ΕΦΑΡΜΟΓΗ ΔΙΑΧΕΙΡΙΣΗΣ ΒΙΒΛΙΟΘΗΚΗΣ

Ονοματεπώνυμο: Κων/νος Τερζής 2020168  
Πανεπιστημιακό Τμήμα: Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων  
Πανεπιστήμιο: *Διεθνές Πανεπιστήμιο Ελλάδας (ΔΙ.ΠΑ.Ε)*Πόλη, Χώρα: Θεσσαλονίκη, Ελλάδα  
Email: [terzikk200221@gmail.com](mailto:terzikk200221@gmail.com)

# **ΛΕΙΤΟΥΡΓΙΚΟΤΗΤΑ ΤΗΣ ΕΦΑΡΓΟΦΗΣ**

Η κύρια λειτουργία της εφαρμογής είναι η διαχείριση των διαφόρων πτυχών μιας βιβλιοθήκης, συμπεριλαμβανομένων των βιβλίων, των μελών, των τμημάτων και των δανείων, παρέχοντας στους διαχειριστές ολοκληρωμένα εργαλεία για την αποτελεσματική διαχείριση της βιβλιοθήκης. Οι χρήστες μπορούν να προσθέσουν και να κατηγοριοποιήσουν βιβλία, εισάγοντας βασικές λεπτομέρειες, όπως τίτλο, συγγραφέα, σελίδες, ενώ παράλληλα τα αντιστοιχίζουν σε συγκεκριμένα τμήματα της βιβλιοθήκης για εύκολη καταλογογράφηση και ανάκτηση. Η διαχείριση των μελών απλοποιείται μέσω της εφαρμογής, επιτρέποντας στους διαχειριστές να εγγράφουν νέα μέλη και να διατηρούν μια βάση δεδομένων με βασικές πληροφορίες για εξατομικευμένες αλληλεπιδράσεις. Επιπλέον, η εφαρμογή διευκολύνει τη δημιουργία και την προσαρμογή των τμημάτων της βιβλιοθήκης, προσαρμοζόμενη στις εξελισσόμενες προτιμήσεις των αναγνωστών και βελτιστοποιώντας την κατανομή των πόρων. Οι διαχειριστές μπορούν επίσης να παρακολουθούν και να διαχειρίζονται τα δάνεια που πραγματοποιούνται εντός της βιβλιοθήκης, εξασφαλίζοντας απρόσκοπτες συναλλαγές για τους προστάτες με χαρακτηριστικά όπως υπενθυμίσεις ημερομηνίας λήξης . Μέσω μιας φιλικής προς το χρήστη διεπαφής και ενός ελκυστικού σχεδιασμού, η εφαρμογή στοχεύει στον εκσυγχρονισμό των λειτουργιών της βιβλιοθήκης και στη βελτίωση της εμπειρίας των χρηστών, δίνοντας προτεραιότητα στην απλότητα και την αποτελεσματικότητα σε κάθε πτυχή της λειτουργικότητάς της.

## **ΑΡΧΕΙΑ ΓΙΑ ΕΙΣΟΔΟς ΣΤΗΝ ΕΦΑΡΜΟΓΗ**

**activity\_main.xml:**

Η διάταξη είναι οπτικά δομημένη μέσα σε μια LinearLayout, δίνοντάς της μια καθαρή και οργανωμένη εμφάνιση. Το φόντο κοσμείται με μια ελκυστική εικόνα φόντου, δίνοντας τον τόνο στην εμπειρία του χρήστη. Επιπλέον, το περιεχόμενο είναι κεντραρισμένο κάθετα στην οθόνη, διασφαλίζοντας ότι είναι εύκολα προσβάσιμο και οπτικά ευχάριστο.

Στο κέντρο της οθόνης, οι χρήστες υποδέχονται μια φόρμα σύνδεσης που παρουσιάζεται μέσα σε ένα CardView. Αυτό το CardView παρέχει ένα οπτικά ελκυστικό δοχείο με στρογγυλεμένες γωνίες και ένα διακριτικό εφέ ανύψωσης, κάνοντάς το να ξεχωρίζει στην οθόνη.

