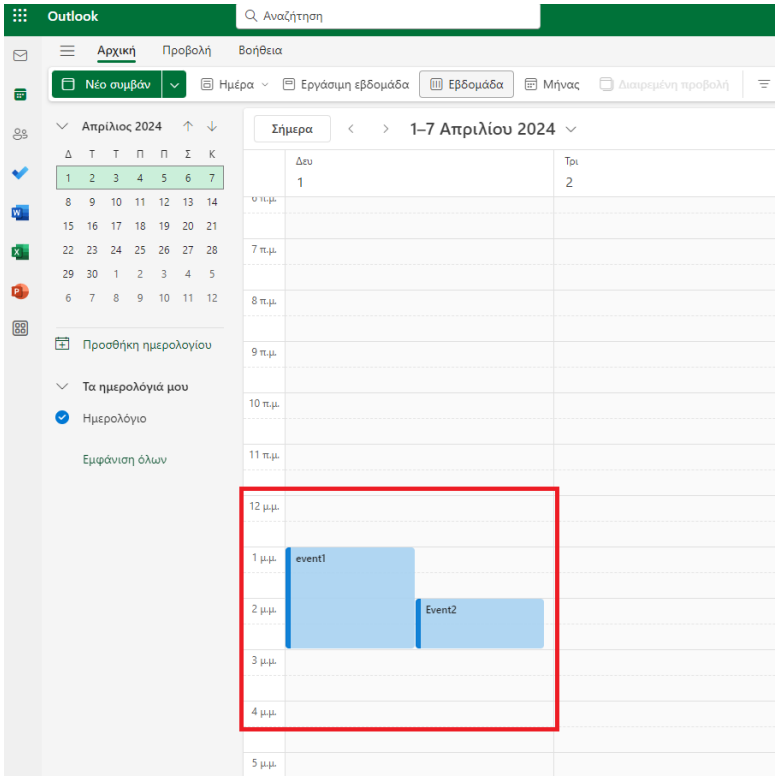
10.2.1 Σενάριο χρήσης



ΕΙΚΟΝΑ 4



ΕΙΚΟΝΑ 5



ΕΙΚΟΝΑ 6

Όπως και στην προηγούμενη εφαρμογή το σύστημα μας προσφέρει αρκετές δυνατότητες χωρίς να έχουμε όμως κάποια επιλογή να ορίσουμε κάποια σημαντικότητα για αυτές (Εικόνα 4).

Υποστηρίζει επίσης τον χαρακτηρισμό των εγγραφών με ετικέτες έτσι ώστε να μπορούμε να ξεχωρίσουμε τις εγγραφές μας (Εικόνα 4).

Σε αυτήν την εφαρμογή παρατηρούμε μια εντελώς διαφορετική αντιμετώπιση κατά την δημιουργία των εγγραφών σε σχέση με πριν. Όταν δημιουργούμε την δεύτερη εγγραφή, η οποία συμπίπτει χρονολογικά με την πρώτη, το σύστημα μας ενημερώνει δείχνοντας μας σε πραγματικό χρόνο με κόκκινο χρώμα ότι έχουμε αυτήν την σύμπτωση (Εικόνα 5). Επίσης, όταν κοιτάμε το ημερολόγιο στην συγκεκριμένη ημερομηνία βλέπουμε τα event μας το ένα δίπλα στο άλλο, σε αντίθεση με πριν που ήταν το ένα πάνω από το άλλο (Εικόνα 6). Αυτό παρέχει μια καλύτερη εικόνα στον χρήστη, έτσι ώστε να μπορεί να δει τις εγγραφές που συμπίπτουν χρονολογικά.

Τέλος, και σε αυτήν την περίπτωση δεν είχαμε καμία ενημέρωση, με μορφή ειδοποίησης, για αυτήν την σύμπτωση αλλά ούτε και την δυνατότητα να την επιλύσουμε αυτόματα.

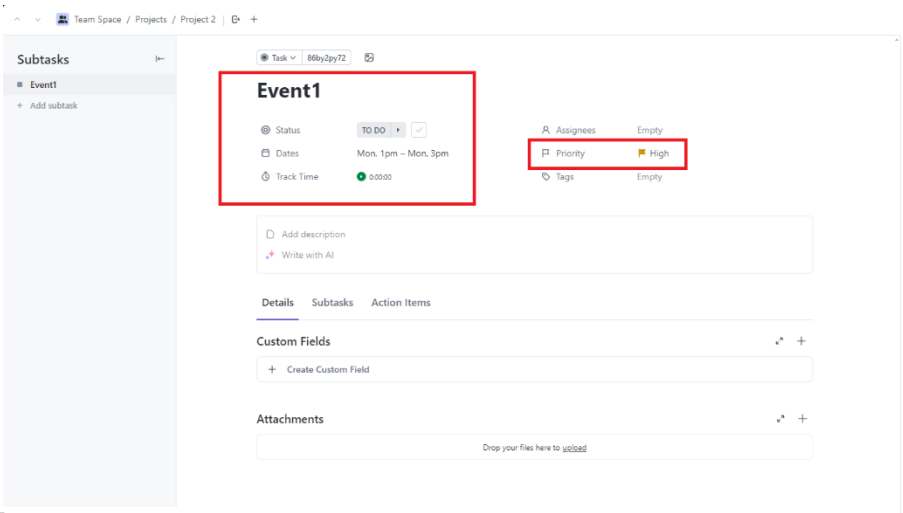
10.3 ClickUp

Το ClickUp είναι μια πλατφόρμα παραγωγικότητας με εύχρηστες λειτουργίες Ημερολογίου που έχουν σχεδιαστεί για να απλοποιούν εργασίες, χρονοδιαγράμματα και συμβάντα.

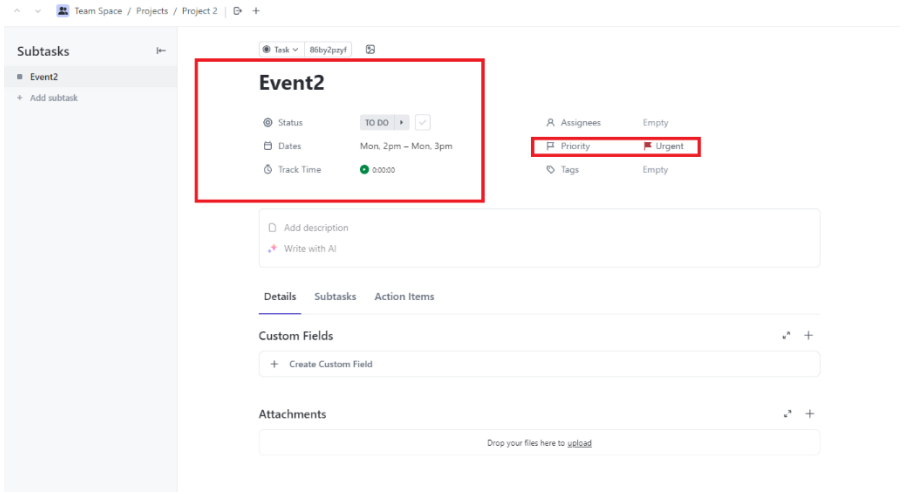
Τα βασικά χαρακτηριστικά που προσφέρει είναι τα εξής:

- Καθημερινές, εβδομαδιαίες και μηνιαίες προβολές για εύκολη προσθήκη εργασιών και νέων συμβάντων
- Προσαρμόσιμη προβολή για τη δημιουργία διαφόρων τύπων ημερολογίου
- Δυνατότητα επισύναψης αρχείων
- Κοινόχρηστα ημερολόγια με άλλους χρήστες
- Ταξινόμηση εργασιών κατά κατάσταση και προτεραιότητα
- Συγχρονισμός διπλής κατεύθυνσης Google calendar, Outlook calendar ή Apple calendar.

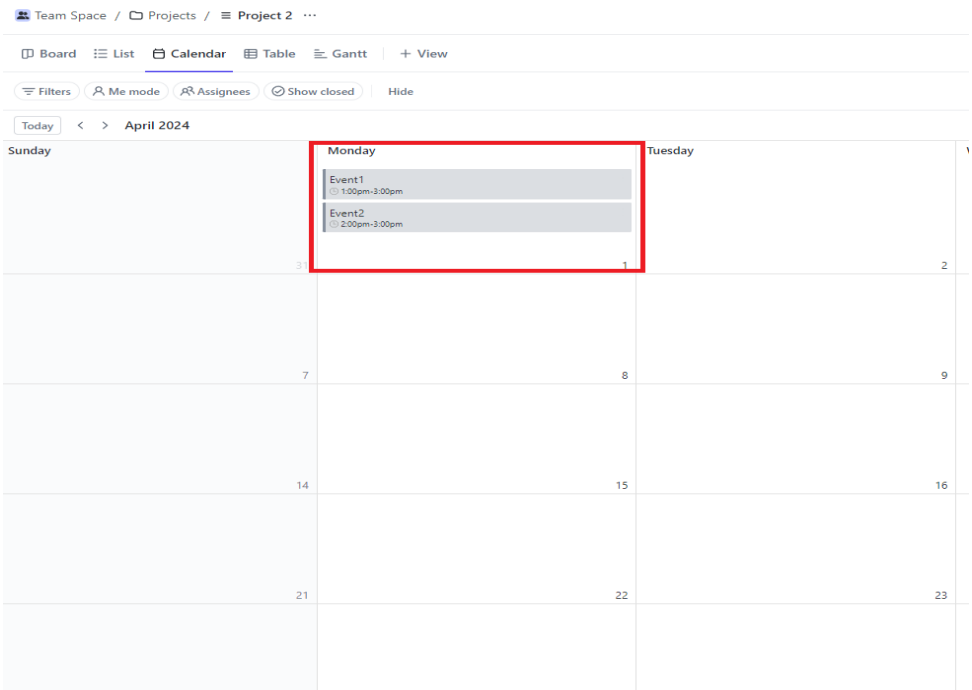
10.3.1 Σενάριο χρήσης



ΕΙΚΟΝΑ 7


Team Space / Projects / Project 2
Subtasks
Event2
Add subtask
Event2
Status
TO DO
Dates
Mon, 2pm – Mon, 3pm
Track Time
Assignees
Empty
Priority
Urgent
Tags
Empty
Add description
Write with AI
Details
Subtasks
Action Items
Custom Fields
Create Custom Field
Attachments
Drop your files here to upload

ΕΙΚΟΝΑ 8


Team Space / Projects / Project 2
Board
List
Calendar
Table
Gantt
View
Filters
Me mode
Assignees
Show closed
Hide
Today
April 2024
Sunday
Monday
Tuesday
Event1
1:00pm-3:00pm
Event2
2:00pm-3:00pm

ΕΙΚΟΝΑ 9

Στην περίπτωση του ClickUp παρατηρούμε πως κατά την διάρκεια την δημιουργίας των εγγραφών μας δίνεται η δυνατότητα να θέσουμε κάποιο επίπεδο σημαντικότητας για αυτές (Εικόνα 7&8). Παρόλα αυτά, καθώς δημιουργούμε την εγγραφή δεν υπάρχει καμία ειδοποίηση για ούτε για την ύπαρξη σύμπτωσης αλλά ούτε και κάποια πρόταση για επίλυση βάση σημαντικότητας(Εικόνα 9).

Η συγκεκριμένη εφαρμογή διαφέρει λίγο σε σχέση με αυτές που αναφέρθηκαν πιο πάνω καθώς δίνεται μεγαλύτερη έμφαση σε ένα κομμάτι που αφορά το project planning και όχι την δημιουργία ημερολογίου. Είναι φυσιολογικό λοιπόν να παρατηρούμε μια πιο περιορισμένη δυνατότητα όσον αφορά την δημιουργία εγγραφών αλλά την ύπαρξη πολλών επιπλέον δυνατοτήτων που αφορούν την διασύνδεση με κάποιο project ή κάποιο πίνακα εργασιών.

10.4 Cozi

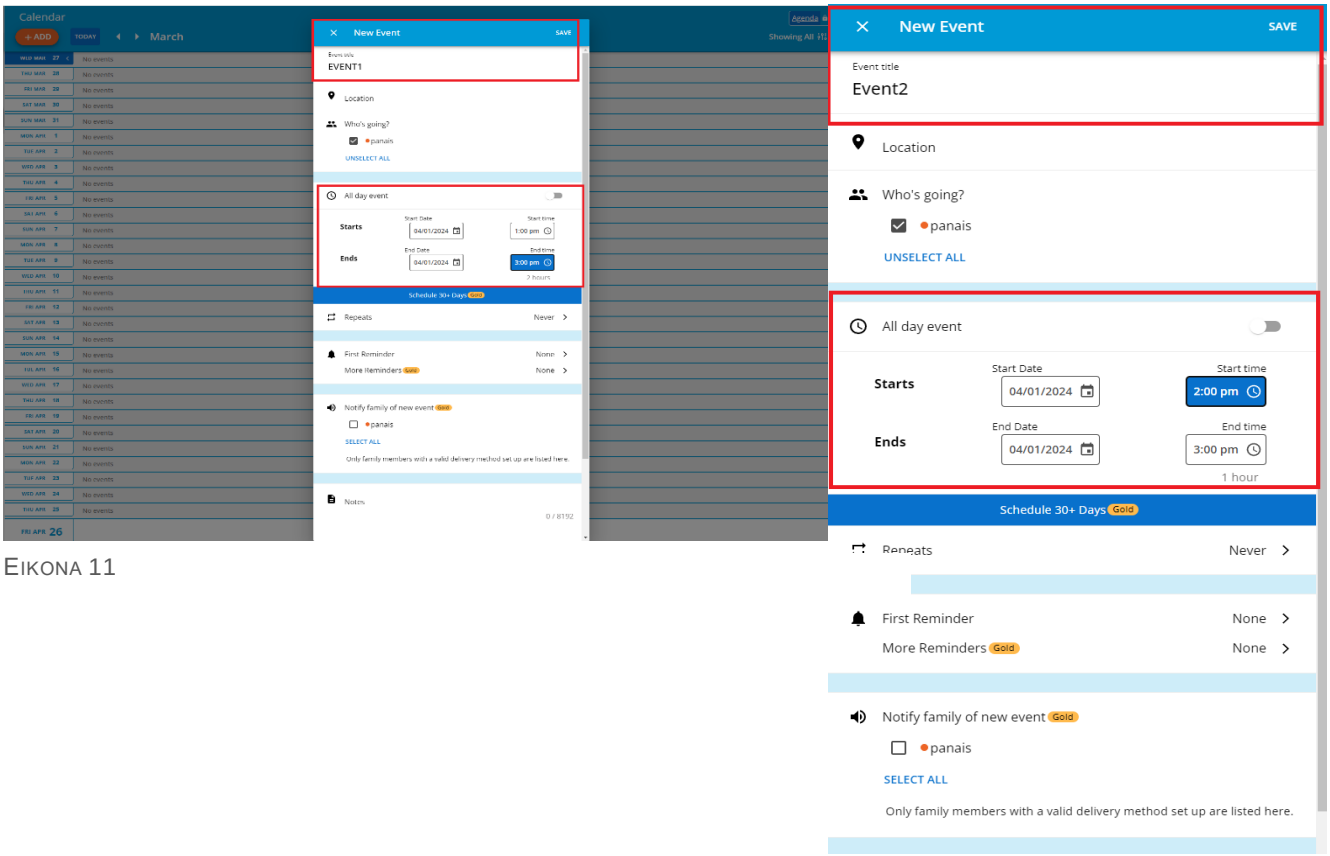
Το Cozi είναι ένα διαδικτυακό ημερολόγιο που βοηθά στον καθημερινό προγραμματισμό των δραστηριοτήτων. Εστιάζει στην δημιουργία ημερολόγιού που κοινοποιείται μεταξύ μιας ομάδας ανθρώπων και την οργάνωση των εγγράφων μεταξύ αυτών των ανθρώπων.

Το ημερολόγιο επίσης, επιτρέπει την εκχώρηση σε κάθε μέλος μιας ομάδας έναν κωδικό χρώματος και να παρακολουθείτε όλη τη δραστηριότητά του από ένα μέρος και να ξεχωρίζουν οι εγγραφές μεταξύ αυτών.

Τα καλύτερα χαρακτηριστικά του Cozi:

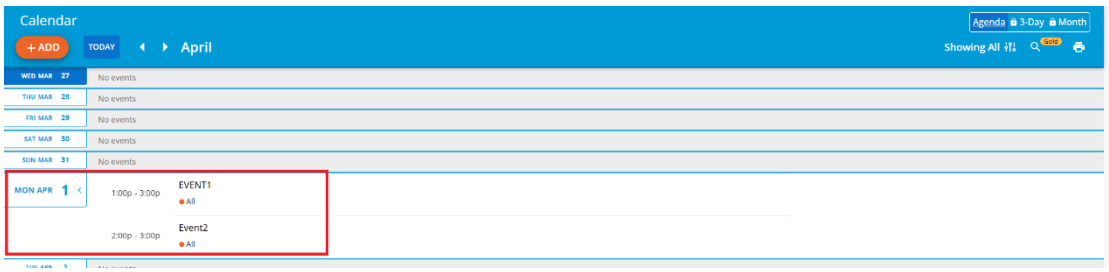
- Τα μέλη της ομάδας λαμβάνουν έναν κωδικό χρώματος
- Μπορεί να γίνει διαμοιρασμός από λίστες υποχρεώσεων ή δραστηριοτήτων.
- Λειτουργεί σε οποιαδήποτε συσκευή ή υπολογιστή

10.4.1 Σενάριο χρήσης



ΕΙΚΟΝΑ 11

ΕΙΚΟΝΑ 10



ΕΙΚΟΝΑ 12

Το Cozi δεν επιλέχθηκε τυχαία καθώς αποτελεί μια πιο απλουστευμένη μορφή ημερολογίου σε σχέση με τα προηγούμενα παραδείγματα. Το συγκεκριμένο πρόγραμμα δίνει μεγαλύτερη έμφαση στην χρήση του από μια ομάδα ανθρώπων και ο σκοπός του είναι να μπορεί να την οργανώσει.

Το σύστημα παρέχει έναν κεντρικό κωδικό, ο οποίος διαμοιράζεται μεταξύ όλων των μελών, για να είναι δυνατή η είσοδος στο ημερολόγιο αυτό. Όπως όμως έχουμε παρατηρήσει και στις προηγούμενες περιπτώσεις, όταν βρισκόμαστε στην περίπτωση που έχουμε χρονολογικές συμπτώσεις, το πρόγραμμα όχι μόνο δεν μας ενημερώνει αλλά και η ορατότητα που δίνει στον χρήστη είναι περιορισμένη (Εικόνα 10&11).

Οι εγγραφές τοποθετούνται η μία κάτω από την άλλη και η μονός τρόπος κάποιος να δει ότι αυτές συμπίπτουν χρονικά είναι να παρατηρήσει το ημερολόγιο πολύ προσεκτικά και συγκεκριμένα τις ώρες που τέθηκα αυτές (Εικόνα 12).

Παρατηρούμε λοιπόν ότι παρόλο που υπάρχουν πολλές παρόμοιες λύσεις διαθέσιμες, καμία δεν μας δίνει κάποιον τρόπο να επιλύουμε αυτόματα τυχόν συμπτώσεις μεταξύ των εγγραφών μας. Στις περισσότερες περιπτώσεις δεν λαμβάνουμε ούτε κάποια ειδοποίηση είτε με την μορφή popup μηνύματος είτε κάποιας οπτικής ενημερώσεις (αλλαγή χρώματος κτλ).

11 Σύντομη Περιγραφή

Το λογισμικό που έχει αναπτυχθεί έχει την μορφή παραθυρικής εφαρμογής. Η εφαρμογή αυτή δίνει την δυνατότητα εγγραφής και ύστερα σύνδεσης στους χρήστες έτσι ώστε ο κάθε ένας από αυτούς να έχει το προφίλ του και να αποθηκεύει τις δικές του εγγραφές.

Το πρόγραμμα επιτρέπει στον χρήστη να επιλέγει την συγκεκριμένη ημερομηνία, στο ημερολόγιο, στην οποία θέλει να προσθέσει μια εγγραφή και έπειτα του δίνει την δυνατότητα να προσθέσει αυτήν την εγγραφή σε ένα συγκεκριμένο χρονικό εύρος δίνοντας και έναν συντελεστή σημαντικότητας σε αυτήν. Επιπρόσθετα, η εφαρμογή δίνει την δυνατότητα διαγραφής αλλά και τροποποιήσεις των ήδη δημιουργημένων εγγραφών του χρήστη.

Η λύση που προσφέρεται είναι στην περίπτωση που δύο εγγραφές συμπίπτουν χρονικά. Σε αυτήν την περίπτωση η εφαρμογή αναγνωρίζει και κατηγοριοποιεί τις εγγραφές αυτές ανάλογα με το αν έπονται, ακολουθούν, περιέχουν (χρονολογικά) την εγγραφή ή περιέχονται (χρονολογικά) και ύστερα επιλύει την «σύγκρουση» βάση του συντελεστή σημαντικότητας αλλά και έπειτα από την επιβεβαίωση του χρήστη.

Στην περίπτωση που ο χρήστης δεν επιλέξει να λύσει αυτόματα αυτήν την συνύπαρξη των εγγράφων τότε το πρόγραμμα περνάει την καινούργια εγγραφή αλλά κρατάει και την παλιά στο χρονικό πλαίσιο που αυτές έχουν περαστεί εξαρχής από τον χρήστη.

12 Ανάλυση απαιτήσεων

Σε αυτό το κομμάτι θα αναλυθεί η καταγραφή των απαιτήσεων της εργασίας και θα δημιουργηθεί ένα αρχικό πλάνο όπου θα περιγράφονται οι λειτουργείες και οι δυνατότητες που θα έχει ο χρήστης στην εφαρμογή. Το σύστημα χρειάζεται κάποια εργαλεία για να εκτελέσει γρήγορα και εύκολα όλες τις απαιτούμενες διεργασίες.

Αρχικά, απαιτείται μια διαδικασία μέσα από την οποία το σύστημα θα αποθηκεύει διάφορα δεδομένα σχετικά με τους χρήστες αλλά και με τα διάφορα event που , τα οποία θα είναι διαθέσιμα για προβολή και επεξεργασία. Αυτό μας οδηγεί στην ανάγκη ύπαρξης μιας βάσης δεδομένων.

Στην συγκεκριμένη εφαρμογή χρησιμοποιείται μια βιβλιοθήκη που ονομάζεται sqflite η οποία υποστηρίζει τη δημιουργία βάσεων τύπου SQL.

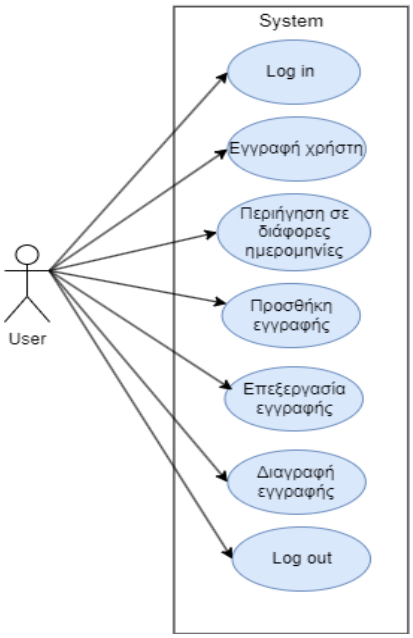
Το λογισμικό έχει την μορφή παραθυρικής εφαρμογής. Έχει αναπτυχθεί με τέτοιο τρόπο ώστε να τρέχει τοπικά σε οποιοδήποτε υπολογιστή που τρέχει λογισμικό windows 10 ή νεότερο. Η εφαρμογή δεν έχει την απαίτηση εγκατάστασης απλά την ύπαρξη του executable αρχείου και των συνοδευτικών βιβλιοθηκών (dll αρχεία).