Η φόρμα σύνδεσης αποτελείται από έναν τίτλο, "Login", που εμφανίζεται εμφανώς στην κορυφή, υποδεικνύοντας στους χρήστες τον σκοπό της φόρμας. Κάτω από τον τίτλο, οι χρήστες βρίσκουν δύο πεδία εισαγωγής με ετικέτες για το όνομα χρήστη και τον κωδικό πρόσβασης. Αυτά τα πεδία είναι διαισθητικά σχεδιασμένα με συνοδευτικά εικονίδια (ένα εικονίδιο προσώπου για το όνομα χρήστη και ένα εικονίδιο κλειδαριάς για τον κωδικό πρόσβασης), καθιστώντας σαφές ποιες πληροφορίες πρέπει να παρέχουν οι χρήστες. Επιπλέον, τα πεδία περιλαμβάνουν κείμενο υπόδειξης για να καθοδηγήσουν τους χρήστες σχετικά με το τι πρέπει να εισάγουν.

Η φόρμα σύνδεσης ολοκληρώνεται με ένα κουμπί με την ένδειξη "Login", το οποίο είναι σχεδιασμένο με χαρακτηριστικό κίτρινο φόντο και στρογγυλεμένες γωνίες. Αυτό το κουμπί χρησιμεύει ως σαφής πρόσκληση προς δράση, προτρέποντας τους χρήστες να ξεκινήσουν τη διαδικασία ελέγχου ταυτότητας.

**MainActivity.java:**

**Η κλάση MainActivity είναι το κεντρικό στοιχείο που είναι υπεύθυνο για τη διαχείριση του ελέγχου ταυτότητας και της αλληλεπίδρασης των χρηστών εντός της εφαρμογής Library App. Αυτή η κλάση παραπέμπει σε διάφορα στοιχεία UI που ορίζονται στο αρχείο διάταξης activity\_main.xml, διευκολύνοντας την εισαγωγή και την αλληλεπίδραση του χρήστη.**

**Κατά τη δημιουργία της δραστηριότητας, που σηματοδοτείται από τη μέθοδο onCreate, η διάταξη activity\_main.xml φορτώνεται και ορίζεται ως προβολή περιεχομένου. Αυτή η διαδικασία αρχικοποιεί κρίσιμα στοιχεία του περιβάλλοντος εργασίας, όπως τα πεδία EditText για το όνομα χρήστη και τον κωδικό πρόσβασης, μαζί με ένα κουμπί για τη σύνδεση, διασφαλίζοντας ότι η διεπαφή σύνδεσης είναι κατάλληλα διαμορφωμένη για την αλληλεπίδραση του χρήστη.**

**Η κλάση MainActivity ενσωματώνει έναν μηχανισμό για τον έλεγχο ταυτότητας του χρήστη. Ελέγχει την κατάσταση σύνδεσης του χρήστη με πρόσβαση σε ένα αντικείμενο SharedPreferences, το οποίο αποθηκεύει πληροφορίες για την κατάσταση σύνδεσης. Εάν ο χρήστης έχει ήδη πιστοποιηθεί, όπως υποδεικνύεται από την τιμή του κλειδιού "loggedIn" στο SharedPreferences, η MainActivity ολοκληρώνει την εκτέλεσή της, μεταβαίνοντας απρόσκοπτα τον χρήστη στην Drawer δραστηριότητα, που αντιπροσωπεύει τις βασικές λειτουργίες της εφαρμογής της βιβλιοθήκης.**