12.1 Χρήστες

Το πρόγραμμα υποστηρίζει την ύπαρξη μόνο μιας κατηγορίας χρήστη, ο οποίος πρέπει να εγγραφεί και να συνδεθεί στην εφαρμογή. Ο χρήστης έχει πρόσβαση σε όλες τις δυνατότητες της εφαρμογής οι οποίες είναι οι εξής (Διάγραμμα 1):

- Log in
- Εγγραφή χρήστη
- Περιήγηση σε διάφορες ημερομηνίες
- Προσθήκη εγγραφής
- Επεξεργασία εγγραφής
- Διαγραφή εγγραφής
- Log out

Παρακάτω φαίνεται το διάγραμμα εργασιών:


System
Log in
Εγγραφή χρήστη
Περιήγηση σε διάφορες ημερομηνίες
Προσθήκη εγγραφής
Επεξεργασία εγγραφής
Διαγραφή εγγραφής
Log out
User

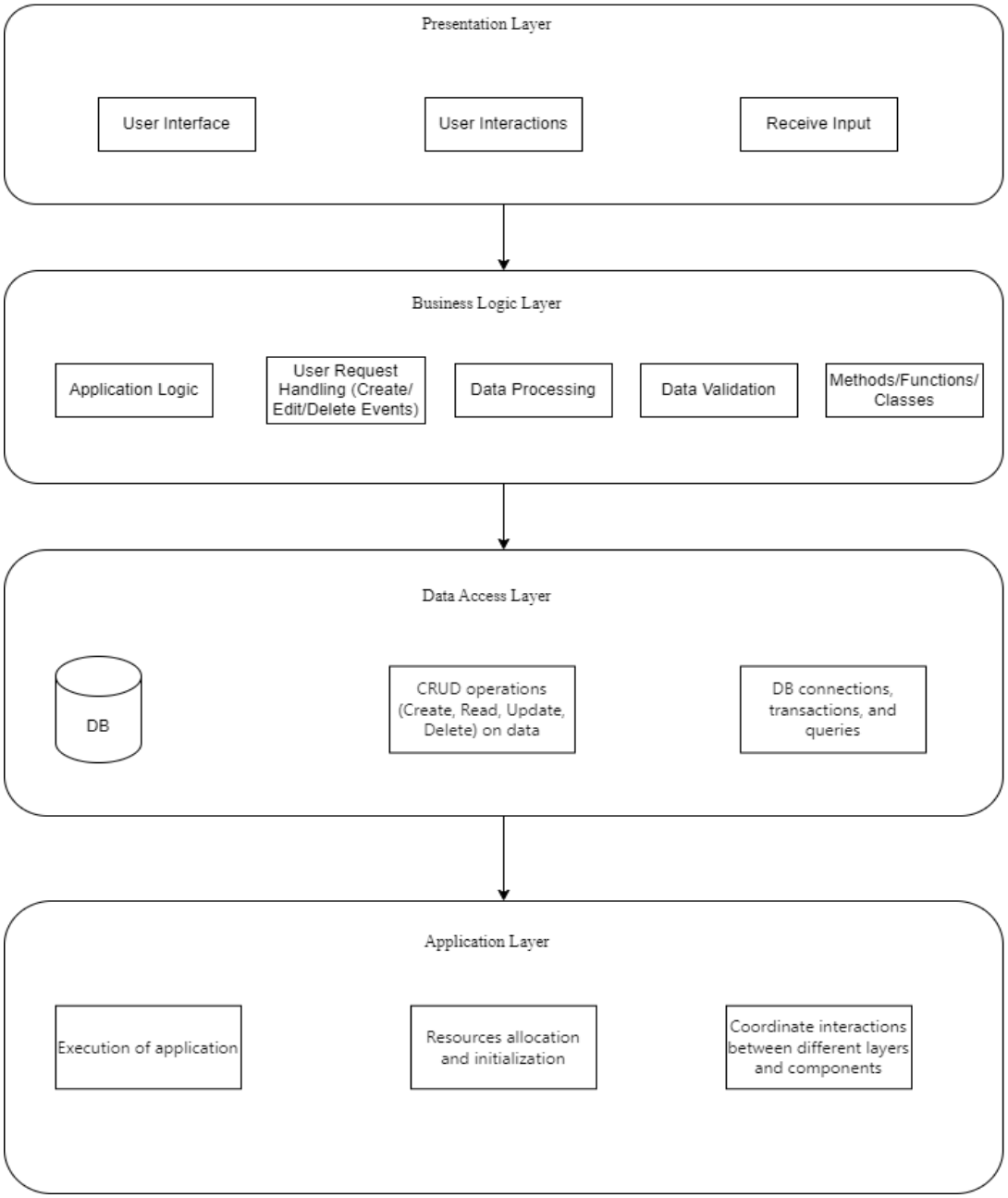
ΔΙΑΓΡΑΜΜΑ 1

13 Σχεδιασμός

13.1 Αρχιτεκτονική Συστήματος

Το σύστημα μας αποτελείτε από μια παραθυρική εφαρμογή η οποία είναι φτιαγμένη να τρέχει γρήγορα και αποτελεσματικά. Παρακάτω θα δούμε το διάγραμμα της αρχιτεκτονική του συστήματος μας που θα την χωρίσουμε στα εξής επίπεδα:

- Presentation Layer
- Business Logic Layer
- Data Access Layer
- Application Layer


Presentation Layer
User Interface
User Interactions
Receive Input
Business Logic Layer
Application Logic
User Request Handling (Create/ Edit/Delete Events)
Data Processing
Data Validation
Methods/Functions/ Classes
Data Access Layer
DB
CRUD operations (Create, Read, Update, Delete) on data
DB connections, transactions, and queries
Application Layer
Execution of application
Resources allocation and initialization
Coordinate interactions between different layers and components

ΔΙΑΓΡΑΜΜΑ 2

13.2 Τεχνολογίες που Χρησιμοποιήθηκαν

Ύστερα από την αρχιτεκτονική του συστήματος που αναφέρθηκε παραπάνω, οι ανάγκες για την υλοποίηση της παραπάνω εφαρμογής καλύφθηκαν με τις ακόλουθες τεχνολογίες και εργαλεία:

- Λειτουργικό σύστημα: Windows 11 Pro (OS Name: Microsoft Windows 11 Pro, OS Version: 23H2 N/A Build 22631.3296)
- Βάση δεδομένων: SQLite με την χρήση της βιβλιοθήκης Plugin sqflite 2.3.2
- Προγραμματιστικό Περιβάλλον: Android Studio Version: 2023.1.1.28
- Πλαίσιο υλοποίησης (framework): Flutter

13.3 Βάση δεδομένων

Το Flutter δεν περιέχει ενσωματωμένη υποστήριξη για δημιουργία/διαχείριση βάσεων δεδομένων, οπότε χρησιμοποιήθηκε για αυτό το σκοπό η βιβλιοθήκη sqflite η οποία υποστηρίζει τη δημιουργία βάσεων τύπου SQL. Παρέχει υποστήριξη εκτέλεσης raw κώδικα SQL σε συνδυασμό με δικές της μεθόδους παραγωγής κώδικα SQL με την παροχή των τιμών που θέλουμε να περιέχονται στο query μας.

Δημιουργήσαμε την κλάση MyPlannerDatabase που περιέχει το instance της βάσης SQL που μας παρέχει το sqflite και μεθόδους για τις διαδικασίες χρήστη και δραστηριοτήτων που θα χρειαστούμε(Εικόνα 13).

```
late final Database db;
Future<void> init() async {
  sqfliteFfiInit();
  databaseFactory = databaseFactoryFfi;
  final String databasesPath = await getDatabasesPath();
  final String path = join(databasesPath, 'my_planner.db');
  db = await openDatabase(
        path,
        onCreate: (Database db, int version) async {
          await db.execute('CREATE TABLE users (id INTEGER PRIMARY KEY AUTOINCREMENT, username TEXT, password TEXT)');
          await db.execute('CREATE TABLE events (id INTEGER PRIMARY KEY AUTOINCREMENT, userid INTEGER, title TEXT, details TEXT, importance INTEGER, date TEXT, time TEXT, duration INTEGER)');
        },/*...*/
        version: 2
      );
}
```

ΕΙΚΟΝΑ 13

Η βάση μας περιέχει 2 πίνακες. Έναν για αποθήκευση χρηστών και έναν για όλες τις δραστηριότητες(Εικόνα 14).

Ο πίνακας χρηστών αποτελείται από 3 στήλες

- id: Μοναδικός ακέραιος που αντιπροσωπεύει τον συγκεκριμένο χρήστη
- username: Όνομα χρήστη ως κείμενο
- password: Κωδικός χρήστη ως κείμενο

Πίνακας δραστηριοτήτων

- id: Μοναδικός ακέραιος για την δραστηριότητα
- userid: Το id του χρήστη-ιδιοκτήτη
- title: Κείμενο, τίτλος

- details: Κείμενο, ελεύθερες λεπτομέρειες
- importance: Ακέραιος, Κρισιμότητα δραστηριότητας
- date: Κείμενο, Ημερομηνία σε μορφή YYYY:MM:dd
- time: Κείμενο, Ώρα σε μορφή HH:mm
- duration: Κείμενο, σε μορφή HHmm

ΕΙΚΟΝΑ 14

13.4 Περιγραφή εργασιών

Προτού αναλύσουμε τις επιμέρους διεργασίες του συστήματος είναι σημαντικό να κάνουμε μια μικρή εισαγωγή για τις δυνατότητες του framework που χρησιμοποιήθηκε. Έτσι θα είναι πιο εύκολη η κατανόηση των διεργασιών που θα ακολουθήσουν.

Παρακάτω θα περιγράφουν κάποια βασικά στοιχεία του προγράμματος καθώς και του framework τα οποία χρησιμοποιούνται σε πολλά σημεία του κώδικα της εφαρμογής.

13.4.1 Dart/Flutter

Η εφαρμογή αναπτύχθηκε με την χρήση του Flutter, framework βασισμένο στη γλώσσα Dart.Η Dart αποτελεί “υπερσύνολο” (superset) της JavaScript, όμως είναι strongly typed και έχει αναπτυχθεί με έμφαση στη ταχύτατη δημιουργία αποδοτικών εφαρμογών και την εύκολη και

άμεση μεταγλώττιση τους σε μορφή κατάλληλη για συσκευές Android, iOS, Windows, Linux, Mac, ακόμη και σε μορφή ιστοσελίδας.

13.4.2 Widget

Το κεντρικό στοιχείο του Flutter είναι το Widget. Κάθε εφαρμογή σε Flutter αναπαρίσταται ως ένα δέντρο από Widgets (Widget Tree).

Για παράδειγμα, μια από τις οθόνες της εφαρμογής είναι ένα Widget και όλα τα περιεχόμενα στοιχεία της σελίδας όπως κουμπιά, κείμενα, εικόνες είναι επίσης Widgets. Στην αναπαράσταση ως δέντρο, ο κόμβος που αντιστοιχεί στην οθόνη αυτή, έχει ως απογόνους τους κόμβους που αντιστοιχούν στα περιεχόμενα widgets της οθόνης.

13.4.3 State

Κάθε Widget μπορεί να είναι Stateful, δηλαδή να έχουν οριστεί μεταβλητές δηλωμένες από τον προγραμματιστή, οι οποίες αντιπροσωπεύουν την τρέχουσα κατάσταση του. Το σύνολο των μεταβλητών αυτών ονομάζεται State και περιέχονται στο αντικείμενο State που ορίζουμε στο κάθε Widget ξεχωριστά.

Καλώντας την μέθοδο setState() του αντικειμένου State μετά από τυχόν αλλαγή τιμών των μεταβλητών, το flutter ξαναχτίζει το αντίστοιχο Widget και όλους τους απογόνους του, με βάση τις τρέχουσες τιμές των μεταβλητών του State του.

13.4.4 BuildContext

Είναι μια δομή που περιέχει πληροφορίες σχετικά με την θέση του χρήστη στο Widget Tree και είναι συνήθως απαραίτητη η παροχή της όταν γίνεται εναλλαγή μεταξύ οθονών ή άλλες ενέργειες μεταξύ widgets τα οποία δεν έχουν συγγένεια τύπου πρόγονος-απόγονος.

13.4.5 build(BuildContext)

Είναι η μέθοδος κάθε Widget η οποία εκτελεί την δημιουργία του και απαιτεί να δοθεί κάποιο build context για να εκτελεστεί (Συνήθως παρέχεται αυτόματα από τον γονιό του).

Ασύγχρονες μέθοδοι στην Dart

Τυπικά ο κώδικάς μας εκτελείται γραμμή προς γραμμή. Σε περίπτωση που κάποια διαδικασία χρειάζεται πολύ χρόνο να ολοκληρωθεί και δεν χρειαζόμαστε άμεσα το αποτέλεσμά της, για να αποφύγουμε την διακοπή εκτέλεσης του προγράμματος εξαιτίας της “βαριάς” διαδικασίας μπορούμε να ορίσουμε τον τύπο επιστροφής μεθόδου ή τον τύπο μεταβλητής σε Future. Το Future είναι ένας τύπος δεδομένου που μας επιτρέπει να εκτελέσουμε κώδικα παράλληλα, και μας επιστρέφει το αποτέλεσμά του όταν η κληθείσα διαδικασία ολοκληρωθεί.

πχ Όταν επικοινωνούμε με μια βάση δεδομένων ή με έναν διακομιστή στο διαδίκτυο, οι απαντήσεις τους δεν είναι άμεσες, απαιτούν κάποιο χρόνο για να απαντήσουν.

Σε αυτή την περίπτωση αντί να περιμένουμε μέχρι να πάρουμε την απάντηση και έπειτα να συνεχίσουμε την εκτέλεση του προγράμματος χρησιμοποιούμε την δομή Future και την μέθοδο της then(), ώστε να συνεχιστεί η λειτουργία του προγράμματος απρόσκοπτα και να διαβαστεί η απάντηση όταν αυτή είναι διαθέσιμη.

Λόγω της ύπαρξης των ασύγχρονων διαδικασιών και των future δομών είναι σημαντικό να αναφερθεί ότι ο κώδικας που παρουσιάζεται δεν απεικονίζει σειριακά τις διαδικασίες που συμβαίνουν σε αντίθεση με τα διαγράμματα που έχουν δημιουργηθεί

13.5 Δομή εφαρμογής

Κατά την εκτέλεση, αρχικοποιούμε μια παγκόσμια μεταβλητή currUser με την τιμή -1, υποδηλώνοντας πως δεν υπάρχει συνδεδεμένος χρήστης. Επιπλέον ορίζουμε την βάση δεδομένων μας σε μια παγκόσμια μεταβλητή plannerDB (Εικόνα 15).

```
int currUser = -1;
final plannerDB = MyPlannerDatabase();

Future<void> main() async {
  WidgetsFlutterBinding.ensureInitialized();
  await plannerDB.init();
  runApp(const MyPlannerApp());
}
```

ΕΙΚΟΝΑ 15

Το σημείο εισόδου του προγράμματος είναι η μέθοδος main() την οποία έχουμε ορίσει ως τύπου Future διότι μέσα της αρχικοποιούμε την βάση δεδομένων με ασύγχρονο τρόπο.

Στην main() καλούμε την μέθοδο WidgetsFlutterBinding.ensureInitialized() για να μπορούμε να χρησιμοποιήσουμε βασικές λειτουργίες του Flutter οι οποίες από προεπιλογή είναι διαθέσιμες αφότου κληθεί η μέθοδος runApp(). Έπειτα αρχικοποιούμε την βάση δεδομένων μας και προχωράμε στην εκκίνηση της εφαρμογής μας (runApp()) με παράμετρο ένα Widget τύπου MaterialApp (Εικόνα 16).

```
class MyPlannerApp extends StatelessWidget {
  const MyPlannerApp({super.key});

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      restorationScopeId: "Test",
      title: 'My Planner',
      debugShowCheckedModeBanner: false,
      theme: ThemeData(
        colorScheme: ColorScheme.fromSeed(seedColor: Colors.lightBlue.shade100),
        useMaterial3: true,
      ),  // ThemeData
      home: const LoginScreen(),
    );  // MaterialApp
  }
}
```

ΕΙΚΟΝΑ 16

Το κύριο Widget μας (αρχικός κόμβος δέντρου) είναι το MyPlannerApp το οποίο αρχικοποιούμε με κάποιες ρυθμίσεις που αφορούν την εμφάνιση και την ονομασία της εφαρμογής μαζί με το Widget που αποτελεί την αρχική οθόνη μας, το LoginScreen.

13.6 Αρχική οθόνη (Login Screen)

Στην αρχική οθόνη έχουμε μια δυναμική φόρμα εισαγωγής δεδομένων. Έχουμε ορίσει δύο είδη ενεργειών Login και Register:

```
final List<String> _actionTypes = ["Είσοδος", "Εγγραφή"];
int _action = 0;
```

ΕΙΚΟΝΑ 17

Με βάση ποια είναι η τρέχουσα επιλογή η φόρμα εμφανίζει τα αντίστοιχα πεδία και το κουμπί υποβολής καλεί την αντίστοιχη διαδικασία αυθεντικοποίησης ή εγγραφής (Εικόνα 17). Η τρέχουσα τιμή αλλάζει από τον χρήστη μέσω του κουμπιού “Πήγαινε στην σελίδα <άλλος τύπος σελίδας>”.

Τα πεδία εισαγωγής κειμένου έχουν το καθένα δικό τους TextEditingController που μας παρέχει το περιεχόμενο του πεδίου ως κείμενο(Εικόνα 18).

```
final _usernameController = TextEditingController();
```

ΕΙΚΟΝΑ 18

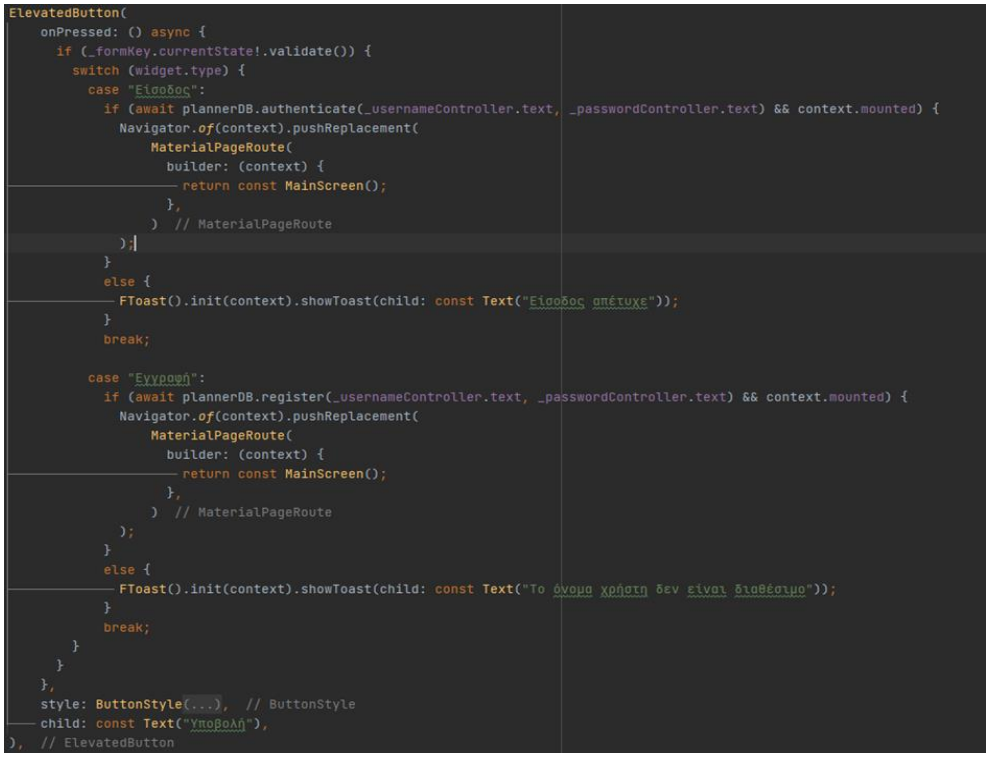
```
TextFormField(
  controller: _usernameController,
  decoration: const InputDecoration(...),  // InputDecoration
  validator: (value) {...},
),  // TextFormField
```

ΕΙΚΟΝΑ 19

Επιπλέον για κάθε πεδίο ορίζουμε μια διαδικασία validator στην οποία ελέγχουμε την εγκυρότητα του κειμένου στο πεδίο (Εικόνα 19). Στην συγκεκριμένη φόρμα ο μόνος περιορισμός που ορίσαμε είναι να μην είναι κενό το κείμενο, παρέχοντας το αντίστοιχο μήνυμα λάθους στον χρήστη (Εικόνα 20).

```
validator: (value) {
  if (value == null || value.isEmpty) {
    return 'Παρακαλώ εισάγετε';
  }
  return null;
},
```

ΕΙΚΟΝΑ 20

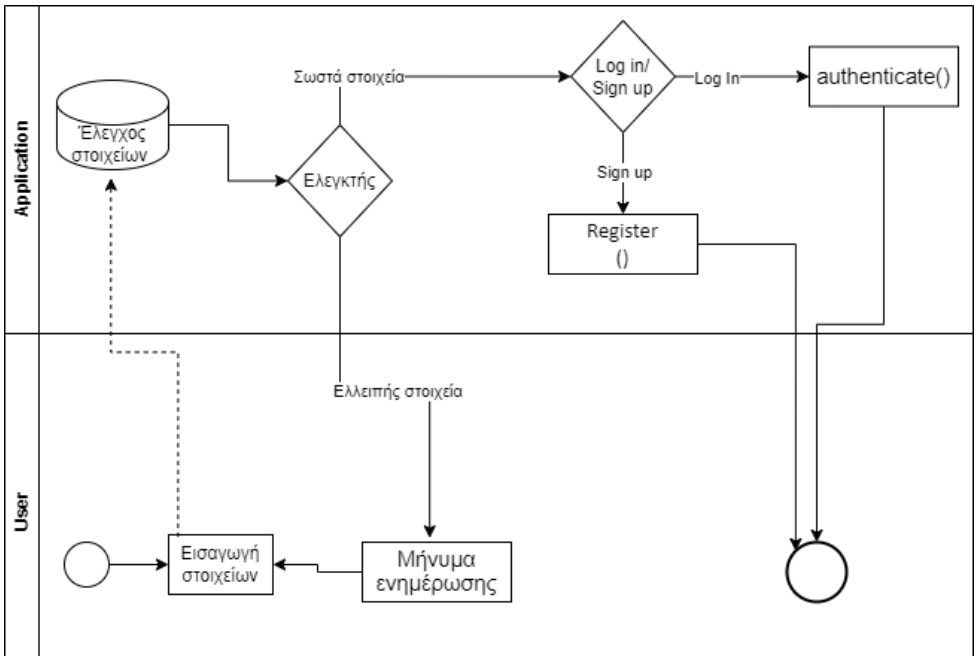


```
ElevatedButton(
    onPressed: () async {
      if (_formKey.currentState!.validate()) {
        switch (widget.type) {
          case "Είσοδος":
            if (await plannerDB.authenticate(_usernameController.text, _passwordController.text) && context.mounted) {
              Navigator.of(context).pushReplacement(
                  MaterialPageRoute(
                    builder: (context) {
                      return const MainScreen();
                    },
                  )  // MaterialPageRoute
              );
            }
            else {
              FToast().init(context).showToast(child: const Text("Είσοδος απέτυχε"));
            }
            break;

          case "Εγγραφή":
            if (await plannerDB.register(_usernameController.text, _passwordController.text) && context.mounted) {
              Navigator.of(context).pushReplacement(
                  MaterialPageRoute(
                    builder: (context) {
                      return const MainScreen();
                    },
                  )  // MaterialPageRoute
              );
            }
            else {
              FToast().init(context).showToast(child: const Text("Το όνομα χρήστη δεν είναι διαθέσιμο"));
            }
            break;
        }
      }
    },
    style: ButtonStyle(...),  // ButtonStyle
    child: const Text("Υποβολή"),
),  // ElevatedButton
```

ΕΙΚΟΝΑ 21

Τέλος, το κουμπί υποβολής εκτελεί τα validators των πεδίων εισαγωγής (Εικόνα 21). Εάν κάποιο κριτήριο δεν πληρείται, δεν απλώς εμφανίζεται το αντίστοιχο μήνυμα. Εάν πληρούνται τα κριτήρια μας συνεχίζουμε στην κλήση αυθεντικοποίησης χρήστη ή εγγραφή νέου χρήστη αντίστοιχα (Διάγραμμα 3).

Παρακάτω απεικονίζεται το διάγραμμα της διαδικασίας του LogIn:



ΔΙΑΓΡΑΜΜΑ 3

13.7 Εγγραφή χρηστών και αυθεντικοποίηση χρηστών

13.7.1 Εγγραφή

Για την εγγραφή ενός νέου χρήστη χρησιμοποιείται η μέθοδος register() του MyPlannerDatabase (Εικόνα 22).

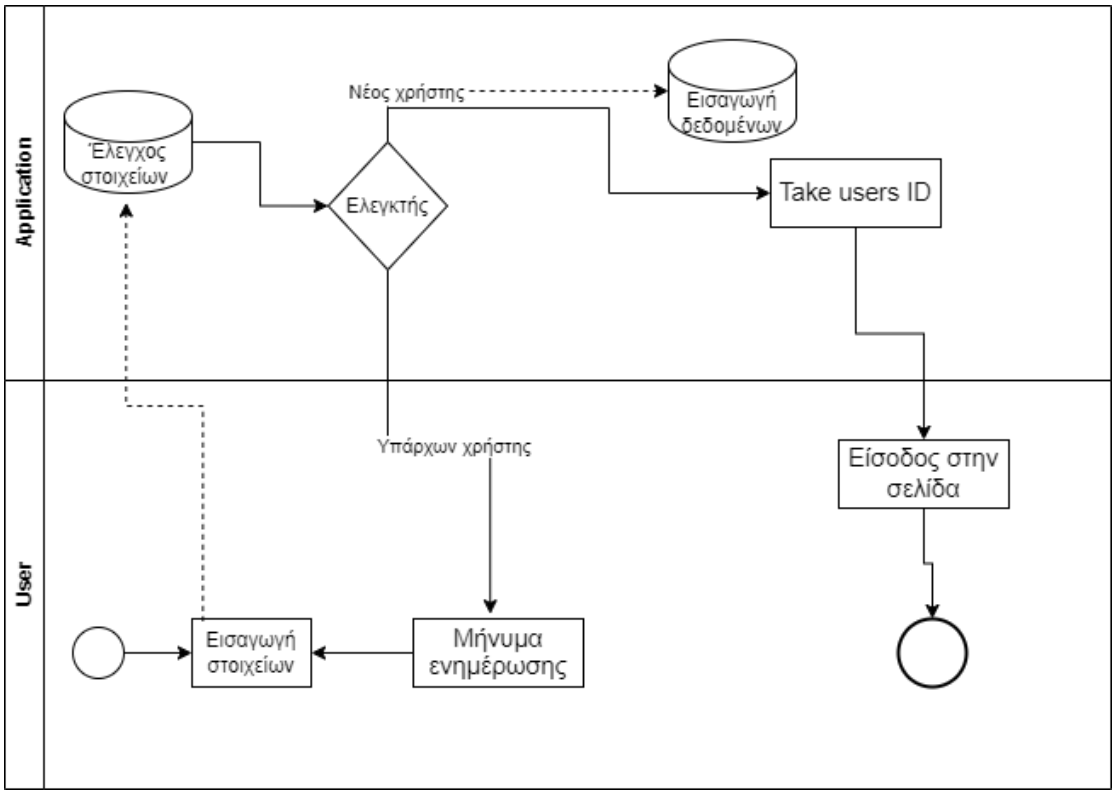
Πρώτα ζητάμε από τη βάση να μας επιστρέψει τυχόν χρήστη με το όνομα που έχουμε δώσει. Εάν υπάρχει ήδη χρήστης με αυτό το όνομα, επιστρέφουμε false.

Εάν δεν υπάρχει χρήστης, εισάγουμε τα στοιχεία του χρήστη στη βάση, αλλάζουμε την τιμή της global μεταβλητής στο id του νέου χρήστη και επιστρέφουμε true (Διάγραμμα 4).

```
Future<bool> register(String username, String password) async {
  // check if username is already used
  final bool usernameAvailable = (await db.query('users', where: 'username = ?', whereArgs: [username])).isEmpty;
  if (usernameAvailable) {
    await db.insert('users', {'username': username, 'password': password});
    currUser = (await db.query('users', where: 'username = ? AND password = ?', whereArgs: [username, password]))[0]['id'] as int;
    return true;
  }
  return false;
}
```

ΕΙΚΟΝΑ 22

Παρακάτω απεικονίζεται το διάγραμμα της διαδικασίας του Register:



ΔΙΑΓΡΑΜΜΑ 4

13.7.2 Αυθεντικοποίηση

Για την αυθεντικοποίηση καλούμε την authenticate() (Εικόνα 23).

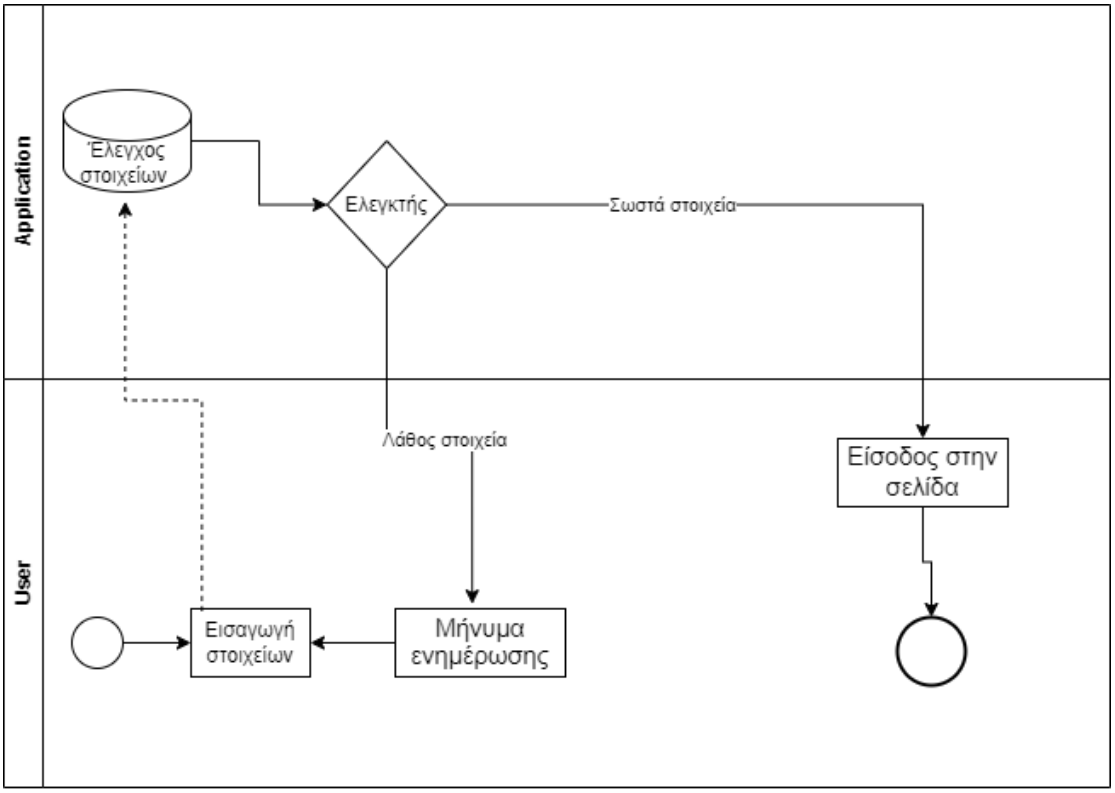
Ζητάμε από την βάση να μας βρει χρήστη με τα στοιχεία που δίνονται και εάν βρεθεί, αλλάζουμε την τιμή της global μεταβλητής στο id του νέου χρήστη και επιστρέφουμε true (Διάγραμμα 5).

```
Future<bool> authenticate(String username, String password) async {
  if ((await db.query('users', where: 'username = ? AND password = ?', whereArgs: [username, password])).isNotEmpty) {
    currUser = (await db.query('users', where: 'username = ? AND password = ?', whereArgs: [username, password]))[0]['id'] as int;
    return true;
  }
  return false;
}
```

ΕΙΚΟΝΑ 23

Παρακάτω απεικονίζεται το διάγραμμα της διαδικασίας του Authenticate:

ΔΙΑΓΡΑΜΜΑ 5



13.8 Δραστηριότητα (Event)

Την μεμονωμένη δραστηριότητα αντιπροσωπεύει η κλάση Event. Παρακάτω περιγράφονται οι μεταβλητές και οι μέθοδοι της κλάσης:

Μεταβλητές:

- id: βλ. βάση δεδομένων
- userid: βλ. βάση δεδομένων
- title: βλ. βάση δεδομένων
- details: βλ. βάση δεδομένων
- importance: βλ. βάση δεδομένων
- dateString: βλ. βάση δεδομένων
- startTimeString: βλ. βάση δεδομένων
- duration: διάρκεια δραστηριότητας ως αντικείμενο Duration
- durationString: βλ. βάση δεδομένων
- startDateTime: ημερομηνία εκκίνησης ως αντικείμενο DateTime
- endDateTime: ημερομηνία τέλους ως αντικείμενο DateTime
- endDateString: ημερομηνία τέλους σε μορφή κειμένου

Μέθοδοι:

- fromMap(): δημιουργεί και επιστρέφει ένα νέο Event με το δοσμένο Map
- toMap(): επιστρέφει το Event σε μορφή Map

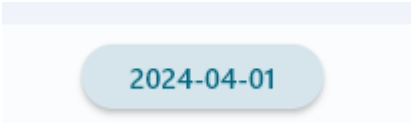
- toString(): επιστρέφει το Event σε μορφή κειμένου
- getStartTimeOfDay(): επιστρέφει την ώρα εκκίνησης σε μορφή TimeOfDay
- updateGeneratedDateTimes():Ενημερώνει τις τιμές των μεταβλητών ημερομηνίας και ώρας σε περίπτωση που έχουν αλλάξει, καθώς κάποιες από αυτές είναι παραγόμενες από εμάς, μόνο για διευκόλυνση και χρήση κατα την εκτέλεση του προγράμματος και δεν αποθηκεύονται στην βάση.
- updateInDatabase(): Ενημερώνει τα στοιχεία της δραστηριότητας στην βάση χρησιμοποιώντας την μέθοδο plannerDB.updateEvent()
- overlapsTodaysEvents():Επιστρέφει λογική τιμή που αντιστοιχεί στο αν η δραστηριότητα μας συμπέφτει με οποιαδήποτε άλλη την συγκεκριμένη ημερομηνία και ώρα
- resolveTodaysOverlaps(): Επιλύει τις συμπτώσεις της δραστηριότητας
- Βοηθητικοί τύποι δεδομένων
- DateTime: Ημερομηνία σε μορφή ISO 8601, με πληθώρα βοηθητικών μεθόδων
- Duration: Χρονικό διάστημα σε οποιαδήποτε μονάδα μέτρησης χρόνου
- TimeOfDay: Ώρα ημέρας. Αποθηκεύει την ώρα σε 2 ακέραιες μεταβλητές, μια για τις ώρες και μια για τα λεπτά
- Map: Συλλογή ζευγαριών κλειδιού-τιμή

13.9 Προβολή δραστηριοτήτων (Κύρια οθόνη)

Στην κύρια οθόνη, έχουμε 4 βασικά widgets:

- Τον επιλογέα ημερομηνίας
- Την λίστα προβολής δραστηριοτήτων
- Το πλήκτρο αποσύνδεσης χρήστη
- Το πλήκτρο δημιουργίας νέας δραστηριότητας

Ο επιλογέας ημερομηνίας είναι ένα ElevatedButton με κείμενο την επιλεγμένη ημερομηνία, το οποίο όταν πατάει ο χρήστης, εμφανίζει popup επιλογής ημερομηνίας (Εικόνα 24) καλώντας την μέθοδο showDatePicker() που μας παρέχει το Flutter.



ΕΙΚΟΝΑ 24

Το πλήκτρο δημιουργίας νέας δραστηριότητας μας πηγαίνει στην οθόνη δημιουργίας event (Εικόνα 25).

ΕΙΚΟΝΑ 25

Η λίστα δραστηριοτήτων προβάλλει για κάθε δραστηριότητα της επιλεγμένης ημερομηνίας ένα widget τύπου κάρτα το οποίο περιέχει τις πληροφορίες της δραστηριότητας, συνοδευόμενες από ένα πλήκτρο το οποίο μας πηγαίνει στην οθόνη επεξεργασίας για την συγκεκριμένη δραστηριότητα και ένα πλήκτρο διαγραφής (Εικόνα 26).

ΕΙΚΟΝΑ 26

Το πλήκτρο αποσύνδεσης χρήστη (Εικόνα 27) μας επιστρέφει στην οθόνη εισόδου.

ΕΙΚΟΝΑ 27

14 Δημιουργία και επεξεργασία δραστηριότητας

Η φόρμα δημιουργίας/επεξεργασίας δραστηριότητας είναι μια φόρμα όμοια με εκείνη της εισόδου χρήστη. Κάθε πεδίο εισόδου αποθηκεύει τις τιμές που εισάγονται σε αντίστοιχες μεταβλητές και κατά την υποβολή της φόρμας ελέγχεται η εγκυρότητα τους (Εικόνα 28).

Σε περίπτωση επεξεργασίας δραστηριότητας η φόρμα αρχικοποιείται με τα ήδη υπάρχοντα δεδομένα της δραστηριότητας (Εικόνα 29).

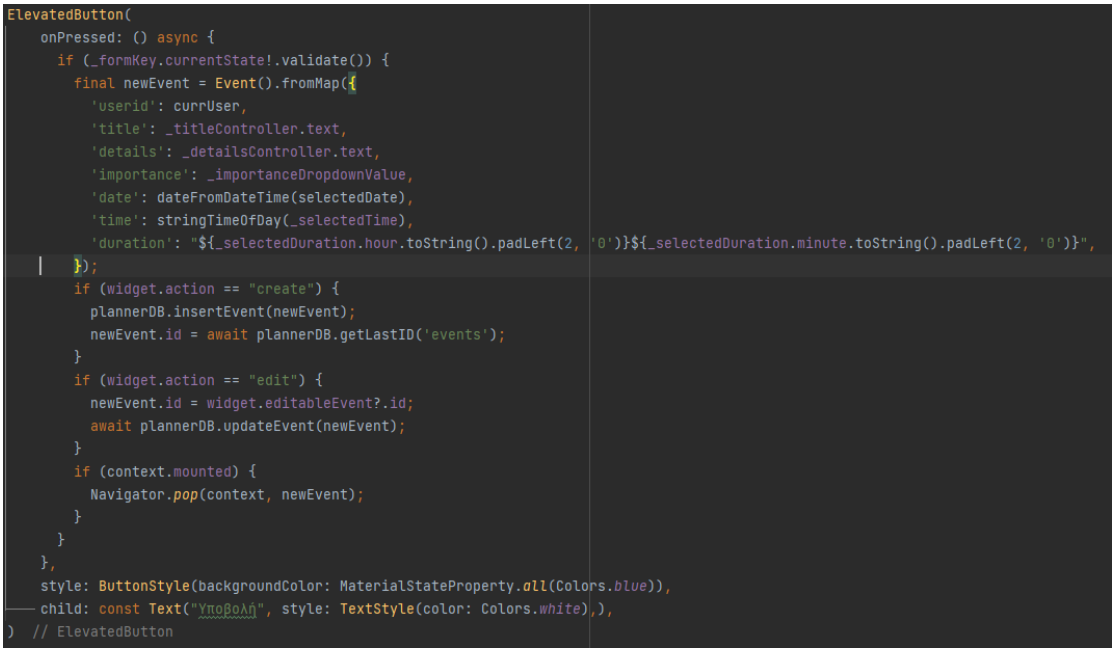
```
class EventScreen extends StatefulWidget {
  const EventScreen({super.key, required this.actionType, required this.context, this.editableEvent});

  final String actionType;
  final Event? editableEvent;
  final BuildContext context;
  @override
  State<EventScreen> createState() => _EventScreenState();
}
```

ΕΙΚΟΝΑ 28

```
class _EventFormState extends State<EventForm> {
  final _formKey = GlobalKey<FormState>();
  final _titleController = TextEditingController();
  final _detailsController = TextEditingController();

  TimeOfDay _selectedTime = TimeOfDay.now();
  TimeOfDay _selectedDuration = const TimeOfDay(hour: 1, minute: 0);

  var _importanceDropdownValue = 1;

  @override
  void initState() {
    super.initState();
    if (widget.action == "edit") {
      fillForm();
    }
  }

  void fillForm() {
    _titleController.text = widget.editableEvent!.title;
    _detailsController.text = widget.editableEvent!.details;

    _selectedTime = widget.editableEvent?.getStartTimeOfDay() ?? TimeOfDay.now();

    _importanceDropdownValue = widget.editableEvent!.importance;
    _selectedDuration = TimeOfDay(hour: widget.editableEvent!.duration.inHours, minute: widget.editableEvent!.duration.inMinutes % 60);
  }
```

ΕΙΚΟΝΑ 29

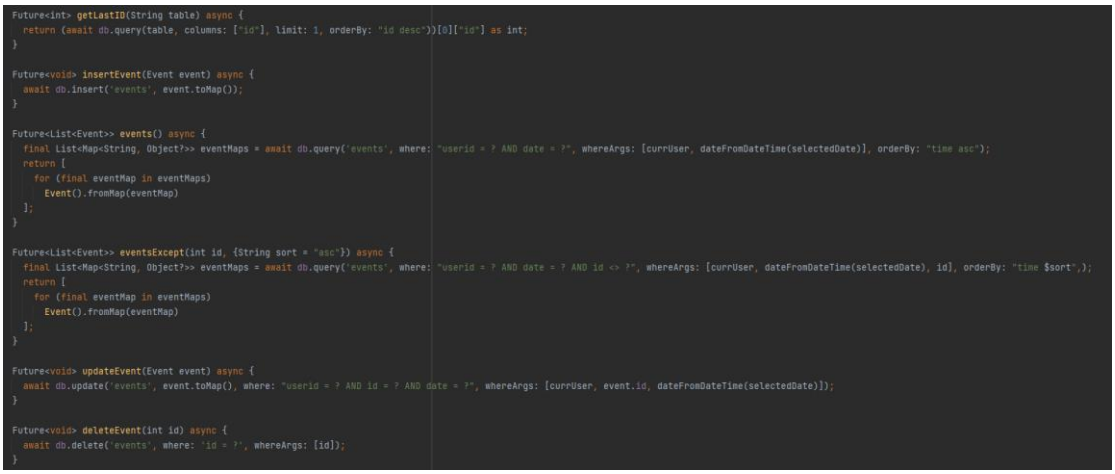
Εάν η υποβολή είναι επιτυχής, δημιουργούμε ένα νέο instance αντικειμένου Event με τις δοθείσες τιμές (Εικόνα 30) και έπειτα το δίνουμε στην βάση δεδομένων για αποθήκευση/ενημέρωση (Διάγραμμα 6).



```
ElevatedButton(
    onPressed: () async {
      if (_formKey.currentState!.validate()) {
        final newEvent = Event().fromMap({
          'userid': currUser,
          'title': _titleController.text,
          'details': _detailsController.text,
          'importance': _importanceDropdownValue,
          'date': dateFromDateTime(selectedDate),
          'time': stringTimeOfDay(_selectedTime),
          'duration': "${_selectedDuration.hour.toString().padLeft(2, '0')}${_selectedDuration.minute.toString().padLeft(2, '0')}",
        });
        if (widget.action == "create") {
          plannerDB.insertEvent(newEvent);
          newEvent.id = await plannerDB.getLastID('events');
        }
        if (widget.action == "edit") {
          newEvent.id = widget.editableEvent?.id;
          await plannerDB.updateEvent(newEvent);
        }
        if (context.mounted) {
          Navigator.pop(context, newEvent);
        }
      }
    },
    style: ButtonStyle(backgroundColor: MaterialStateProperty.all(Colors.blue)),
    child: const Text("Υποβολή", style: TextStyle(color: Colors.white),),
)  // ElevatedButton
```

Εικονα 30

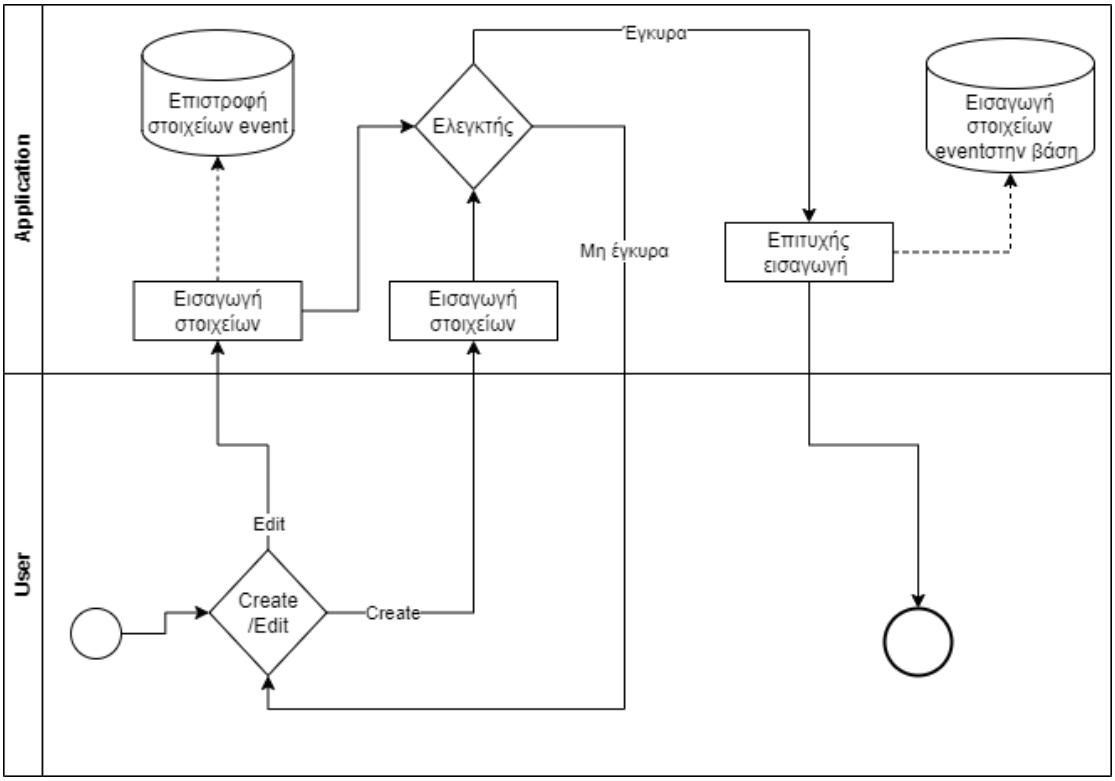
Οι μέθοδοι της βάσης δέχονται ως παράμετρο ένα αντικείμενο Event και χρησιμοποιούν τη μέθοδο του *Event.toMap()* (Εικόνα 31) για να το μετατρέψουν σε μια συλλογή κλειδιών-τιμών όπου το κάθε κλειδί αντιστοιχεί στο όνομα του column της βάσης στο οποίο αντιστοιχεί η τιμή του.



```
Future<int> getLastID(String table) async {
  return (await db.query(table, columns: ["id"], limit: 1, orderBy: "id desc"))[0]["id"] as int;
}

Future<void> insertEvent(Event event) async {
  await db.insert('events', event.toMap());
}

Future<List<Event>> events() async {
  final List<Map<String, Object?>> eventMaps = await db.query('events', where: "userid = ? AND date = ?", whereArgs: [currUser, dateFromDateTime(selectedDate)], orderBy: "time asc");
  return [
    for (final eventMap in eventMaps)
      Event().fromMap(eventMap)
  ];
}

Future<List<Event>> eventsExcept(int id, {String sort = "asc"}) async {
  final List<Map<String, Object?>> eventMaps = await db.query('events', where: "userid = ? AND date = ? AND id <> ?", whereArgs: [currUser, dateFromDateTime(selectedDate), id], orderBy: "time $sort",);
  return [
    for (final eventMap in eventMaps)
      Event().fromMap(eventMap)
  ];
}

Future<void> updateEvent(Event event) async {
  await db.update('events', event.toMap(), where: "userid = ? AND id = ? AND date = ?", whereArgs: [currUser, event.id, dateFromDateTime(selectedDate)]);
}

Future<void> deleteEvent(int id) async {
  await db.delete('events', where: 'id = ?', whereArgs: [id]);
}
```