**Η λειτουργικότητα του κουμπιού σύνδεσης ενσωματώνεται σε έναν ακροατή κλικ. Όταν γίνεται κλικ, αυτός ο ακροατής επικυρώνει το όνομα χρήστη και τον κωδικό πρόσβασης που έχουν εισαχθεί σε σχέση με προκαθορισμένες τιμές. Εάν τα διαπιστευτήρια ταιριάζουν ("admin" και για τα δύο, στην προκειμένη περίπτωση), η σύνδεση θεωρείται επιτυχής. Στη συνέχεια, η εφαρμογή αποθηκεύει την κατάσταση σύνδεσης χρησιμοποιώντας το SharedPreferences, εμφανίζει ένα μήνυμα που επιβεβαιώνει την επιτυχή σύνδεση και πλοηγεί τον χρήστη στην επόμενη σχετική δραστηριότητα, η οποία παρέχει πρόσβαση στα χαρακτηριστικά της βιβλιοθήκης. Σε περίπτωση αποτυχίας σύνδεσης, ένα μήνυμα υποδεικνύει την αποτυχία, προτρέποντας τον χρήστη να επαναλάβει τον έλεγχο ταυτότητας.**

## **ΑΡΧΕΙΑ ΤΟΥ DRAWER MENU**

**nav\_header\_drawer.xml:**

**Το αρχείο nav\_header\_drawer.xml είναι υπεύθυνο για τη δημιουργία της εξατομικευμένης επικεφαλίδας του χρήστη στο συρτάρι πλοήγησης της εφαρμογής Βιβλιοθήκη, προσφέροντας μια φιλόξενη και ενημερωτική εμπειρία.**

**Στην επάνω αριστερή γωνία της κεφαλίδας, εντοπίζετε η εικόνα ενώς ανθρώπινου avatar που εμφανίζεται μέσα σε ένα ImageView. Η εικόνα έχει το κατάλληλο μέγεθος και κοσμείται με γεμίσματα για να εξασφαλίσει μια άνετη εμπειρία προβολής.**

**Δίπλα στην εικόνα προφίλ, υπάρχει ένα TextView που εμφανίζει της θέση του χρήστη στο σύστημα μας, στην περίπτωση μας Admin. Αυτό το στοιχείο κειμένου είναι στρατηγικά τοποθετημένο στα δεξιά της εικόνας προφίλ, διασφαλίζοντας ότι είναι εμφανώς ορατό χωρίς να επισκιάζει την κύρια οπτική του προφίλ. Το κείμενο είναι διαμορφωμένο ώστε να είναι εύκολα αναγνώσιμο, με έντονη γραμματοσειρά και κατάλληλο μέγεθος κειμένου.**

**Ο σχεδιασμός της διάταξης, που χαρακτηρίζεται από ένα κίτρινο χρώμα φόντου, προσθέτει μια πινελιά ζωντάνιας στην κεφαλίδα, ενισχύοντας την οπτική της ελκυστικότητα. Επιπλέον, η διάταξη είναι βελτιστοποιημένη για διάφορα μεγέθη και προσανατολισμούς οθόνης, εξασφαλίζοντας συνεπή παρουσίαση σε διαφορετικές συσκευές.**

**content\_drawer.xml:**

**Το αρχείο content\_drawer.xml ορίζει την κύρια περιοχή περιεχομένου της δραστηριότητας Drawer στην εφαρμογή Library App, όπου παρουσιάζεται στον χρήστη η κύρια πλοήγηση και η λειτουργικότητα της εφαρμογής.**

**Η διάταξη περιλαμβάνει ένα NavHostFragment, το οποίο είναι ένα δοχείο για τη φιλοξενία γραφημάτων πλοήγησης που ορίζονται στο αρχείο navigation/mobile\_navigation.xml. Αυτό το fragment χρησιμεύει ως ο κύριος προορισμός για την πλοήγηση μεταξύ διαφορετικών οθονών ή προορισμών εντός της εφαρμογής.**

**Το NavHostFragment διαμορφώνεται ως ο προεπιλεγμένος ξενιστής πλοήγησης χρησιμοποιώντας το χαρακτηριστικό app:defaultNavHost, υποδεικνύοντας ότι θα χειρίζεται τις ενέργειες πλοήγησης εντός της δραστηριότητας. Επιπλέον, το χαρακτηριστικό app:navGraph καθορίζει το γράφημα πλοήγησης που θα χρησιμοποιηθεί από το NavHostFragment, καθορίζοντας τη δομή και τη ροή της πλοήγησης εντός της εφαρμογής.**