Εικονα 31

Παρακάτω απεικονίζεται το διάγραμμα της διαδικασίας της δημιουργίας και επεξεργασίας εγγραφων:



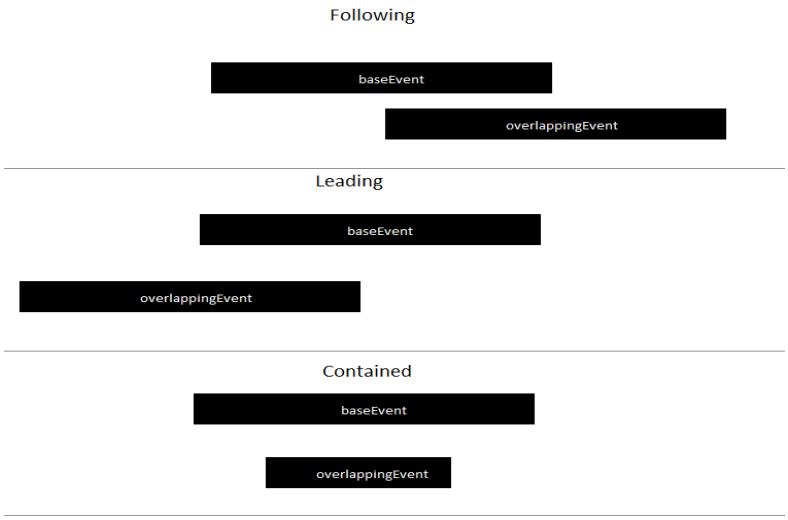
ΔΙΑΓΡΑΜΜΑ 6

14.1 Χρονολογικές συμπτώσεις

Για τον σκοπό εύρεσης και επίλυσης συμπτώσεων, δημιουργούμε ένα νέο αντικείμενο Overlap (Διάγραμμα 7). Τα χαρακτηριστικά του είναι τα εξής:

- baseEvent: Event το οποίο αποτελεί το επίκεντρο του ελέγχου μας, δεν σκοπεύουμε να το επεξεργαστούμε σε καμία περίπτωση.
- overlappingEvent: Event το οποίο ενδέχεται να συμπίπτει χρονικά με το baseEvent. Δύναται επεξεργασίας σε περίπτωση σύμπτωσης.
- isValid: λογική τιμή που αντιπροσωπεύει την εγκυρότητα της σύμπτωσης

- type: τύπος σύμπτωσης (μια από τις ακόλουθες)



ΔΙΑΓΡΑΜΜΑ 7

```
Overlap(this.baseEvent, this.overlappingEvent) {
  bool isFollowing =  overlappingEvent.startDateTime.compareTo(baseEvent.startDateTime) >= 0
      && overlappingEvent.startDateTime.compareTo(baseEvent.endDateTime) <= 0;
  bool isContained = isFollowing
      && overlappingEvent.endDateTime.isBefore(baseEvent.endDateTime);
  bool isLeading = overlappingEvent.startDateTime.isBefore(baseEvent.startDateTime)
      && overlappingEvent.endDateTime.isAfter(baseEvent.startDateTime);
  bool isContaining = isLeading
      && overlappingEvent.endDateTime.isAfter(baseEvent.endDateTime);
  if (isFollowing || isContained) {
    type = "following";
    isValid = true;
  }
  if (isLeading || isContaining) {
    type = "leading";
    isValid = true;
  }
```

ΕΙΚΟΝΑ 32

Για απλοποίηση της υλοποίησης αντιμετωπίζουμε την περίπτωση contained ως following και την περίπτωση containing ως leading (Εικόνα 32).

14.1.1 Εύρεση

Για την εύρεση των συμπτώσεων έχουμε προσθέσει στο αντικείμενο Event την μέθοδο overlapsTodaysEvents(), η οποία για κάθε άλλο event της ημέρας, ελέγχει εάν το Overlap μεταξύ του συγκεκριμένου Event (baseEvent) και του event της ημέρας (overlappingEvent) είναι έγκυρο. Εάν τουλάχιστον ένα τέτοιο event βρεθεί, επιστρέφεται η τιμή true (Εικόνα 33).

```
/// Returns true if at least one of the rest of today's event overlap with this event
Future<bool> overlapsTodaysEvents({bool forced = false}) async {
  // Get all today's events except this, because we don't want to check for overlap against itself
  List<Event> otherEvents = await plannerDB.eventsExcept(id!);
  for(int i = 0; i < otherEvents.length; i++) {
    if (Overlap(this, otherEvents[i]).isValid) {
      return true;
    }
  }
  return false;
}
```

ΕΙΚΟΝΑ 33

14.1.2 Επίλυση

Ακολουθώντας την χρήση της overlapsTodaysEvents() εάν ο χρήστης θελήσει να επιλυθούν οι συμπτώσεις, χρησιμοποιούμε της μέθοδο resolveTodaysOverlaps() (Εικόνα 34) η οποία επιλύει μια-προς-μια τις συμπτώσεις, με τη βοήθεια του Overlap, άλλα και με την χρήση δύο άλλως μεθόδων των _resolve() και resolve() (Εικόνα 35) οι οποίες ελέγχουν την σημαντικότητα των event που κάνουν το overlap, έτσι ώστε να ξέρουμε ποιο event θα μετακινηθεί, καθώς και χρονική αλληλουχία των event και προσθέτουν το νέο event στην κατάλληλη ώρα (Διάγραμμα 8).

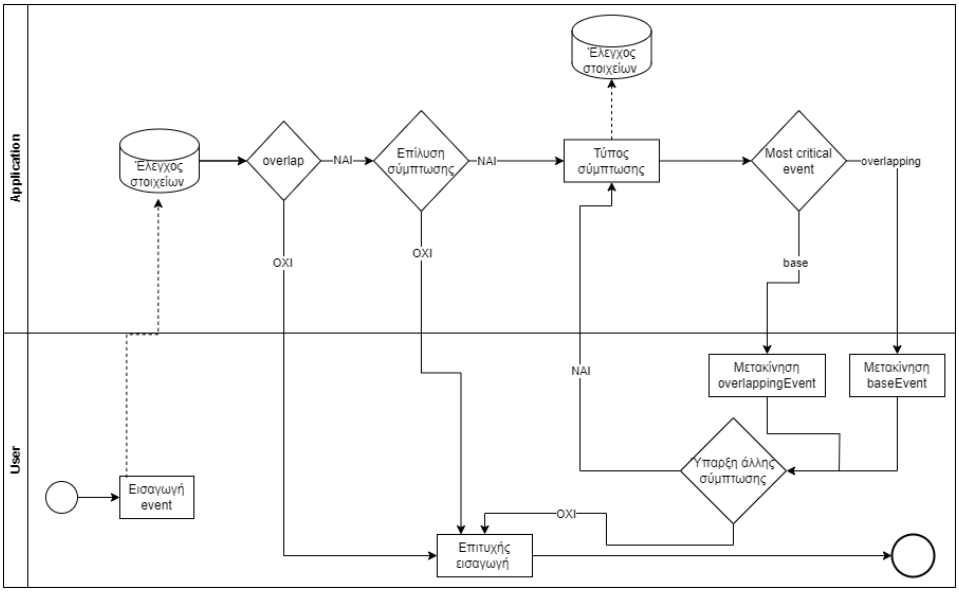
```dart
Future<void> resolveTodaysOverlaps({bool forced = false}) async {
  List<Event> ascEvents = await plannerDB.eventsExcept(id!, sort: "asc");
  Duration followingOffset = Duration.zero;
  for (var event in ascEvents) {
    final overlap = Overlap(this, event);
    if (overlap.isValid && overlap.type == "following") {
      Duration offset = event.startDateTime.difference(startDateTime);
      overlap.resolve(forced: forced, offset: followingOffset);
      followingOffset += offset;
    }
  }

  List<Event> descEvents = await plannerDB.eventsExcept(id!, sort: "desc");
  Duration leadingOffset = Duration.zero;
  for (var event in descEvents) {
    final overlap = Overlap(this, event);
    if (overlap.isValid && overlap.type == "leading") {
      Duration offset = event.startDateTime.difference(startDateTime);
      overlap.resolve(forced: forced, offset: leadingOffset);
      leadingOffset += offset;
    }
  }
}
```

ΕΙΚΟΝΑ 34

```dart
void resolve({bool forced = false, Duration offset = Duration.zero}) {
  if (forced || baseEvent.importance >= overlappingEvent.importance) {
    _resolve(offset);
  }
}

void _resolve(Duration offset) {
  if (type == "following") {
    overlappingEvent.startDateTime = baseEvent.endDateTime.add(offset);
  }
  if (type == "leading") {
    overlappingEvent.startDateTime = baseEvent.startDateTime.subtract(overlappingEvent.duration + offset);
  }
  overlappingEvent.updateGeneratedDateTimes();
  overlappingEvent.updateInDatabase();
}
}
```

ΕΙΚΟΝΑ 35

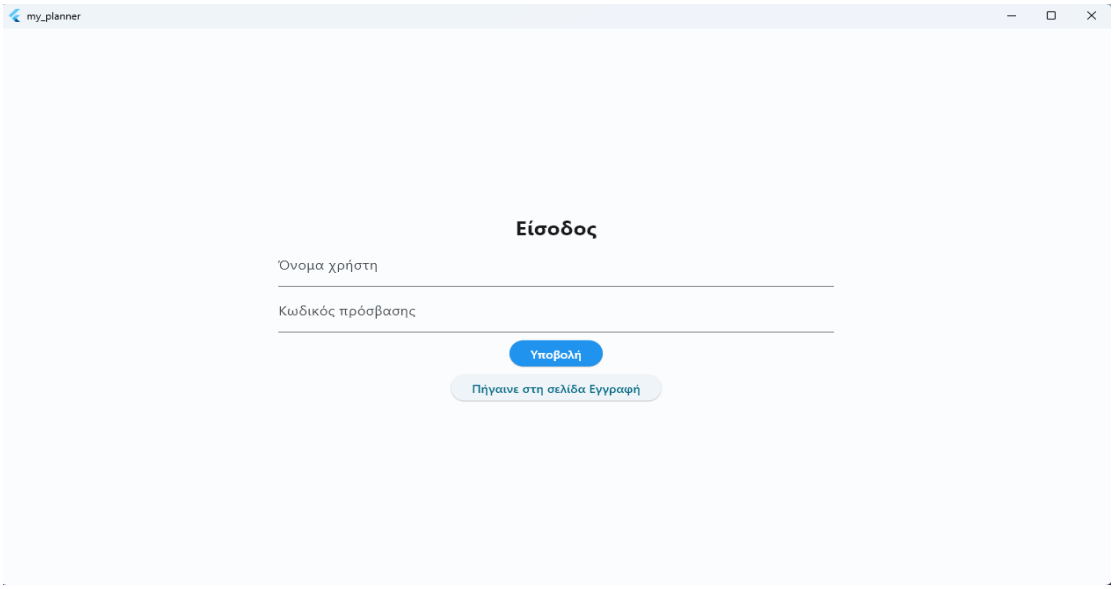
Παρακάτω απεικονίζεται το διάγραμμα της διαδικασίας του της επίλυσης συμπτώσεων:


Application
User
Έλεγχος στοιχείων
overlap
NAI
Επίλυση σύμπτωσης
NAI
Τύπος σύμπτωσης
Έλεγχος στοιχείων
Most critical event
overlapping
OXI
OXI
base
NAI
Μετακίνηση overlappingEvent
Μετακίνηση baseEvent
Υπαρξη άλλης σύμπτωσης
Εισαγωγή event
OXI
Επιτυχής εισαγωγή

ΔΙΑΓΡΑΜΜΑ 8

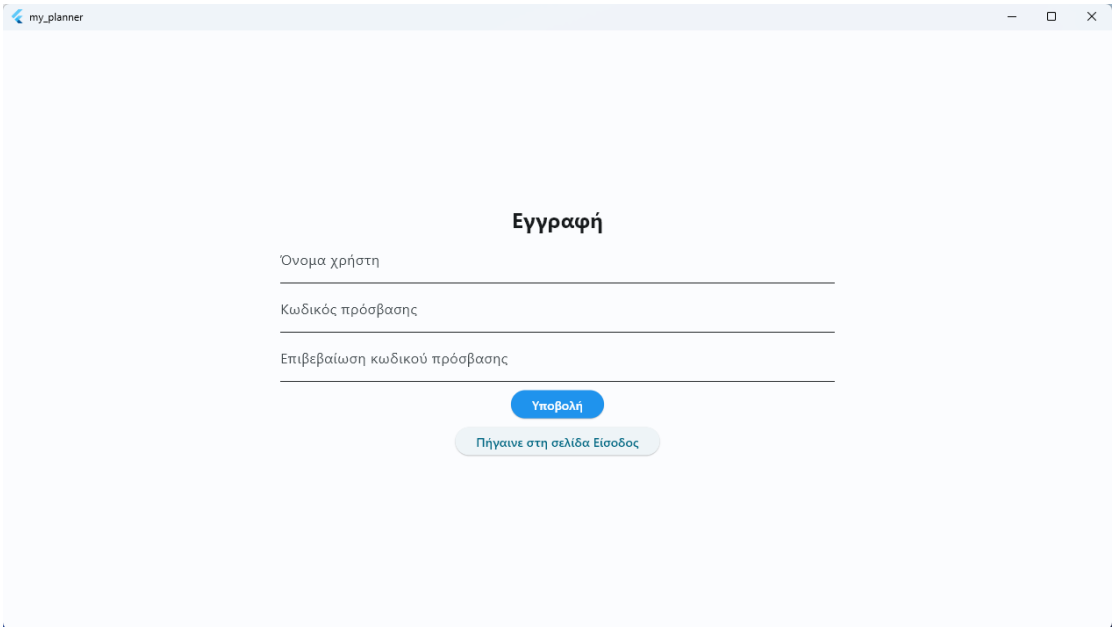
15 Παρουσίαση Σεναρίου Χρήσης

Είσοδος στην εφαρμογή:



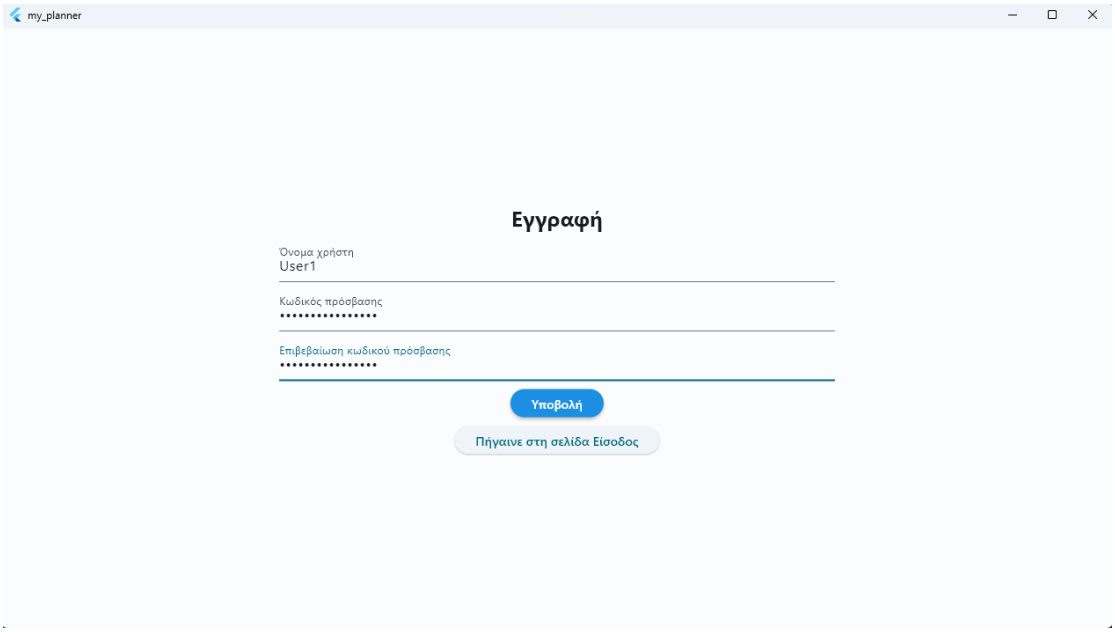
ΕΙΚΟΝΑ 36

Εγγραφή του χρήστη:



ΕΙΚΟΝΑ 37

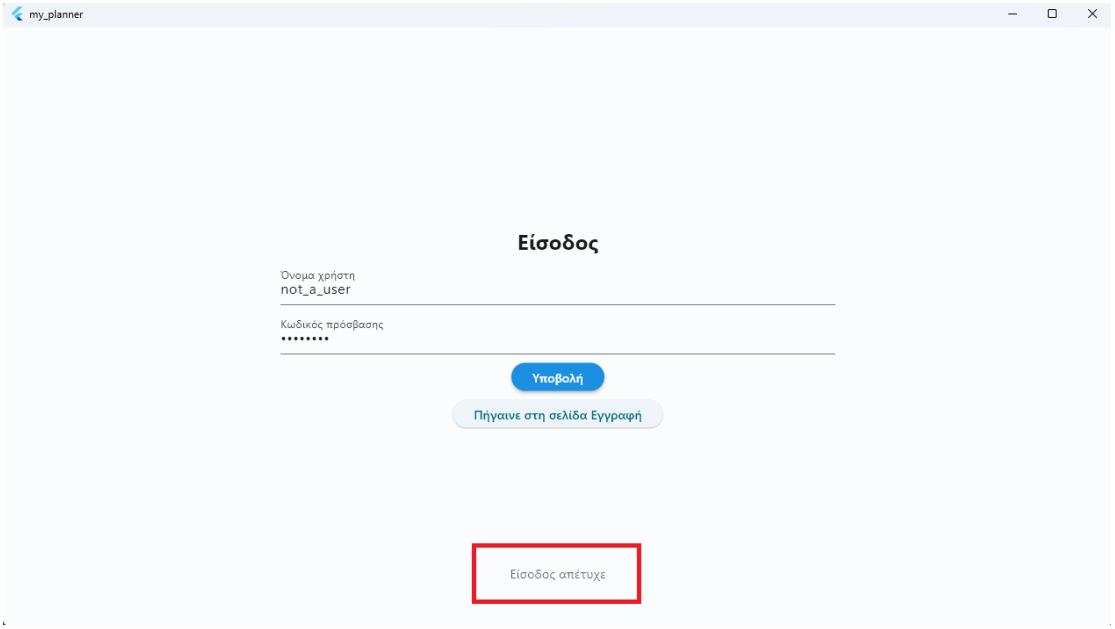
Δημιουργία χρήστη:



ΕΙΚΟΝΑ 38

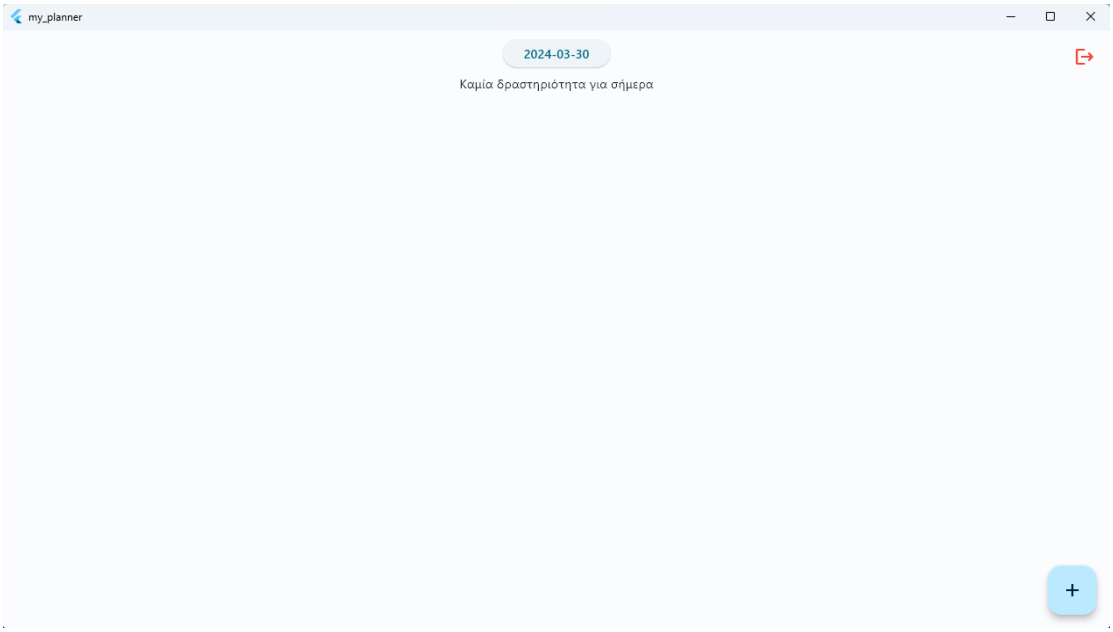
Log in χρήστη:

Κάνοντας log in με έναν χρήστη που δεν υπάρχει και παίρνουμε σφάλμα (Εικόνα 39).



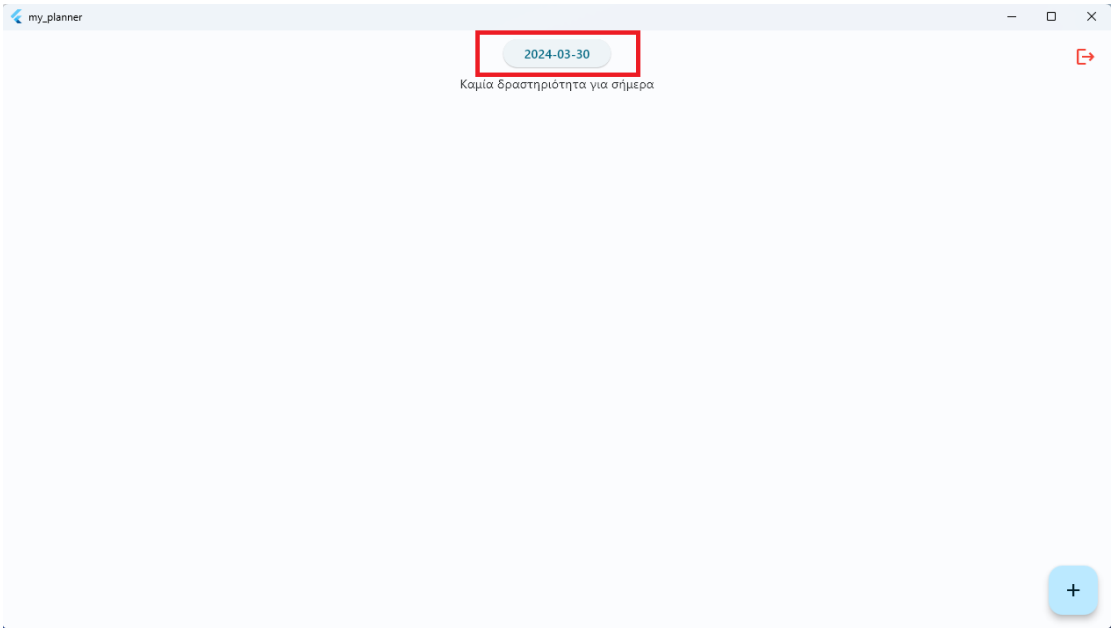
ΕΙΚΟΝΑ 39

Κάνοντας Log in με τον χρήστη που δημιουργήσαμε εισερχόμαστε στην αρχική σελίδα της εφαρμογής (Εικόνα 40).

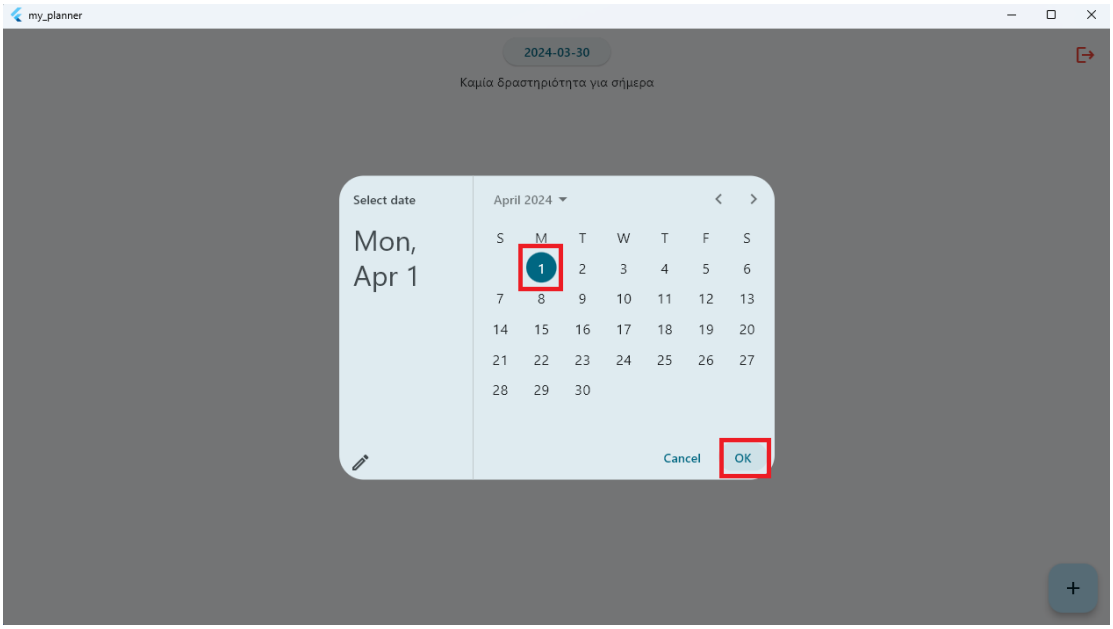


ΕΙΚΟΝΑ 40

Επιλογή ημερομηνίας:

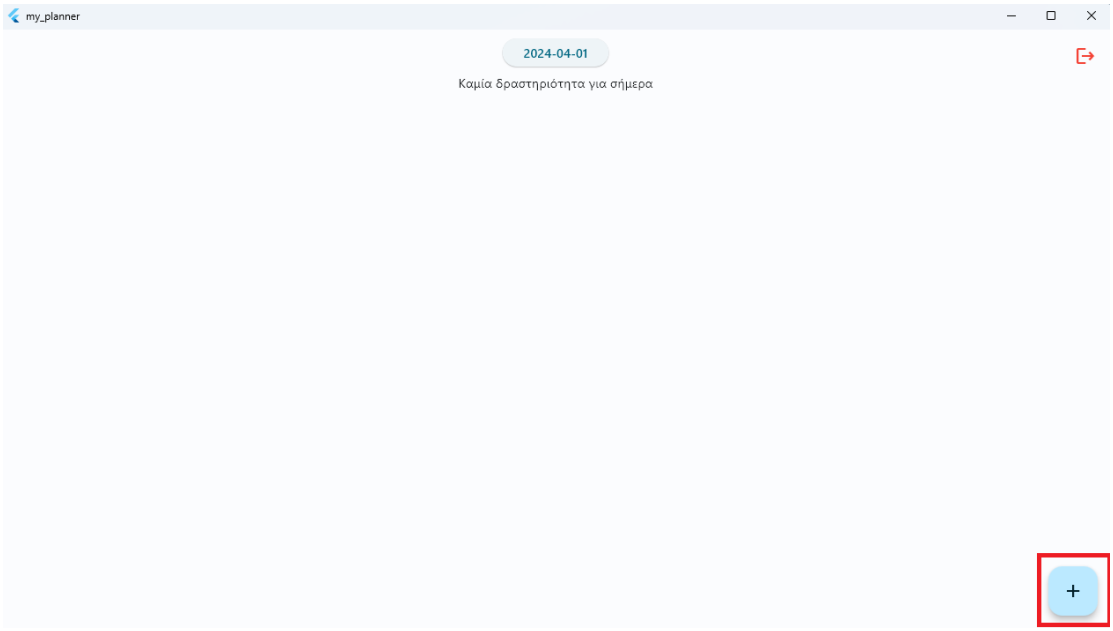


ΕΙΚΟΝΑ 41

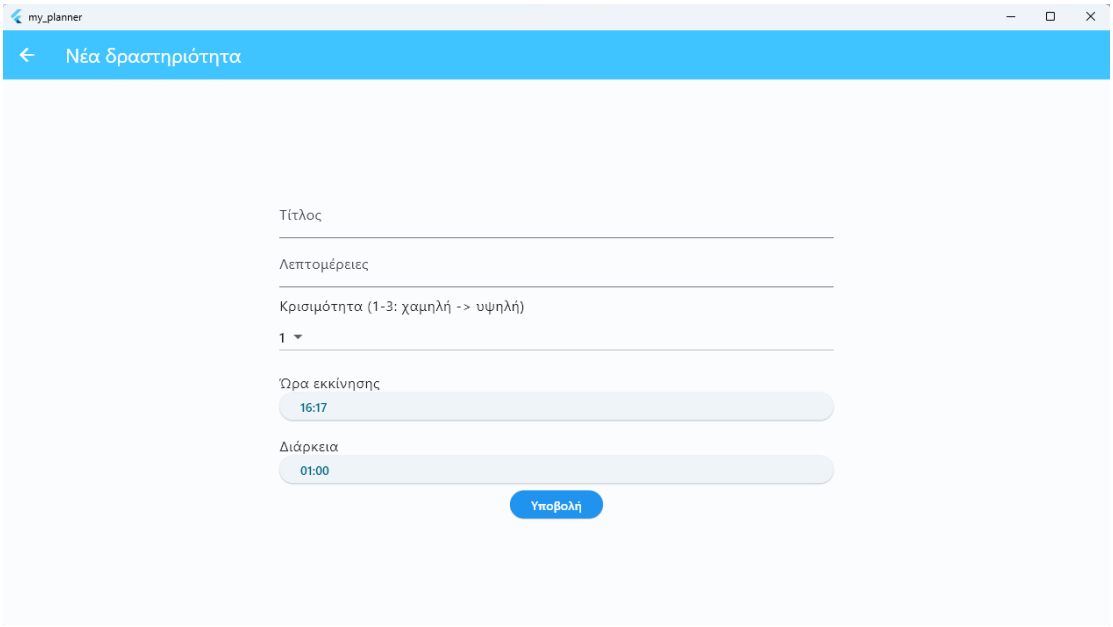


ΕΙΚΟΝΑ 42

Προσθήκη συμβάντος με μικρή σημαντικότητα:

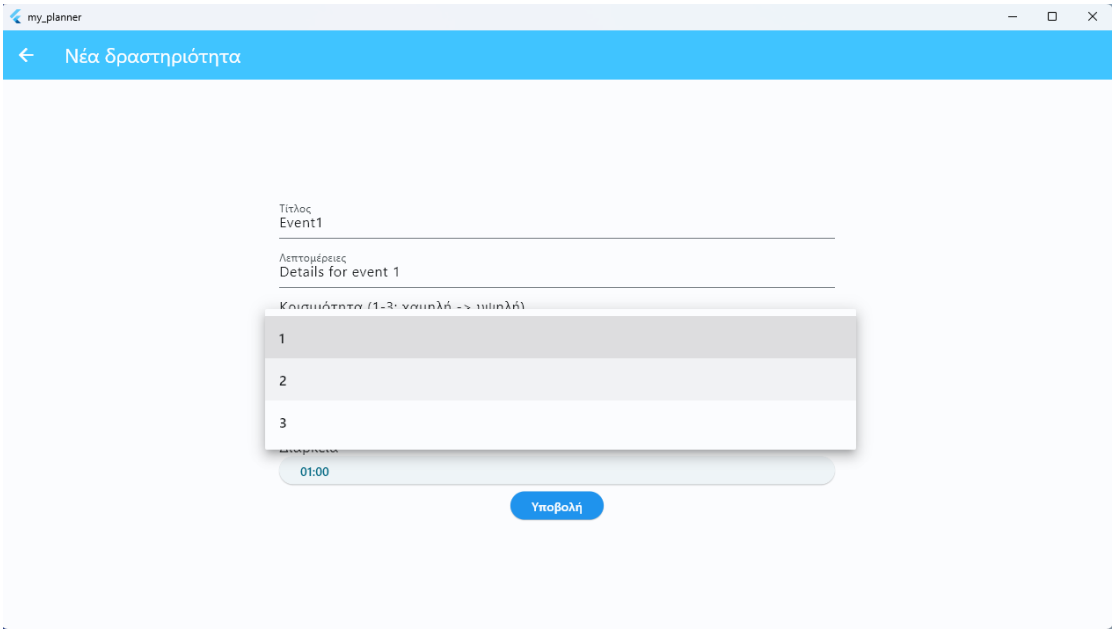


ΕΙΚΟΝΑ 43

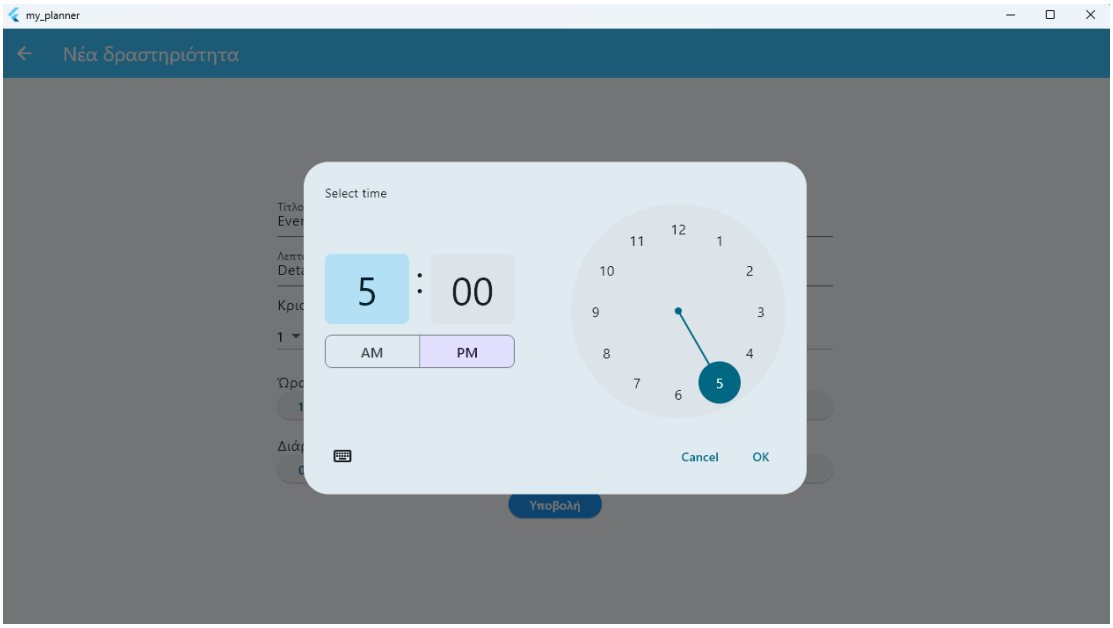


ΕΙΚΟΝΑ 44

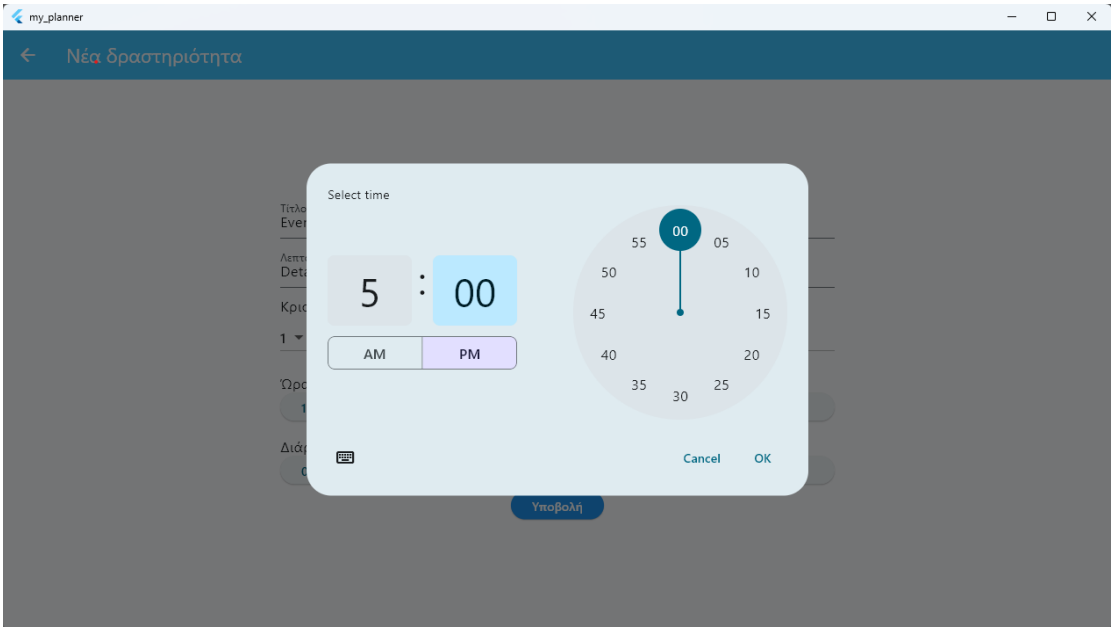
Συμπλήρωση δεδομένων.



ΕΙΚΟΝΑ 45

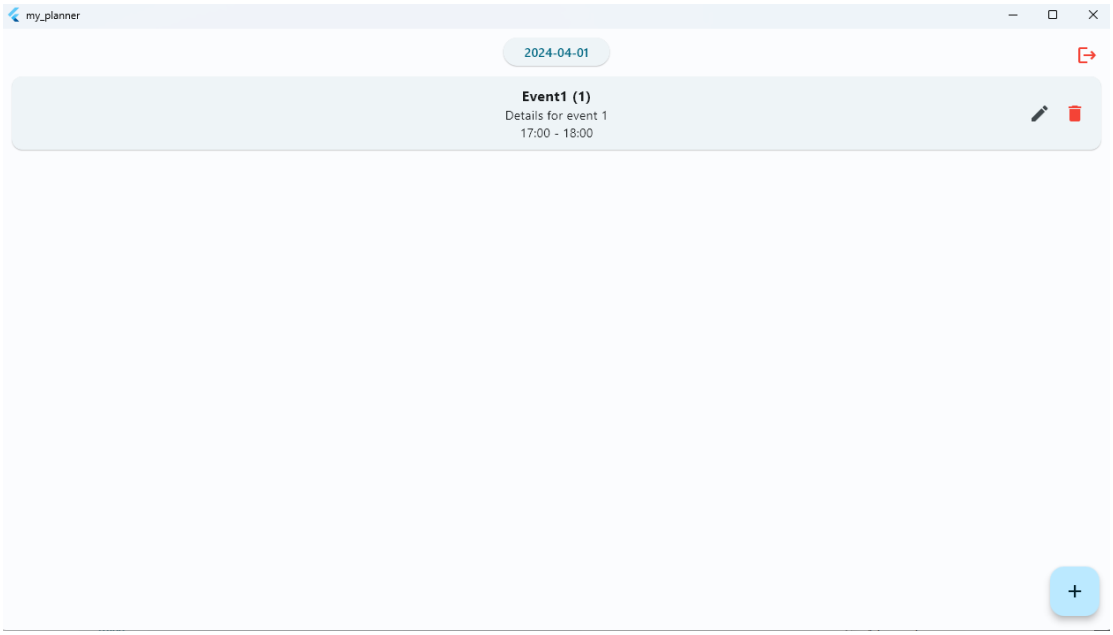


ΕΙΚΟΝΑ 46



ΕΙΚΟΝΑ 47

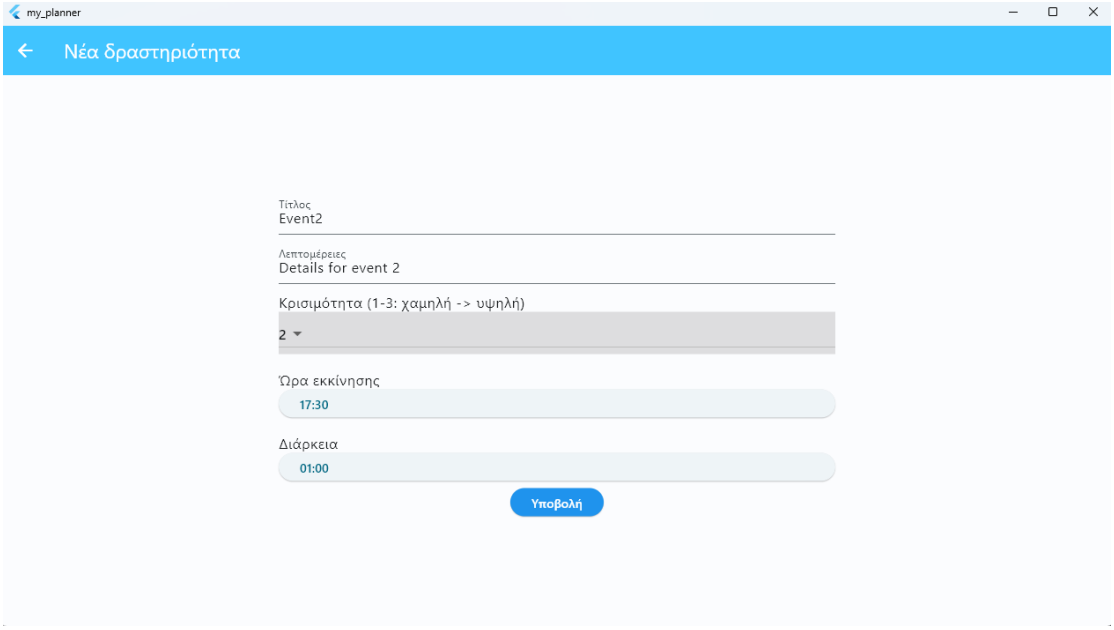
Πατώντας υποβολή βλέπουμε το event στην σελίδα δραστηριοτήτων.



ΕΙΚΟΝΑ 48

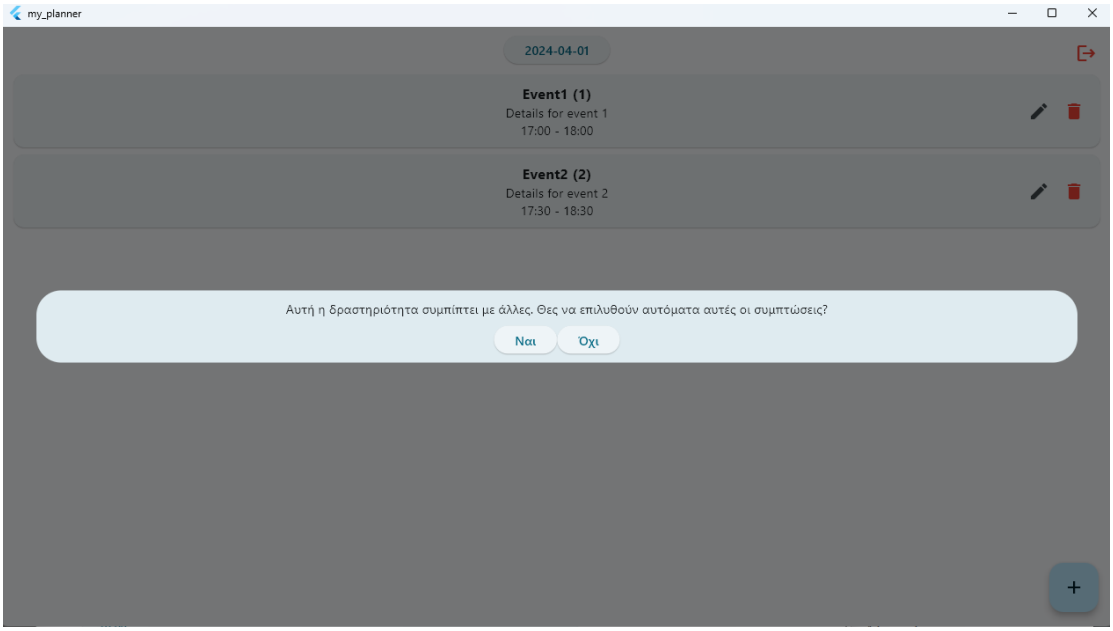
Προσθήκη συμβάντος που συμπίπτει με το προηγούμενο αλλά με μεγαλύτερη σημαντικότητα:

Επαναλαμβάνουμε την διαδικασία που περιεγράφηκε πριν απλά αυτήν την φορά το event μας θα έχει μεγαλύτερη σημαντικότητα (Εικόνα 49).