**app\_bar\_drawer.xml:**

**Το αρχείο διάταξης app\_bar\_drawer.xml είναι καθοριστικό για τον καθορισμό της εμφάνισης και της λειτουργικότητας της γραμμής εφαρμογών στη δραστηριότητα Drawer της εφαρμογής.**

**Ο περιέκτης AppBarLayout εντός αυτής της διάταξης περιλαμβάνει βασικά στοιχεία όπως η γραμμή εργαλείων, που χρησιμεύει ως η κύρια γραμμή ενεργειών. Η γραμμή εργαλείων στεγάζει στοιχεία ελέγχου πλοήγησης, συμπεριλαμβανομένου του κουμπιού εναλλαγής του συρταριού πλοήγησης για την πρόσβαση στο συρτάρι πλοήγησης, διευκολύνοντας την απρόσκοπτη πλοήγηση εντός της εφαρμογής. Επίσης, εμφανίζει τον τίτλο της εφαρμογής, παρέχοντας στους χρήστες πληροφορίες σχετικά με την τρέχουσα θέση τους στην ιεραρχία της εφαρμογής.**

**Το αρχείο διάταξης παραπέμπει επίσης στο αρχείο content\_drawer.xml, το οποίο χρησιμεύει ως η κύρια περιοχή περιεχομένου στη δραστηριότητα Drawer.**

**Επιπλέον, η διάταξη περιλαμβάνει ένα FloatingActionButton (FAB), το οποίο συμβολίζεται ως MailButton. Τοποθετημένο στην κάτω δεξιά γωνία, το FAB χρησιμεύει ως ένα εμφανές κουμπί για την εκκίνηση ενεργειών που σχετίζονται με το ηλεκτρονικό ταχυδρομείο, ενισχύοντας την αλληλεπίδραση του χρήστη και παρέχοντας γρήγορη πρόσβαση σε σχετικές λειτουργίες.**

**activity\_drawer.xml:**

**Το αρχείο activity\_drawer.xml ορίζει τη συνολική δομή της δραστηριότητας Drawer, υλοποιώντας ένα πρότυπο συρταριού πλοήγησης. Στον πυρήνα του, χρησιμοποιεί ένα `DrawerLayout`, το οποίο χρησιμεύει ως ο βασικός περιέκτης για τη διάταξη της δραστηριότητας. Εντός αυτής της διάταξης, περιλαμβάνει τη διάταξη `app\_bar\_drawer.xml`, η οποία αναπαριστά την περιοχή της γραμμής εργαλείων και του κύριου περιεχομένου της δραστηριότητας.**

**Επιπλέον, ενσωματώνει ένα `NavigationView`, τοποθετημένο στην αριστερή πλευρά της οθόνης, παρέχοντας επιλογές πλοήγησης στο χρήστη. Η προβολή πλοήγησης περιλαμβάνει διάφορα στοιχεία μενού που καθορίζονται από το αρχείο πόρων μενού `activity\_main\_drawer.xml`. Επιπροσθέτως, η προβολή πλοήγησης περιέχει μια διάταξη κεφαλίδας (`nav\_header\_drawer.xml`) για την εμφάνιση πρόσθετων πληροφοριών ή λεπτομερειών του προφίλ του χρήστη. Συνολικά, αυτό το αρχείο διάταξης αποτελεί το θεμέλιο της δραστηριότητας Drawer, επιτρέποντας την πλοήγηση και την αλληλεπίδραση εντός της εφαρμογής.**

**mobile\_navigation.xml:**

**Το αρχείο mobile\_navigation.xml ορίζει το γράφημα πλοήγησης για την εφαρμογή βιβλιοθήκης, καθορίζοντας τους διάφορους προορισμούς και ενέργειες στη ροή πλοήγησης της εφαρμογής.**