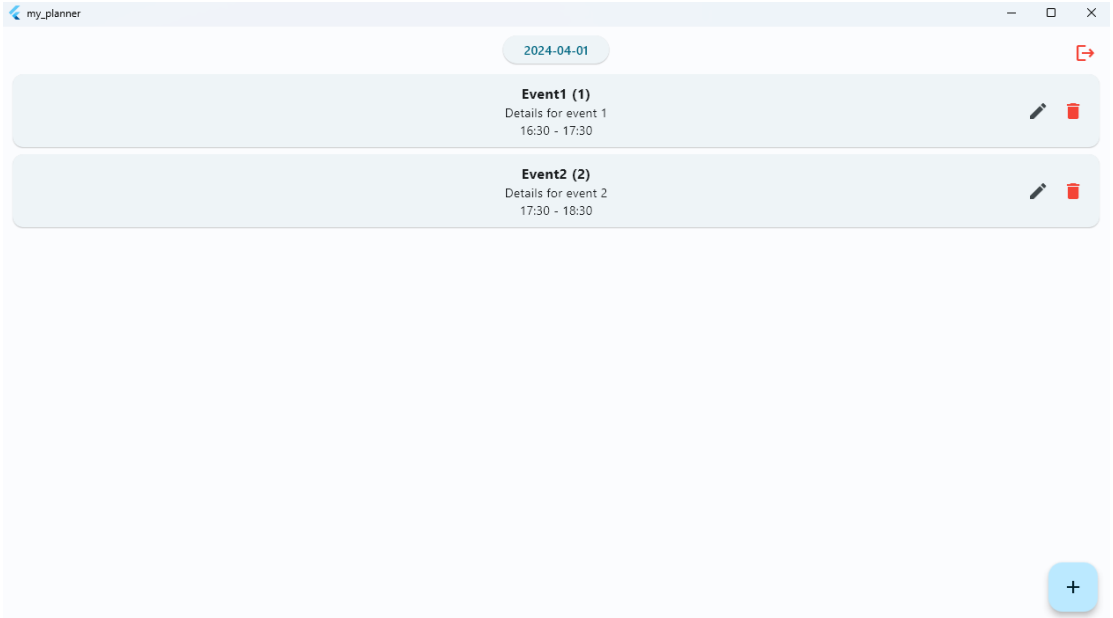
ΕΙΚΟΝΑ 49

Παρατηρούμε ότι τα δύο event μας συμπίπτουν χρονολογικά, έτσι πατώντας υποβολή ερχόμαστε αντιμέτωποι με το εξής μήνυμα (Εικόνα 50):



ΕΙΚΟΝΑ 50

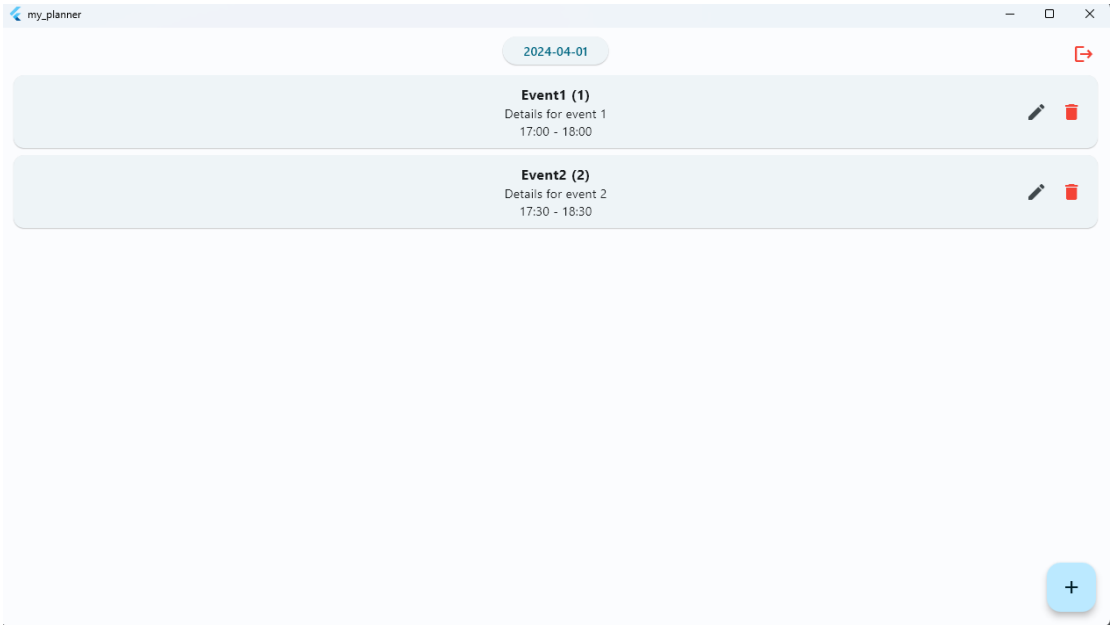
Επιλογή αυτόματης επίλυσης σύμπτωσης :



ΕΙΚΟΝΑ 51

Παρατηρούμε δύο πράγματα, πρώτον ότι το event με την μικρότερη σημαντικότητα μετακινήθηκε έτσι ώστε να μπει στην θέση του το event με την μεγαλύτερη σημαντικότητα (Εικόνα 51). Επίσης, παρατηρούμε ότι το event τοποθετήθηκε αυτόματα χρονολογικά πιο πριν, καθώς άρχιζε πιο νωρίς. Σε αντίστοιχη περίπτωση που το event ακολουθούσε, θα μετακινούταν πιο μετά.

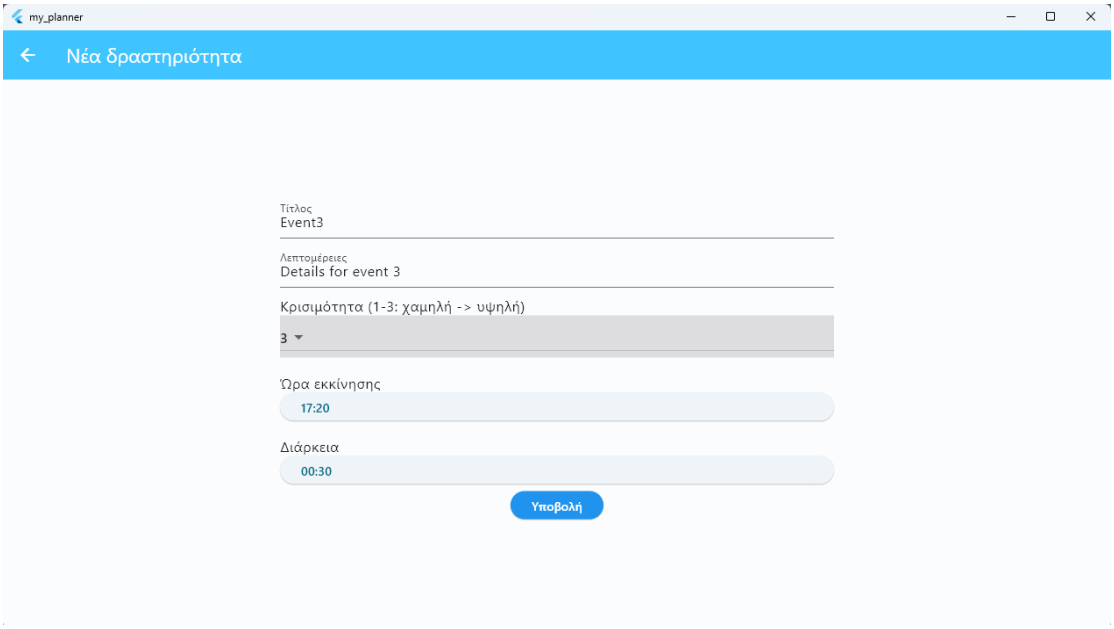
Επιλογή μη αυτόματης επίλυσης σύμπτωσης:



ΕΙΚΟΝΑ 52

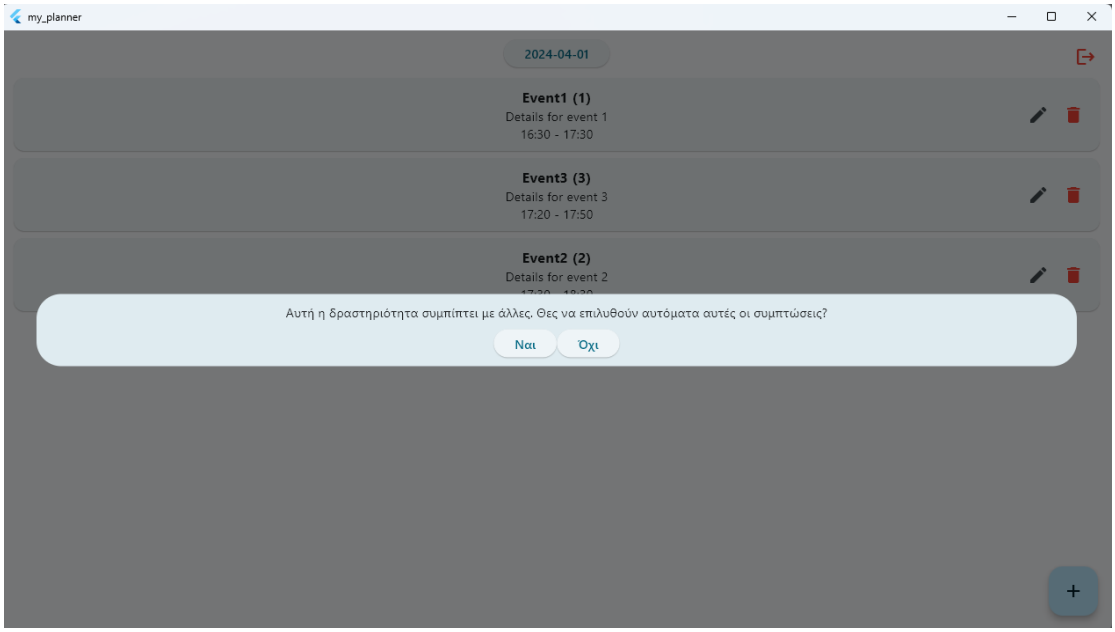
Σε αυτήν την περίπτωση παρατηρούμε ότι τα event παρέμειναν στην αρχική ώρα που είχαν προστεθεί (Εικόνα 52).

Προσθήκη συμβάντος που συμπίπτει με τα προηγούμενα αλλά με μεγαλύτερη σημαντικότητα και από τα δύο:

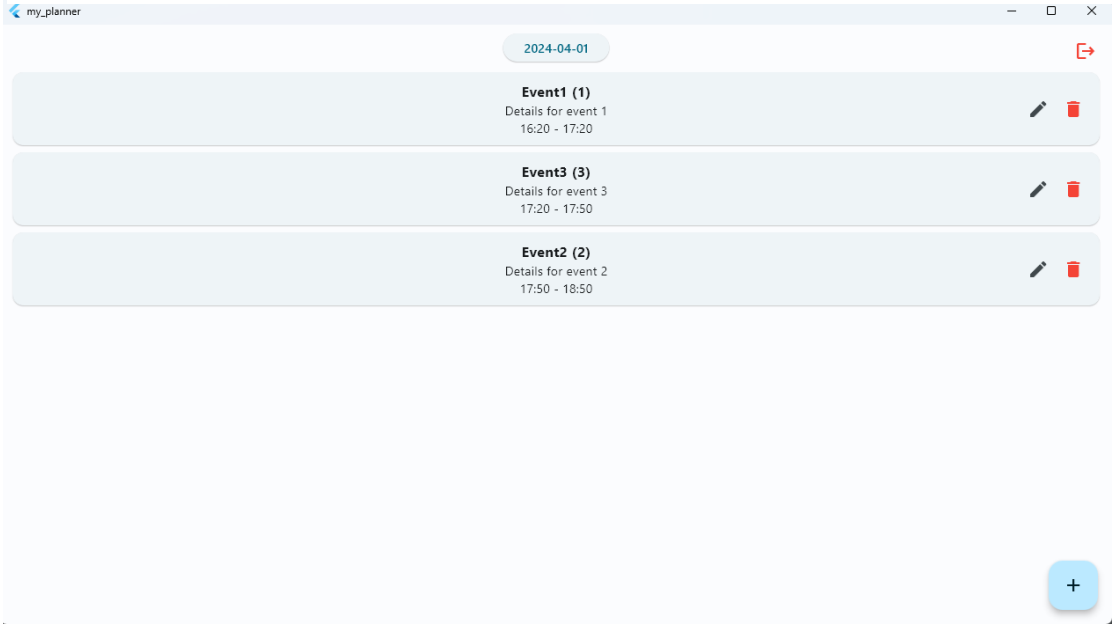


ΕΙΚΟΝΑ 53

Όπως και πριν έχουμε το pop up που μας ενημερώνει για τις χρονολογικές συμπτώσεις.



ΕΙΚΟΝΑ 54

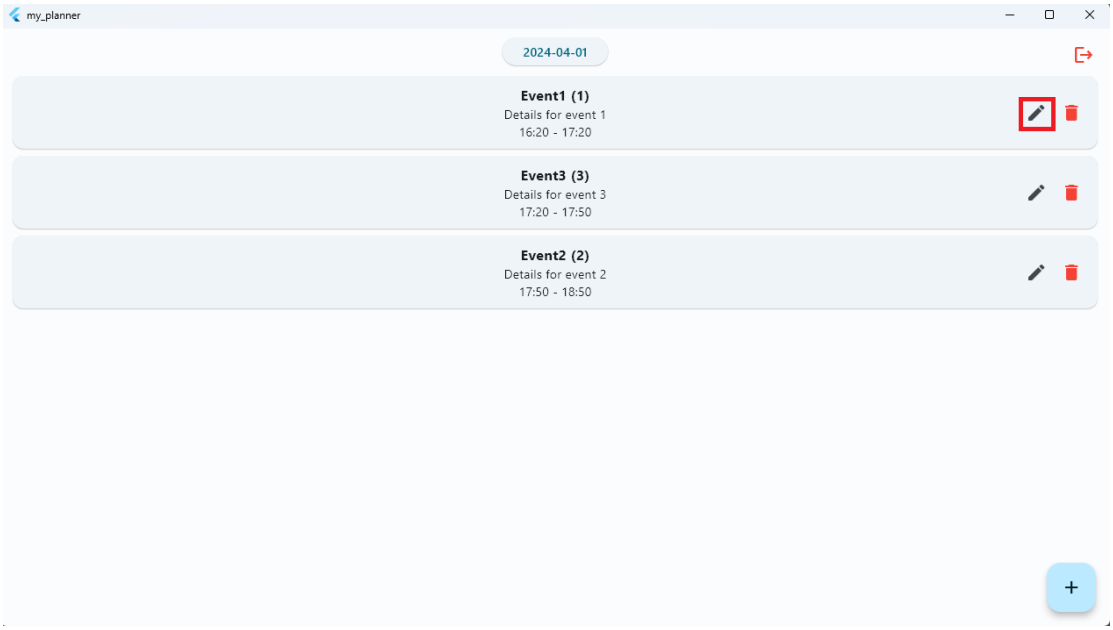


ΕΙΚΟΝΑ 55

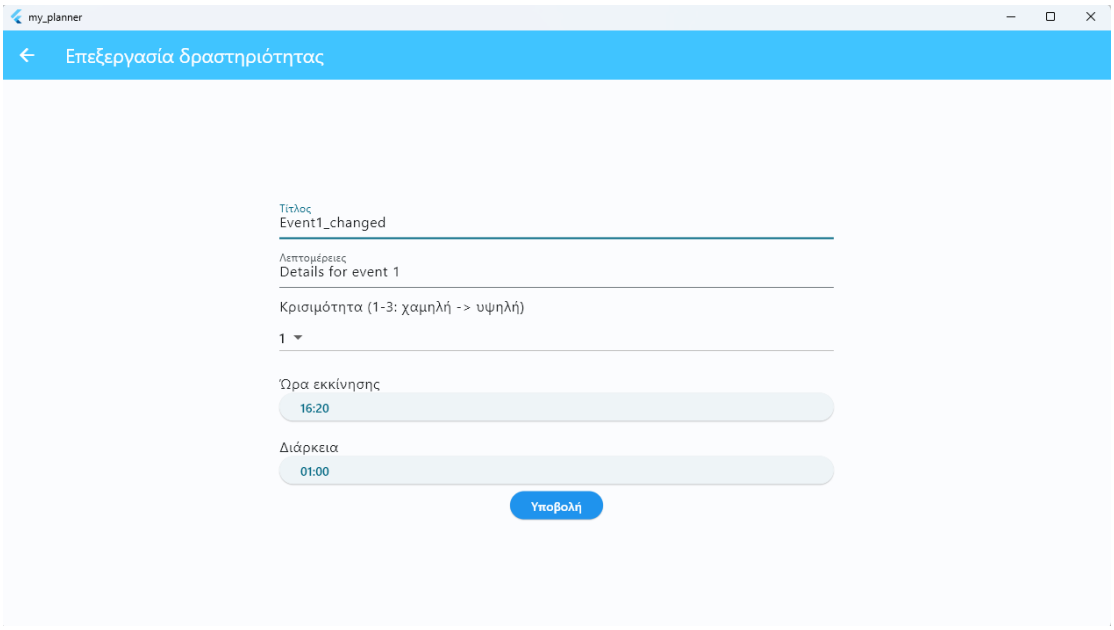
Σε αυτήν την περίπτωση παρατηρούμε ότι τα event με την μικρότερη σημαντικότητα μετακινήθηκαν το ένα πιο πριν και το άλλο πιο μετά κρατώντας έτσι την χρονολογική αλληλουχία τους (Εικόνα 55).

Επεξεργασία event:

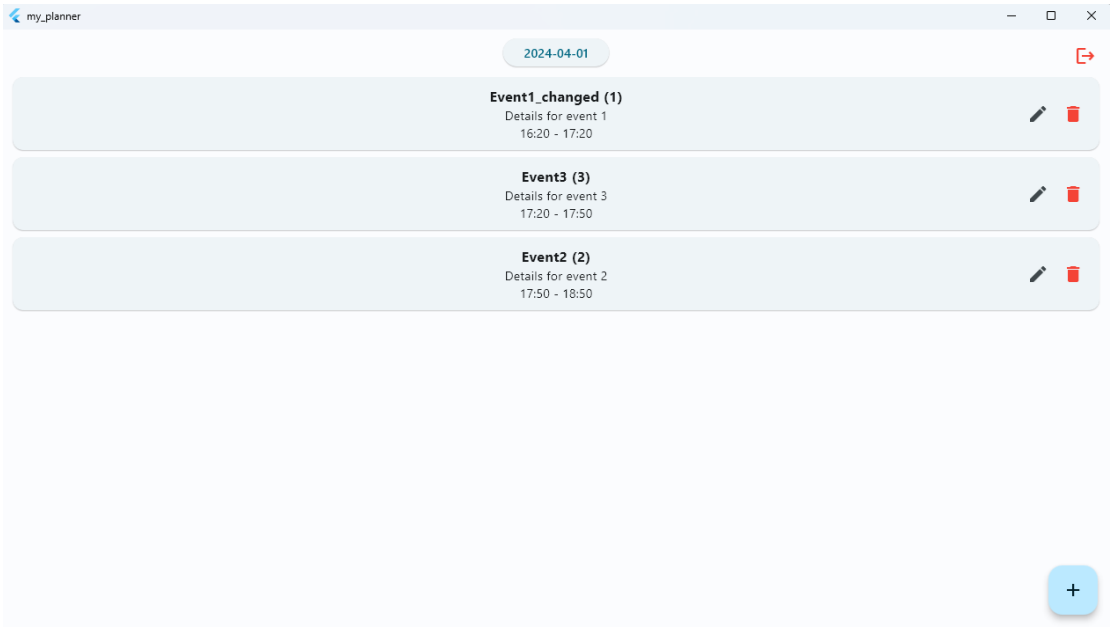
Πατώντας το κουμπί επεξεργασίας (μολυβί δίπλα από το event) μεταφερόμαστε στην σελίδα επεξεργασίας.



ΕΙΚΟΝΑ 56



ΕΙΚΟΝΑ 57

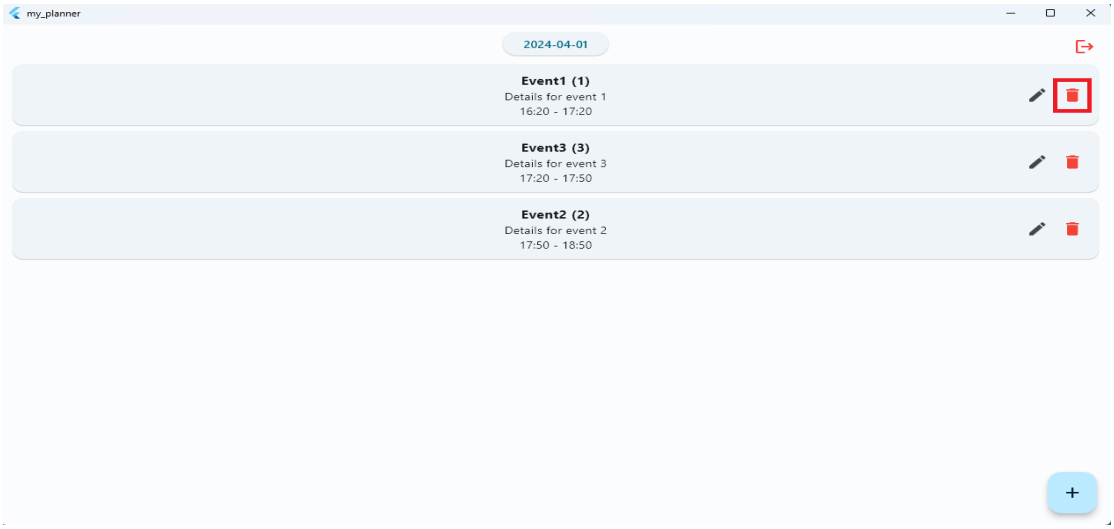


ΕΙΚΟΝΑ 58

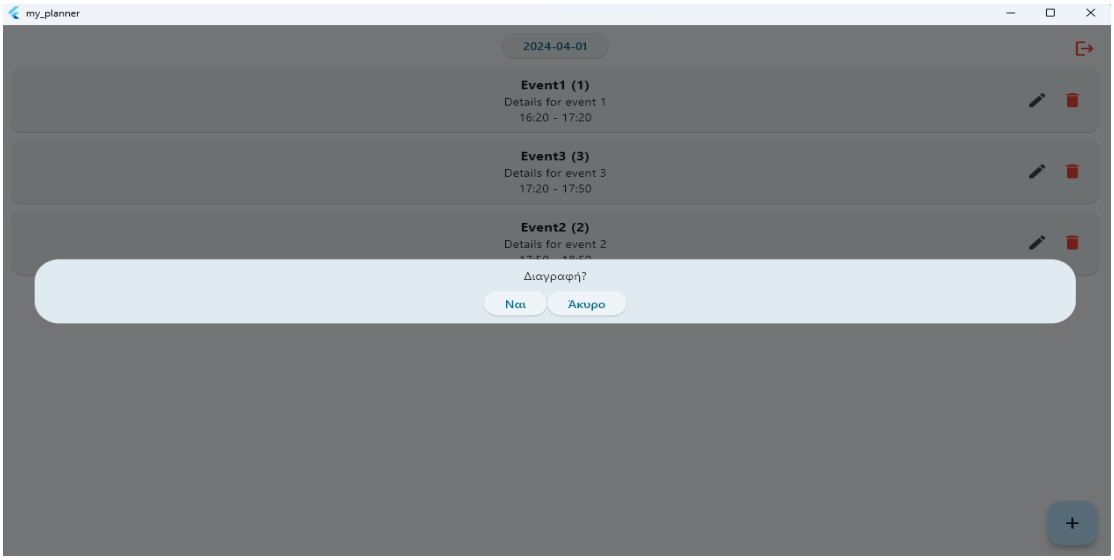
Είναι σημαντικό να αναφερθεί ότι σε περίπτωση που γίνει αλλαγή στην ώρα έναρξης ή την διάρκεια του event ξανά τσεκάρουμε αν υπάρχει τυχόν χρονολογική σύμπτωση και ακολουθούμε την διαδικασία που είδαμε και πιο πάνω (Εικόνα 51 & 55).

Διαγραφή event:

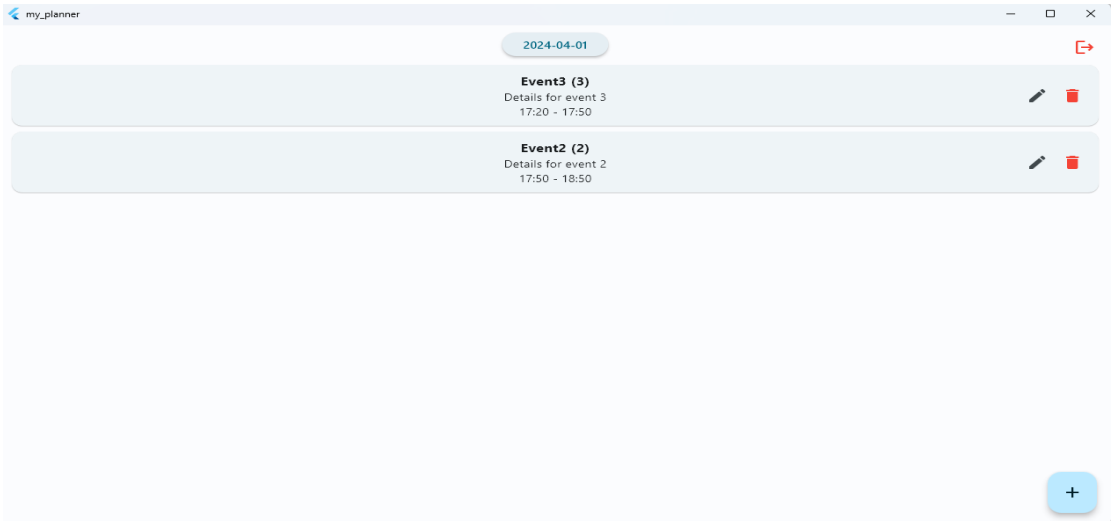
Πατώντας το κουμπί διαγραφής (κάδος δίπλα από το event) ο χρήστης μπορεί να διαγράψει ένα event..



ΕΙΚΟΝΑ 59



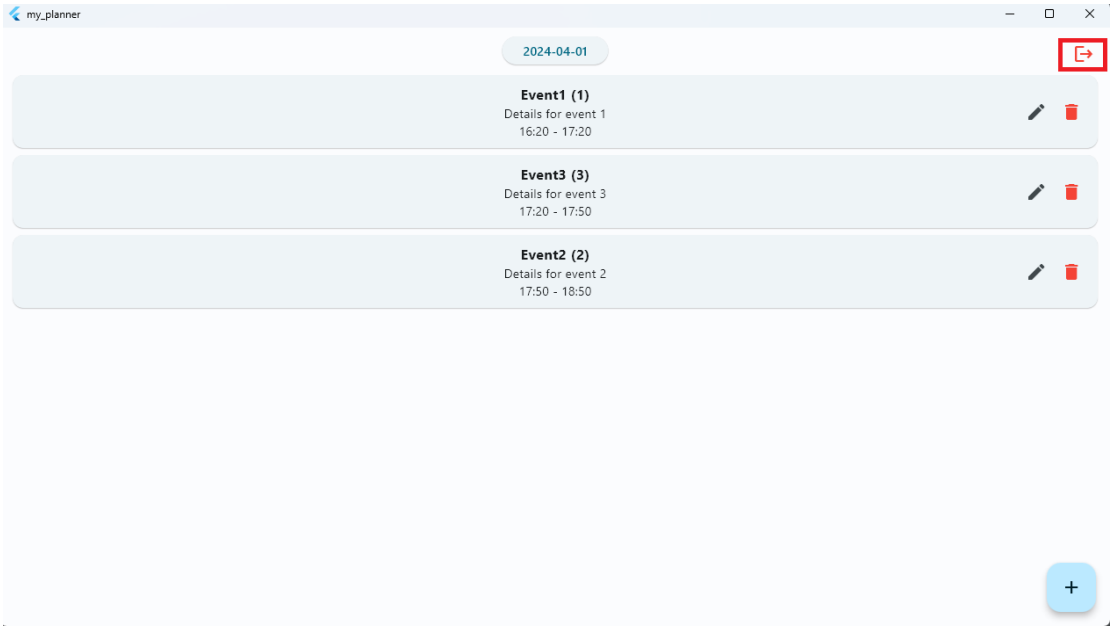
ΕΙΚΟΝΑ 60



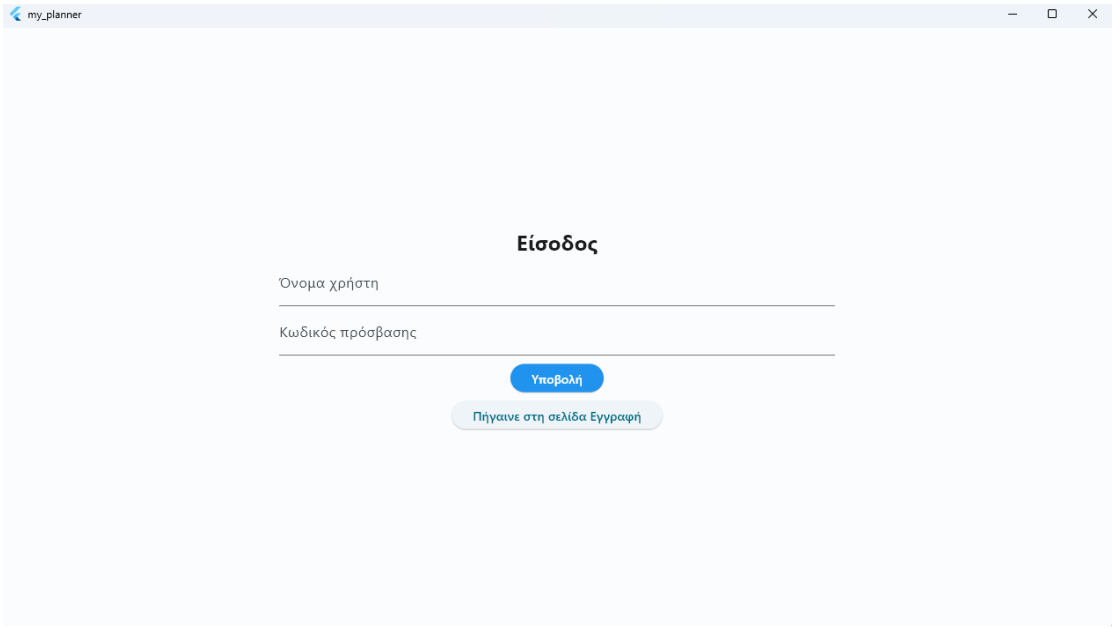
ΕΙΚΟΝΑ 61

Log out:

Πατώντας το κουμπί Log out ο χρήστης μεταφέρεται στην αρχική σελίδα εισόδου.



ΕΙΚΟΝΑ 62

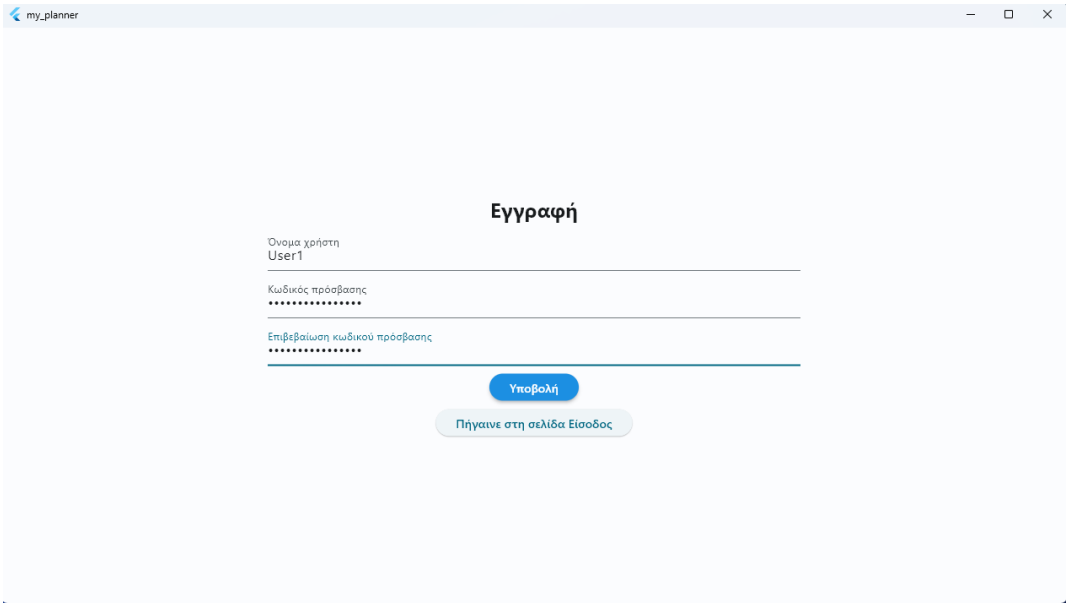


ΕΙΚΟΝΑ 63

16 Εγχειρίδιο Χρήστη

- Εγγραφή νέου χρήστη

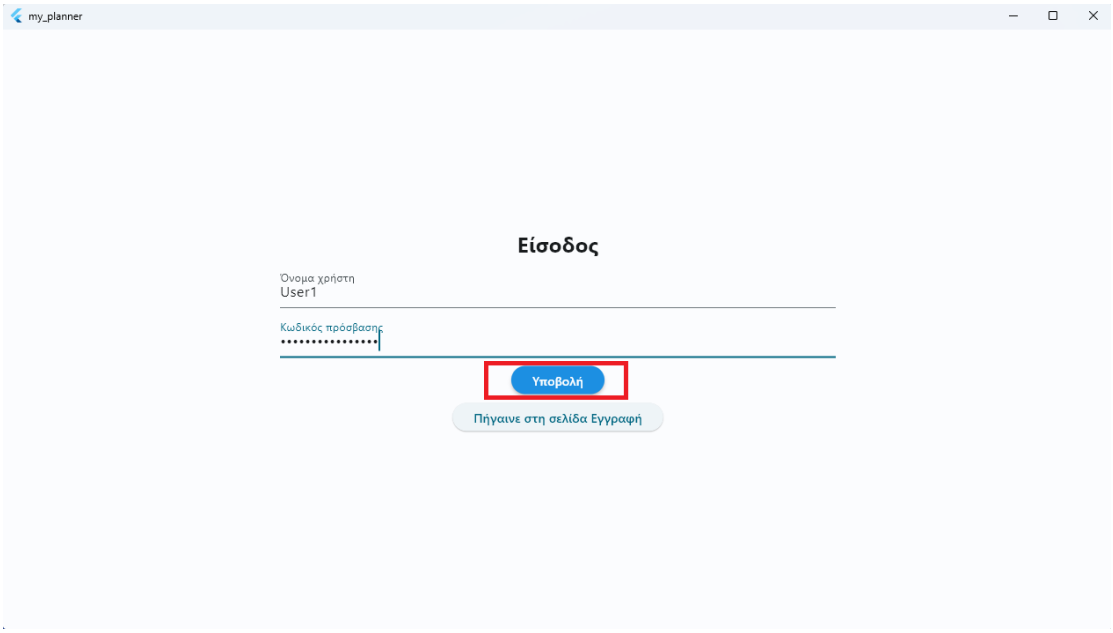
Στην αρχική οθόνη επιλέγουμε το κουμπί “Πήγαινε στη σελίδα εγγραφής”. Συμπληρώνουμε το επιθυμητό όνομα χρήστη και κωδικό, επιβεβαιώνουμε τον κωδικό και επιλέγουμε “Υποβολή” (Εικόνα 64). Μας υποδέχεται η κύρια σελίδα.



ΕΙΚΟΝΑ 64

- Είσοδος χρήστη

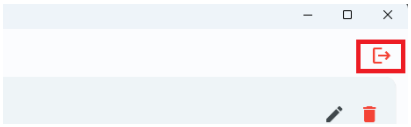
Στην αρχική οθόνη συμπληρώνουμε τα στοιχεία που υποβάλαμε κατά τη διαδικασία της εγγραφής. Επιλέγουμε “Υποβολή” και μας υποδέχεται η κύρια σελίδα (Εικόνα 65).



ΕΙΚΟΝΑ 65

- Αποσύνδεση χρήστη

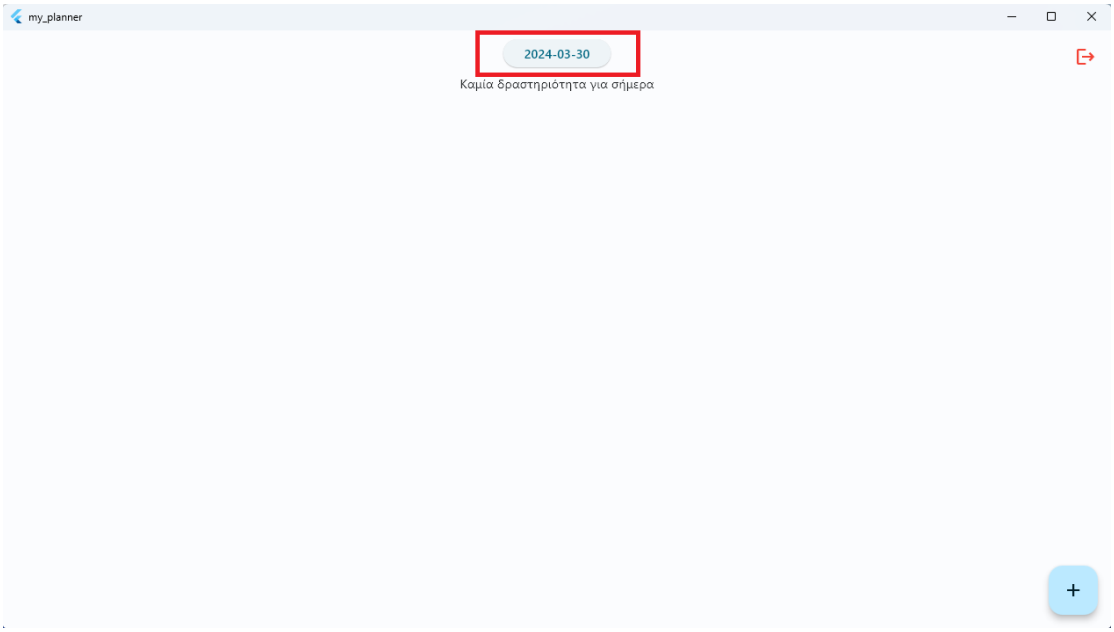
Στην πάνω δεξιά γωνία της κύριας σελίδας θα βρείτε το κουμπί αποσύνδεσης. Πατώντας το, επιστρέφετε πίσω στην αρχική σελίδα με την φόρμα εισόδου (Εικόνα 66).



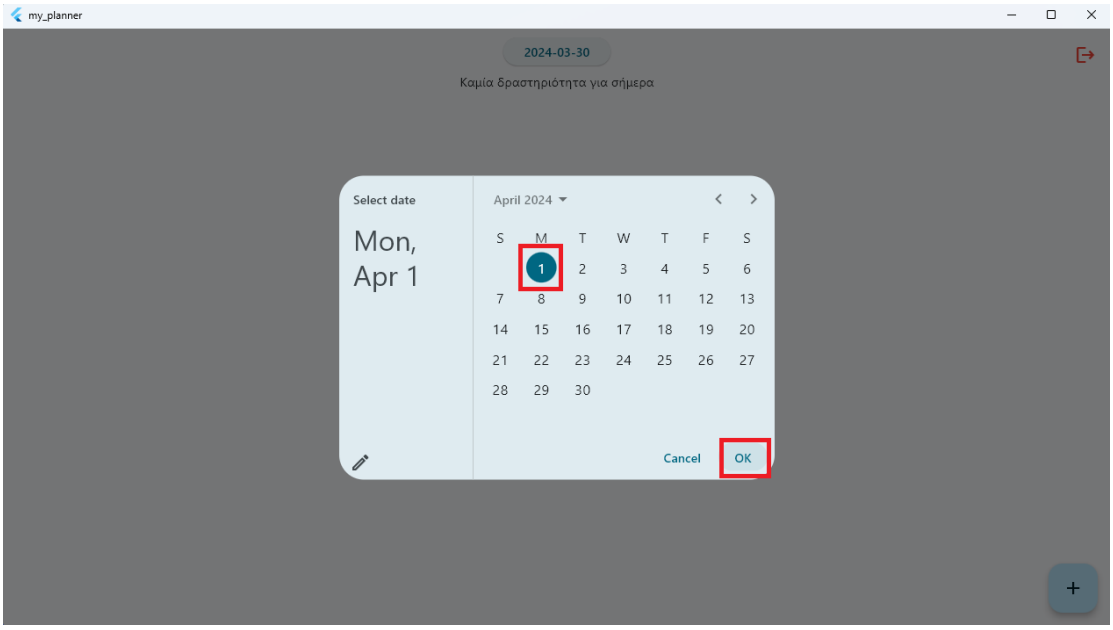
ΕΙΚΟΝΑ 66

- Επιλογή ημερομηνίας

Στο κεντρικό πάνω μέρος της κύριας σελίδας θα βρείτε τον επιλογέα ημερομηνίας (Εικόνα 67). Πατώντας πάνω του, θα εμφανιστεί ένα “popup” μενού επιλογής ημερομηνίας (Εικόνα 68). Η ημερομηνία αυτή αντιπροσωπεύει της ημέρα της οποίας τις αποθηκευμένες δραστηριότητες έχουμε επιλέξει να δούμε/επεξεργαστούμε.



ΕΙΚΟΝΑ 67

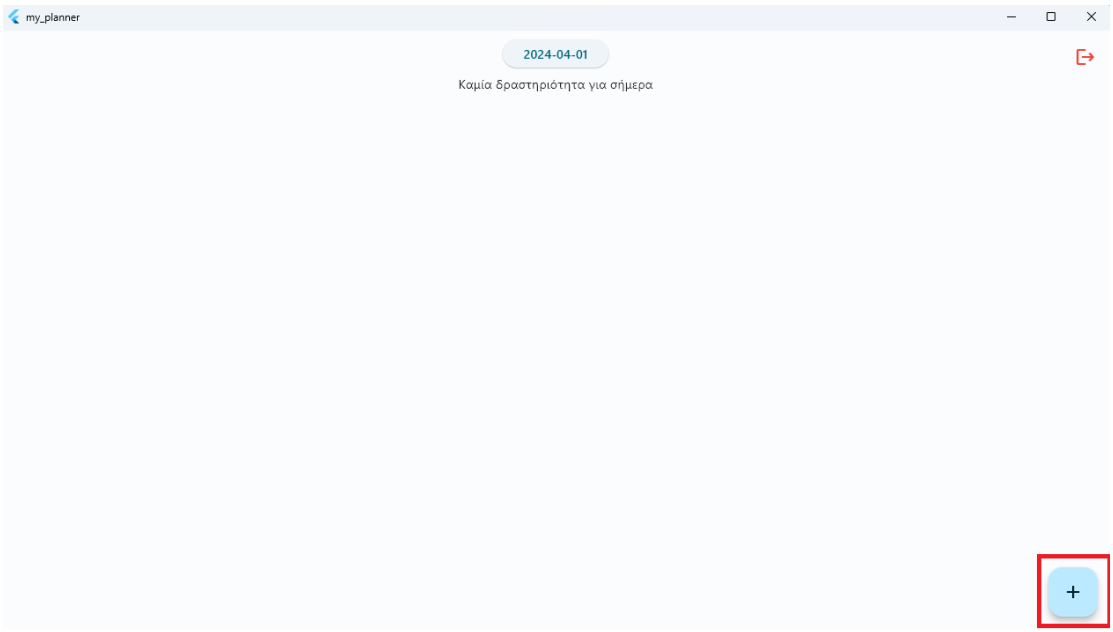


ΕΙΚΟΝΑ 68

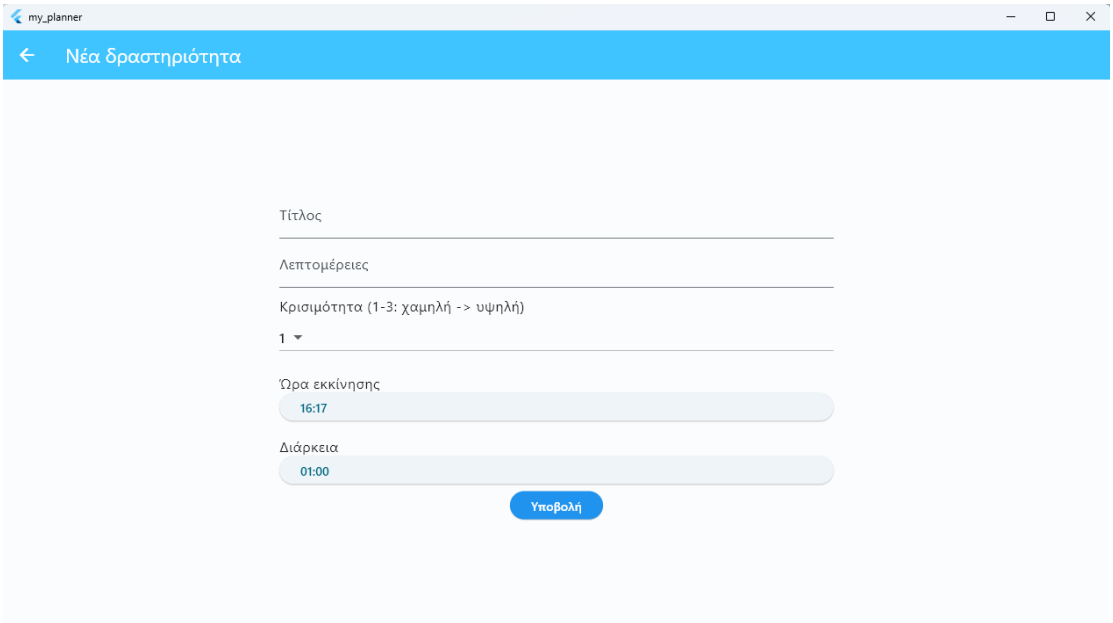
- Δημιουργία νέας δραστηριότητας

Πατώντας στο κουμπί με το σύμβολο “+” στο δεξιό κάτω μέρος της κύριας σελίδας, θα εμφανιστεί η φόρμα δημιουργίας νέας δραστηριότητας (Εικόνα 69).

Εδώ θα κληθείτε να ορίσετε τον τίτλο, τυχόν λεπτομέρειες, το βαθμό κρισιμότητας, την ώρα εκκίνησης καθώς και την διάρκεια (σε ώρες ή και λεπτά) της δραστηριότητας. Πατώντας “Υποβολή” θα αποθηκευτεί η νέα δραστηριότητα και θα επιστρέψετε στην κύρια σελίδα, η οποία πλέον θα περιέχει και την νέα δραστηριότητα (Εικόνα 70).



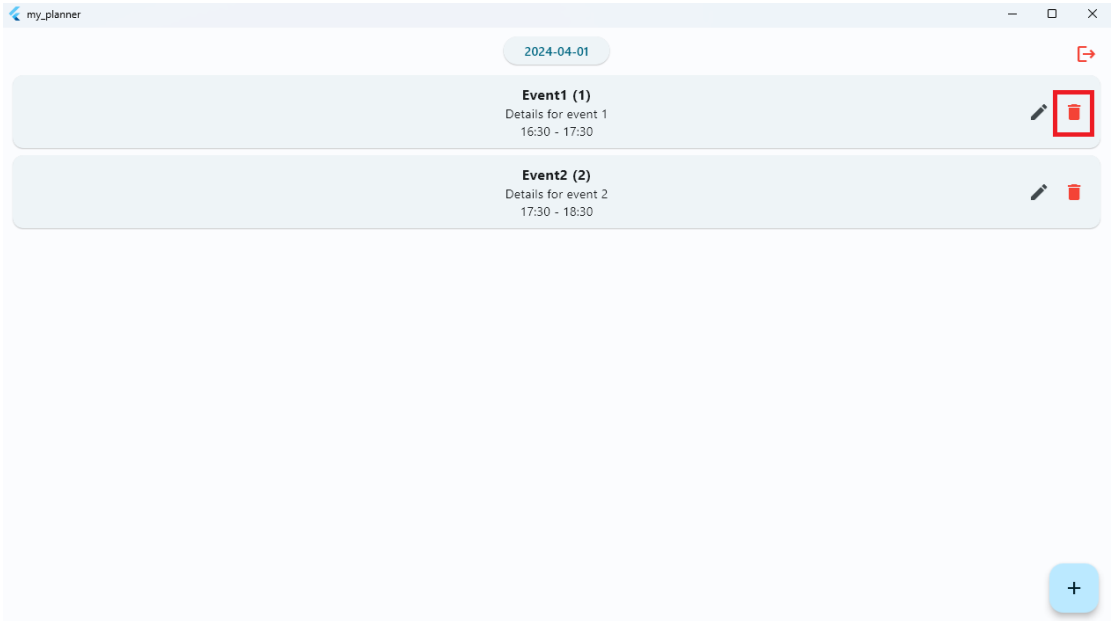
ΕΙΚΟΝΑ 69



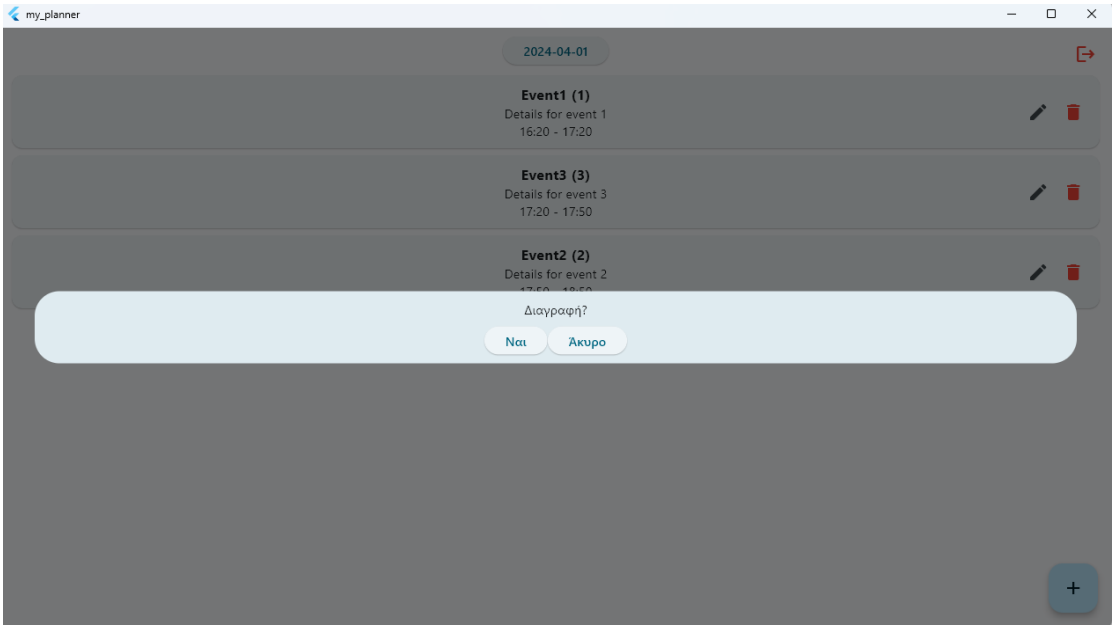
ΕΙΚΟΝΑ 70

- Διαγραφή δραστηριότητας

Στην κύρια οθόνη κάθε δραστηριότητα διαθέτει ένα κόκκινο κουμπί με το σύμβολο ενός κάδου απορριμμάτων (Εικόνα 71). Πατώντας το διαγράφετε την συγκεκριμένη δραστηριότητα (Εικόνα 72).