**Ο γράφος πλοήγησης αποτελείται από πολλαπλά στοιχεία <fragment>, καθένα από τα οποία αντιπροσωπεύει έναν ξεχωριστό προορισμό εντός της εφαρμογής. Αυτά τα fragments αντιστοιχούν σε διαφορετικές οθόνες ή λειτουργίες της εφαρμογής της Βιβλιοθήκης, όπως η αρχική οθόνη, η αναζήτηση, η διαχείριση βιβλίων, η διαχείριση μελών, η διαχείριση τμημάτων και οι οθόνες διαχείρισης δανείων. Το σημείο εκκίνησης της ροής πλοήγησης της εφαρμογής ορίζεται στο Home Fragment .**

**Η ενέργεια με popUp ορίζεται εντός της ετικέτας <fragment> για τον προορισμό "nav\_home". Καθορίζει τον τρόπο με τον οποίο η εφαρμογή θα πρέπει να πλοηγείται από την αρχική οθόνη στην οθόνη αναζήτησης. Όταν ενεργοποιείται αυτή η ενέργεια, η εφαρμογή πλοηγείται στον προορισμό "nav\_search" ενώ διαχειρίζεται την πίσω στοίβα.**

**Συγκεκριμένα, τα χαρακτηριστικά της ενέργειας popUp καθορίζουν τη συμπεριφορά της ενέργειας πλοήγησης. Το χαρακτηριστικό "app:popUpTo" καθορίζει τον προορισμό στον οποίο θα πρέπει να πλοηγηθεί η εφαρμογή μετά την εκτέλεση της ενέργειας, ενώ το χαρακτηριστικό "app:popUpToInclusive" καθορίζει εάν ο καθορισμένος προορισμός θα πρέπει να συμπεριληφθεί στη διαδικασία εκκαθάρισης της back stack.**

<fragment  
 android:id="@+id/nav\_home"  
 android:name="com.example.libraryapp.ui.home.HomeFragment"  
 android:label="@string/menu\_home"  
 tools:layout="@layout/fragment\_home">  
 <action  
 android:id="@+id/action\_nav\_home\_to\_nav\_search"  
 app:destination="@id/nav\_search"  
 app:popUpTo="@id/nav\_home"  
 app:popUpToInclusive="true" />  
</fragment>

**activity\_main\_drawer.xml:**

**Το αρχείο activity\_main\_drawer.xml είναι υπεύθυνο για τον ορισμό των στοιχείων μενού του συρταριού πλοήγησης στην εφαρμογή Library, παρουσιάζοντας στους χρήστες διάφορες επιλογές για την πλοήγηση σε διάφορες ενότητες και λειτουργίες.**

**Τα στοιχεία του μενού οργανώνονται σε ομάδες χρησιμοποιώντας την ετικέτα <group>, με κάθε ομάδα να έχει μία μόνο συμπεριφορά που μπορεί να ελεγχθεί. Αυτό διασφαλίζει ότι μόνο ένα στοιχείο μέσα σε κάθε ομάδα μπορεί να επιλεγεί κάθε φορά, παρέχοντας σαφήνεια στην τρέχουσα επιλογή του χρήστη.**

**Η πρώτη ομάδα περιλαμβάνει στοιχεία μενού για τις λειτουργίες αρχικής σελίδας και αναζήτησης, που αντιπροσωπεύονται από εικονίδια και αντίστοιχους τίτλους. Αυτά τα στοιχεία επιτρέπουν στους χρήστες να πλοηγηθούν στην αρχική οθόνη ή να ξεκινήσουν μια αναζήτηση εντός της εφαρμογής.**

**Οι επόμενες ομάδες, όπως τα "Βιβλία", "Μέλη", "Τμήματα" και "Δάνεια", περιέχουν η καθεμία ένα υπομενού . Μέσα σε κάθε υπομενού, τα μεμονωμένα στοιχεία μενού αντιπροσωπεύουν ενέργειες που σχετίζονται με τη διαχείριση βιβλίων, μελών, τμημάτων και δανείων, αντίστοιχα.**