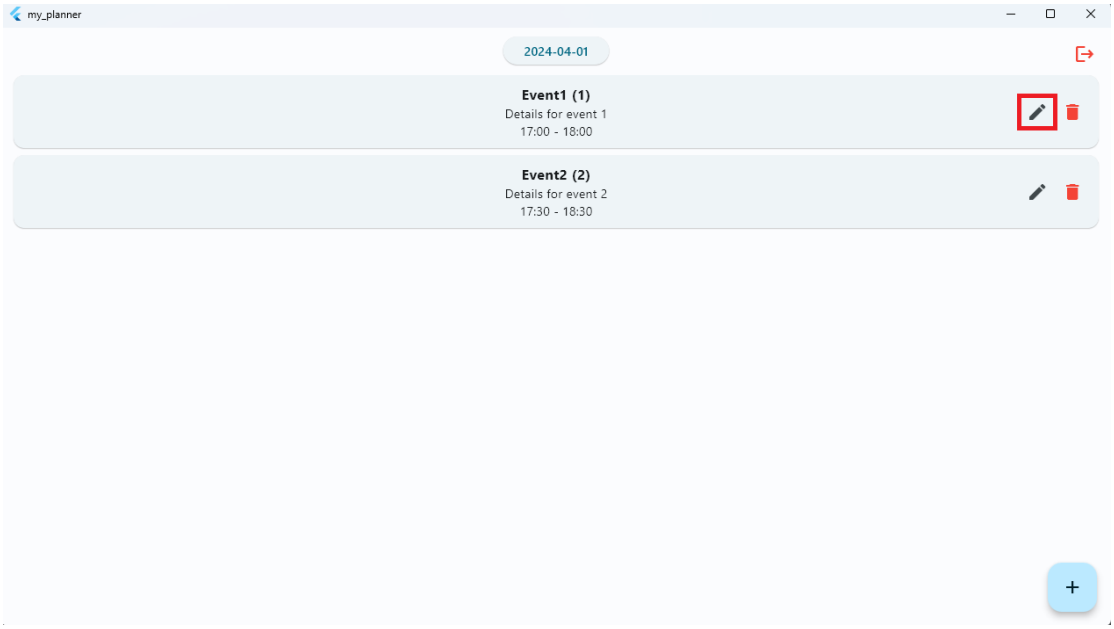
ΕΙΚΟΝΑ 71



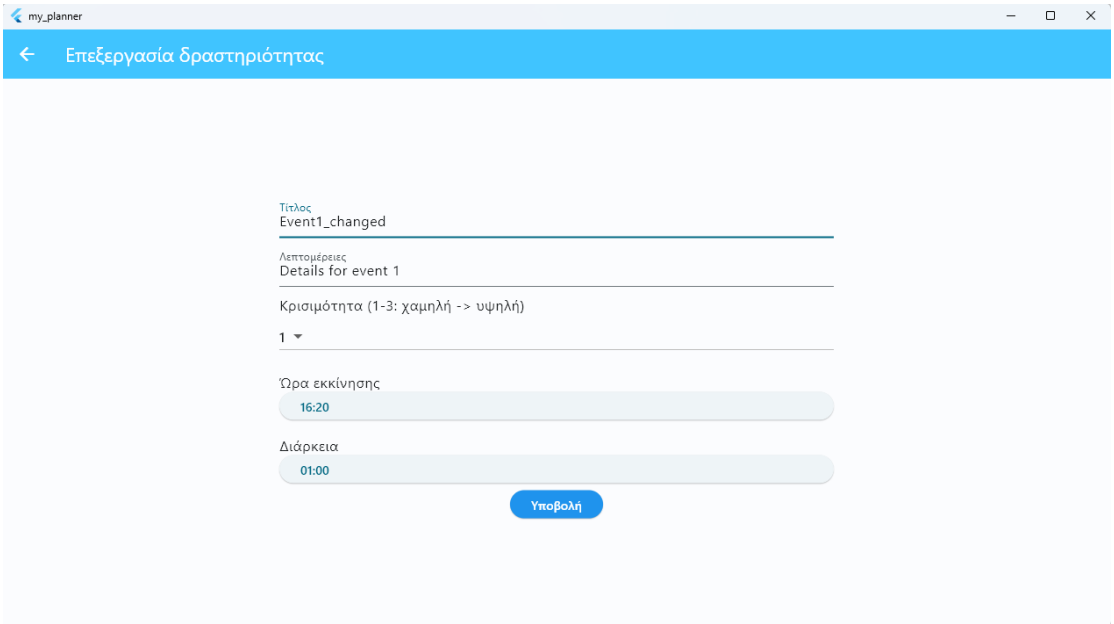
ΕΙΚΟΝΑ 72

- Επεξεργασία δραστηριότητας

Στη κύρια οθόνη κάθε δραστηριότητα διαθέτει ένα γκρίζο κουμπί με το σύμβολο μολυβιού (Εικόνα 73), το οποίο μας μεταφέρει σε μια φόρμα επεξεργασίας της συγκεκριμένης δραστηριότητας, όμοιο με τη φόρμα δημιουργίας νέας δραστηριότητας (Εικόνα 74).



ΕΙΚΟΝΑ 73

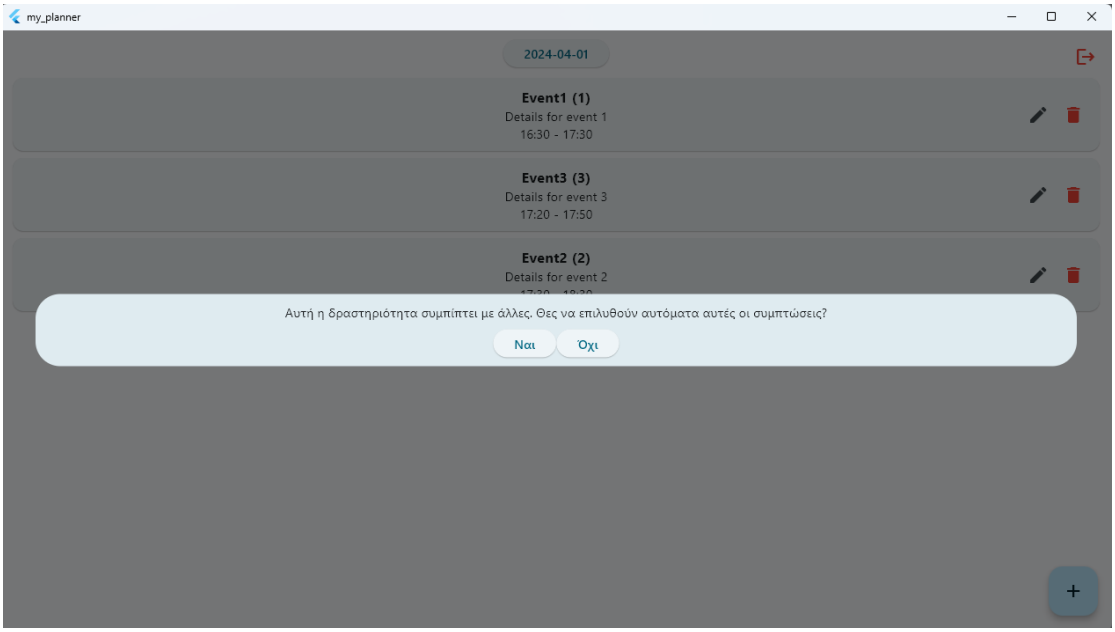


ΕΙΚΟΝΑ 74

- Αυτόματη επίλυση συμπτώσεων

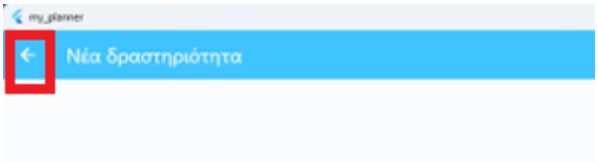
Αμέσως μετά την δημιουργία/επεξεργασία οποιασδήποτε δραστηριότητας, γίνεται αυτόματος έλεγχος συμπτώσεων όσον αφορά το χρονικό πλαίσιο, μεταξύ όλων των δραστηριοτήτων της επιλεγμένης ημέρας. Εάν βρεθούν συμπτώσεις θα εμφανιστεί ένα μήνυμα το οποίο ρωτάει εάν θέλετε οι συμπτώσεις αυτές να επιλυθούν αυτόματα, διαδικασία η οποία γίνεται λαμβάνοντας υπόψη το επίπεδο κρισιμότητας και την χρονική αλληλουχία τους (Εικόνα 75).

Μπορείτε να απορρίψετε την επιλογή αυτή σε περίπτωση που δεν θέλετε να αλλάξει κάτι, ή εάν θέλετε να τις επιλύσετε με δικό σας τρόπο.



ΕΙΚΟΝΑ 75

Καθ' όλη την εφαρμογή θα βρείτε κουμπιά με το κείμενο "Ακύρωση" (Εικόνα 76) ή το σύμβολο πίσω βέλους (Εικόνα 77), τα οποία μπορείτε να χρησιμοποιήσετε σε περίπτωση που δεν επιθυμείτε να εκτελέσετε την τρέχουσα επιλεγμένη ενέργεια και θα επιστρέψετε στην προηγούμενη κατάσταση δίχως εφαρμογή αλλαγών.



ΕΙΚΟΝΑ 77

ΕΙΚΟΝΑ 76

17 Οφέλη που αναμένουμε να έχουμε από την λύση που προτείνεται στην πτυχιακή

Τα οφέλη τα όποια αναμένουμε να έχουμε, σε αντίθεση με τις λύσεις που αναφέρθηκαν και πιο πάνω, είναι ότι η επίλυση των χρονικών συμπτώσεων γίνεται αυτόματα χωρίς ο χρήστης να χρειάζεται κάθε φορά να πάει να αλλάζει την εγγραφή χειροκίνητα. Η αλλαγή πραγματοποιείται δυναμικά σε πραγματικό χρόνο και έτσι μπορούμε να δίνουμε και οπτική ενημέρωση στον χρήστη ότι η εγγραφή του μετακινήθηκε χρονολογικά.

Επίσης, η εφαρμογή αναγνωρίζει αν η πιο σημαντική εγγραφή πραγματοποιείται πριν ή μετά τις λιγότερο σημαντικής εγγραφής και έτσι μπορεί να μετακίνηση χρονολογικά αυτήν πίσω ή εμπρός αντίστοιχα. Με αυτόν τον τρόπο ο χρήστης δεν χάνει ούτε μπερδεύει την αλληλουχία των εγγραφών του στο ημερολόγιό του.

Τέλος, σε κάθε περίπτωση δίνουμε την δυνατότητα στον χρήστη να ακύρωση την αυτόματη επίλυση και να κρατήσει τις εγγραφές στις αρχικές χρονολογικές περιόδους που έχει ορίσει χωρίς αυτές να αλλάξουν. Έτσι, αν ο χρήστης θελήσει μπορεί να έχει και εγγραφές που συμπίπτουν χρονολογικά ανεξαρτήτως τις σημαντικότητάς τους.

18 Συμπεράσματα

18.1 Σύνοψη

Όπως αναφέρθηκε και στην αρχή, στις μέρες μας τα πάντα κινούνται σε τόσο γρήγορους ρυθμούς που είναι πολλές φορές ακατόρθωτο να καταφέρουμε να κρατάμε όλες μας τις υποχρεώσεις σε τάξη. Για τον λόγο αυτό είναι απαραίτητη η ύπαρξη μιας εφαρμογής που όχι μόνο μπορεί να περιέχει όλο το καθημερινό μας πρόγραμμα με τον καθημερινό φόρτο εργασίας μας αλλά και να μας δίνει την δυνατότητα να επιλύουμε εύκολα και γρήγορα τυχόν χρονολογική σύμπτωση μεταξύ των υποχρεώσεών μας.

18.2 Περιορισμοί και προβλήματα που συναντήθηκαν

Υλοποιώντας την παραπάνω εργασία καλύφθηκε σε μεγάλο βαθμό το πρόβλημα αυτό. Οι χρήστες πλέον έχουν πρόσβαση σε ένα πολύ φιλικό περιβάλλον, όπου μπορούν να περιηγηθούν εύκολα και με μόλις λίγα βήματα μπορούν να έχουν πρόσβαση σε ένα ημερολόγιο μέσω του οποίου μπορούν να οργανώσουν όλες τους τις υποχρεώσεις.

Παρόλα αυτά, δεν σημαίνει ότι το λογισμικό My_Planner είναι άριστο. Υπήρχαν πολλοί περιορισμοί που εμφανίστηκαν με κύριο περιορισμό την αδυναμία δημιουργία εγγραφών που διατρέχουν περισσότερες από μια ημέρες. Πιο συγκεκριμένα, τα εργαλεία που έχουμε στην διάθεσή μας δεν μας δίνουν την δυνατότητα να προσομοιώσουμε τον χρόνο ως ένα συνεχές διάγραμμα που απαρτίζεται από 24ωρά αλλά σε αντίθεση είναι πολύ πιο απλό να φτιάξουμε ένα μοντέλο που απαρτίζεται από ξεχωριστές ημέρες όπου η κάθε μία από αυτές έχει συγκεκριμένες ώρες.

18.3 Μελλοντικές επεκτάσεις

Επιπρόσθετα, θα μπορούσαμε μελλοντικά να εισάγουμε και την δυνατότητα ειδοποιήσεων στην εφαρμογή που θα ενημερώνουν τον χρήστη ότι έχει φτάσει η ώρα που έχει οριστεί στην εγγραφή του. Αυτό θα είχε και σαν προϋπόθεση την συνεχής εκτέλεση της εφαρμογής, ίσως σε κάποια hibernated μορφή έτσι ώστε να μην σπαταλά τους ίδιους πόρους σε σύγκριση με όταν τρέχει κανονικά.

Μια άλλη προσθήκη που μπορεί να γίνει μελλοντικά είναι η ανάπτυξη της και για άλλα λειτουργικά συστήματα ή και συσκευές. Όπως περιγράφεται και πιο πάνω ή ανάπτυξη της εφαρμογής έγινε με την χρήση Android studio και του flutter. Το flutter υποστηρίζει την δημιουργία της εφαρμογής και για Linux/MacOS συστήματα καθώς και για κινητά τηλέφωνα με την προσθήκη κάποιων βιβλιοθηκών.

Τέλος, το πρόγραμμα μας είναι ελλιπής όσον αφορά διάφορες ενέργειες που ο χρήστης μπορεί να εκτελέσει. Πιο συγκεκριμένα, εκτός από τις 3 βασικές δυνατότητες (δημιουργία, επεξεργασία και διαγραφή εγγραφών), θα μπορούσαμε να επεκτείνουμε την λειτουργικότητα προσθέτοντας δυνατότητες όπως :

- Δημιουργία περισσότερων από ένα ημερολόγιών για έναν χρήστη. Έτσι ώστε να μπορεί να διαχωρίσει τις υποχρεώσεις του σε κατηγορίες (προσωπικά, επαγγελματικά κτλ.)
- Την δυνατότητα δημιουργίας κάποιων επαναλαμβανομένων εγγράφων με μια μόνο προσθήκη.

Έτσι, θα δίναμε μια πιο ευρεία γκάμα ενεργειών στον τελικό χρήστη.

19 Βιβλιογραφία

[1] Sqflite: 2.3.2. (2024, Φεβρουάριος 28). Ανάκτηση από https://pub.dev/packages/sqflite

[2] fluttertoast: 8.2.4. ((2024, Φεβρουάριος 28). Ανάκτηση από https://pub.dev/packages/fluttertoast

[3] Clickup. (2024, Μάρτιος 29). Ανάκτηση από https://clickup.com/

[4] Microsoft Outlook (Wikipedia). (2024, Μάρτιος 29). Ανάκτηση από

https://en.wikipedia.org/wiki/Microsoft_Outlook

[5] Flutter FutureBuilder. (2024, Φεβρουάριος 28). Ανάκτηση από https://api.flutter.dev/flutter/widgets/FutureBuilder-class.html

[5] Flutter async. (2024, Φεβρουάριος 28). Asynchronous programming: futures, async, await. Ανάκτηση από https://dart.dev/codelabs/async-await

[6] Cozi. (2024, Μάρτιος 29). Ανάκτηση από https://www.cozi.com/

[7] Async and Wait. (2024, Φεβρουάριος 28). Asynchronous programming with async and await. Ανάκτηση από https://learn.microsoft.com/en-us/dotnet/csharp/asynchronous-programming/

[8] Flutter widgets. (2024, Φεβρουάριος 28). Widget catalog. Ανάκτηση από https://docs.flutter.dev/ui/widgets

[9] Architecture Diagram. (2024, Απρίλιος 1). System Architecture Diagram: A Complete Tutorial. Ανάκτηση από https://www.edrawsoft.com/article/system-architecture-diagram.html

[10] Kabudi, T., Pappas, I. and Olsen, D.H., 2021. AI-enabled adaptive learning systems: A systematic mapping of the literature. *Computers and Education: Artificial Intelligence*, *2*, p.100017.

[11] Hayes-Roth, B., 1995. An architecture for adaptive intelligent systems. Artificial intelligence, 72(1-2), pp.329-365. (2024, Ιούλιος 1)

[12] Rodil, K., Elisabeth Larsen, C., Caesar Faelled, C., Færch Skov, E., Gustafsen, T., Krummheuer, A. and Rehm, M., 2020, October. Spending time: co-designing a personalized calendar at the care center. In Proceedings of the 11th Nordic Conference on Human-Computer Interaction: Shaping Experiences, Shaping Society (pp. 1-11).

[13] Sarker, I.H., Colman, A., Han, J., Kayes, A.S.M. and Watters, P., 2020. CalBehav: A machine learning-based personalized calendar behavioral model using time-series smartphone data. *The Computer Journal*, *63*(7), pp.1109-1123.

[14] Oppermann, R. and Rasher, R., 1997. Adaptability and adaptivity in learning systems. *Knowledge transfer*, *2*, pp.173-179.

[15] Wong, T., Wagner, M. and Treude, C., 2022. Self-adaptive systems: A systematic literature review across categories and domains. *Information and Software Technology*, *148*, p.106934.

[16] Saputri, T.R.D. and Lee, S.W., 2020. The application of machine learning in self-adaptive systems: A systematic literature review. *IEEE Access*, *8*, pp.205948-205967.

[17] Stige, Å., Zamani, E.D., Mikalef, P. and Zhu, Y., 2023. Artificial intelligence (AI) for user experience (UX) design: a systematic literature review and future research agenda. *Information Technology & People*.

[18] Henrichs E, Lesch V, Straesser M, Kounev S, Krupitzer C. A literature review on optimization techniques for adaptation planning in adaptive systems: State of the art and research directions. Information and Software Technology. 2022 Sep 1;149:106940.

[19] Virvou, M., 2023. Artificial Intelligence and User Experience in reciprocity:

Contributions and state of the art. Intelligent Decision Technologies 17 (2023) 73–125

73 DOI 10.3233/IDT-230092 IOS Press

[20] Virvou, M., 2018, July. A new era towards more engaging and human-like computer-

based learning by combining personalisation and artificial intelligence techniques.

In Proceedings of the 23rd Annual ACM Conference on Innovation and Technology in

Computer Science Education (pp. 2-3).
[21] Chrysafiadi, K. and Virvou, M., 2013. Dynamically personalized e-training in computer programming and the language C. IEEE transactions on education, 56(4), pp.385-392.
[22] Chrysafiadi, K., Virvou, M. and Tsihrintzis, G.A., 2022. A fuzzy-based mechanism for automatic personalized assessment in an e-learning system for computer programming. Intelligent Decision Technologies, 16(4), pp.699-714.
[23] Papadimitriou, S., Chrysafiadi, K. and Virvou, M., 2023, July. Adaptive quizzes using fuzzy genetic algorithm. In 2023 14th International Conference on Information, Intelligence, Systems & Applications (IISA) (pp. 1-8). IEEE.
[24] Chrysafiadi, K., Virvou, M., Tsihrintzis, G.A. and Hatzilygeroudis, I., 2023. An Adaptive Learning Environment for Programming Based on Fuzzy Logic and Machine Learning. International Journal on Artificial Intelligence Tools, 32(05), p.2360011.
[25] Alonistioti, N., Tsichrintzi, E.A., Chrysafiadi, K. and Alepis, E., 2023, July. Requirements for Fuzzy Logic in Personalisation of Fire Emergency Alerts. In 2023 14th International Conference on Information, Intelligence, Systems & Applications (IISA) (pp. 1-8). IEEE.
[26] Chrysafiadi, K., Virvou, M., Tsihrintzis, G. A., & Hatzilygeroudis, I. (2023). Evaluating the user's experience, adaptivity and learning outcomes of a fuzzy-based intelligent tutoring system for computer programming for academic students in Greece. Education and Information Technologies, 28(6), 6453-6483.
[27] Chrysafiadi, K., Virvou, M., & Sakkopoulos, E. (2020). Optimizing programming language learning through student modeling in an adaptive web-based educational environment. Machine Learning Paradigms: Advances in Learning Analytics, 205-223.
[28] [7] Caya, R., & Neto, J. J. (2018). A bibliometric review about adaptivity. Procedia computer science, 130, 1114-1119.
[29] Markou, G., Alexiadis, A., & Refanidis, I. (2015). Web Services and Automated Planning for Intelligent Calendars.
[30] Ahuja, P. (2018). Al Planning Assistant for Scheduling Daily Activities.
[31] Madeira, R. N., Santos, P. A., & Correia, N. (2019, December). Using Personalisation to improve User Experience in Public Display Systems with Mobile Interaction. In Proceedings of the 17th International Conference on Advances in Mobile Computing & Multimedia (pp. 3-12).
[32] Hsiao, H. L., & Tang, H. H. (2024, June). A Study on the Application of Generative AI Tools in Assisting the User Experience Design Process. In International Conference on Human-Computer Interaction (pp. 175-189). Cham: Springer Nature Switzerland.
[33] Chanchamnan, P., Ho, C., & San, S. Design in the age of Artificial Intelligence: A literature review on the enhancement of User Experience Design with AI.
[34] Virvou, M., Tsihrintzis, G. A., Bourbakis, N. G., & Jain, L. C. (2022). Handbook on Artificial Intelligence-Empowered Applied Software Engineering: VOL. 2: Smart Software Applications in Cyber-Physical Systems (Vol. 3). Springer International Publishing AG.
[35] Tsihrintzis, G.A., Virvou, M. and Phillips-Wren, G., 2019. Surveys in artificial intelligence-based technologies. Intelligent Decision Technologies, 13(4), pp.393-394.