**Drawer.java:**

**Η κλάση " Drawer " χρησιμεύει ως κύρια δραστηριότητα για την εφαρμογή, τη διαχείριση του συρταριού πλοήγησης και την προετοιμασία διαφόρων εξαρτημάτων και υπηρεσιών. Κατά τη δημιουργία, η δραστηριότητα ρυθμίζει τη ρύθμιση παραμέτρων της γραμμής εφαρμογής, τον ελεγκτή περιήγησης και την προβολή περιήγησης. Προετοιμάζει επίσης τη βάση δεδομένων Room και συνδέεται με το Firebase Firestore.**

**Μία από τις βασικές λειτουργίες αυτής της κλάσης είναι ο χειρισμός του ελέγχου ταυτότητας χρήστη. Ελέγχει εάν ο χρήστης είναι συνδεδεμένος μέσω των sharedPreferences. Εάν ο χρήστης δεν είναι συνδεδεμένος, τον ανακατευθύνει στο MainActivity για έλεγχο ταυτότητας.**

**Επιπλέον, η κλάση περιλαμβάνει λειτουργικότητα για την αποστολή μηνυμάτων ηλεκτρονικού ταχυδρομείου, η οποία ενεργοποιείται όταν ο χρήστης κάνει κλικ στο κουμπί αλληλογραφίας. Ξεκινά επίσης μια υπηρεσία παρασκηνίου, "DueDateCheckService", υπεύθυνη για τον έλεγχο των ημερομηνιών λήξης των δανείων.**

**Εξήγηση ειδικού κώδικα:**

binding.appBarDrawer.mailbutton.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View view) {  
 String emailAddress = "librarysupport@gmail.com";  
  
 Intent emailIntent = new Intent(Intent.*ACTION\_SENDTO*);  
 emailIntent.setData(Uri.*parse*("mailto:" + emailAddress));  
 startActivity(emailIntent);  
 }  
});

* **`binding.appBarDrawer.mailbutton.setOnClickListener`: Αυτή η γραμμή ορίζει έναν `OnClickListener` στην προβολή `mailbutton` εντός του `appBarDrawer`, η οποία αποτελεί μέρος της διάταξης που συνδέεται με το αντικείμενο `binding`.**
* **`new View.OnClickListener() {`: Αυτό δημιουργεί μια νέα περίπτωση μιας ανώνυμης εσωτερικής κλάσης που υλοποιεί τη διεπαφή `OnClickListener` για το χειρισμό συμβάντων κλικ.**
* **`public void onClick(View view ) {`: Αυτή η γραμμή ξεκινά τον ορισμό της μεθόδου `onClick`, η οποία θα καλείται όταν γίνεται κλικ στο κουμπί `mailbutton`. Λαμβάνει ως παράμετρο ένα αντικείμενο `View` (στην προκειμένη περίπτωση, το `mailbutton`).**
* **`String emailAddress = "librarysupport @gmail.com";`: Εδώ, δημιουργείται μια μεταβλητή συμβολοσειράς `emailAddress` και αρχικοποιείται με τη διεύθυνση ηλεκτρονικού ταχυδρομείου στην οποία θα σταλεί το μήνυμα ηλεκτρονικού ταχυδρομείου.**
* **`Intent emailIntent = new Intent(Intent.ACTION\_SENDTO);`: Αυτή η γραμμή δημιουργεί ένα νέο αντικείμενο `Intent`, καθορίζοντας ότι πρόκειται για μια ενέργεια για την αποστολή δεδομένων σε κάποιον μέσω ηλεκτρονικού ταχυδρομείου.**
* **`emailIntent.setData(Uri.parse("mailto:" + emailAddress));`: Αυτό ορίζει τα δεδομένα για την πρόθεση ηλεκτρονικού ταχυδρομείου. Δημιουργεί ένα αντικείμενο `Uri` χρησιμοποιώντας την `Uri.parse()` με το σχήμα `mailto:` και προσθέτει τη μεταβλητή `emailAddress` σε αυτό, υποδεικνύοντας τη διεύθυνση ηλεκτρονικού ταχυδρομείου του παραλήπτη.**
* **`startActivity(emailIntent);`: Τέλος, αυτή η γραμμή ξεκινά τη δραστηριότητα που καθορίζεται από το `emailIntent`, το οποίο στην προκειμένη περίπτωση είναι ένας πελάτης ηλεκτρονικού ταχυδρομείου με προσυμπληρωμένη τη διεύθυνση ηλεκτρονικού ταχυδρομείου του παραλήπτη.**

**Συνολικά, αυτός ο κώδικας δημιουργεί έναν ακροατή κλικ για την προβολή `mailbutton`. Όταν πατηθεί, ξεκινάει ένα πρόγραμμα-πελάτη ηλεκτρονικού ταχυδρομείου με ένα νέο μήνυμα ηλεκτρονικού ταχυδρομείου που απευθύνεται στον καθορισμένο παραλήπτη.**

## **ΑΡΧΕΙΑ ΓΙΑ ΑΡΧΙΚΗ ΟΘΟΝΗ**

**fragment\_home.xml:**

**Το αρχείο fragment\_home.xml, ορίζει τη διάταξη για την αρχική οθόνη της εφαρμογής βιβλιοθήκης. Χρησιμοποιεί ένα ConstraintLayout ως βασική διάταξη, προσφέροντας ισχυρούς περιορισμούς για τη διάταξη των θυγατρικών προβολών του.**

**Στο κέντρο της διάταξης βρίσκεται ένα TextView με ένα φιλόξενο μήνυμα καλωσορίσματος προς τους χρήστες. Αυτό το TextView είναι οριζόντια κεντραρισμένο και τοποθετημένο ελαφρώς κάτω από την κορυφή της οθόνης, δημιουργώντας έναν οπτικά ελκυστικό χαιρετισμό. Το μήνυμα καλωσορίσματος είναι διαμορφωμένο με λευκό χρώμα κειμένου, μεγαλύτερο μέγεθος γραμματοσειράς και κεντραρισμένη στοίχιση για βελτιωμένη αναγνωσιμότητα.**

**Κάτω από το μήνυμα καλωσορίσματος υπάρχει ένα Κουμπί με την ένδειξη " Explore ", το οποίο προσκαλεί τους χρήστες να ανακαλύψουν περαιτέρω την εφαρμογή της βιβλιοθήκης. Αυτό το κουμπί είναι οριζόντια κεντραρισμένο και τοποθετημένο κάτω από το μήνυμα καλωσορίσματος με ένα μικρό περιθώριο, εξασφαλίζοντας έναν σαφή οπτικό διαχωρισμό μεταξύ των δύο στοιχείων. Το χρώμα φόντου του κουμπιού είναι κίτρινο, παρέχοντας ένα ζωντανό και ελκυστικό οπτικό στοιχείο. Επιπλέον, το χρώμα του κειμένου έχει οριστεί σε λευκό για καλύτερη αντίθεση και αναγνωσιμότητα σε σχέση με το φόντο.**

**HomeFragment.java:**

**Το αρχείο HomeFragment.java είναι υπεύθυνο για την εμφάνιση της αρχικής οθόνης της εφαρμογής βιβλιοθήκης. Φουσκώνει τη διάταξη που ορίζεται στο αρχείο fragment\_home.xml, που περιέχει στοιχεία διεπαφής χρήστη, όπως ένα μήνυμα καλωσορίσματος και ένα κουμπί "Explore". Όταν οι χρήστες κάνουν κλικ στο κουμπί "Εξερεύνηση", το θραύσμα τους πλοηγεί στη λειτουργία αναζήτησης της εφαρμογής.**

## **ΑΡΧΕΙΑ ΓΙΑ ΔΙΑΧΕΙΡΗΣΗ ΒΙΒΛΙΩΝ**

**fragment\_insert\_book.xml:**

**Το αρχείο fragment\_insert\_book.xml ορίζει το UI μιας φόρμας που χρησιμοποιείται για την εισαγωγή λεπτομερειών βιβλίου. Χρησιμοποιεί ένα `RelativeLayout` ως βασική διάταξη, επιτρέποντας την ευέλικτη τοποθέτηση των θυγατρικών προβολών.**

**Μέσα στο `RelativeLayout`, υπάρχει ένα `CardView` με στρογγυλεμένες γωνίες και μια μικρή ανύψωση για να του δώσει μια εμφάνιση που μοιάζει με κάρτα. Αυτό το `CardView` χρησιμεύει ως περιέκτης για τα στοιχεία της φόρμας.**

**Μέσα στο `CardView`, υπάρχει ένα `LinearLayout` τοποθετημένο κάθετα για να περιέχει διάφορα πεδία εισαγωγής και ένα κουμπί. Κάθε πεδίο εισόδου είναι τυλιγμένο σε ένα `TextInputLayout`, το οποίο παρέχει μια είσοδο κειμένου σε στυλ Material Design με αιωρούμενη ετικέτα. Υπάρχουν τρία πεδία εισαγωγής κειμένου για τον τίτλο του βιβλίου, τον συγγραφέα και τον αριθμό των σελίδων, αντίστοιχα.**

**Επιπλέον, υπάρχει ένα TextView που προτρέπει τον χρήστη να επιλέξει ένα τμήμα της βιβλιοθήκης, ακολουθούμενο από ένα Spinner widget για την επιλογή τμήματος. Το Spinner είναι διαμορφωμένο με ένα προσαρμοσμένο drawable περιγράμματος.**

**Τέλος, υπάρχει ένα LinearLayout για να περιέχει το κουμπί εισαγωγής, το οποίο είναι ένα κουμπί Material Design (`MaterialButton`). Αυτό το κουμπί επιτρέπει στους χρήστες να υποβάλλουν τη φόρμα και να εισάγουν τα στοιχεία του βιβλίου στην εφαρμογή.**

**fragment\_update\_book.xml:**

**Το αρχείο fragment\_update\_book.xml, περιγράφει τη διεπαφή χρήστη μιας φόρμας που χρησιμοποιείται για την ενημέρωση των στοιχείων ενός βιβλίου . Χρησιμοποιεί ένα RelativeLayout ως βασική διάταξη, παρέχοντας ευέλικτη διάταξη των θυγατρικών προβολών του.**

**Μέσα στο RelativeLayout περιέχεται ένα CardView που κοσμείται με στρογγυλεμένες γωνίες και μια λεπτή ανύψωση, προσδίδοντας μια όψη κάρτας. Αυτό το CardView λειτουργεί ως υποδοχή για τα στοιχεία της φόρμας.**

**Ενσωματωμένο μέσα στο CardView βρίσκεται ένα κάθετο LinearLayout που είναι επιφορτισμένο με τη στέγαση διαφόρων πεδίων εισόδου και ενός κουμπιού. Κάθε πεδίο εισαγωγής περικλείεται μέσα σε ένα TextInputLayout, παρέχοντας μια είσοδο κειμένου διαμορφωμένη σύμφωνα με τις αρχές του Material Design, με μια αιωρούμενη ετικέτα. Συγκεκριμένα, υπάρχουν πεδία εισαγωγής κειμένου που προορίζονται για τον τίτλο του βιβλίου, τον συγγραφέα και τον αριθμό των σελίδων, αντίστοιχα.**

**Επιπλέον, υπάρχει ένα TextView που καλεί τον χρήστη να ορίσει ένα τμήμα της βιβλιοθήκης, το οποίο διαδέχεται ένα widget Spinner προσαρμοσμένο για την επιλογή τμήματος. Το Spinner είναι προσαρμοσμένο με ένα προσαρμοσμένο σχεδιάσιμο πλαίσιο.**

**Τέλος, ένα LinearLayout ορίζεται για να φιλοξενήσει το κουμπί ενημέρωσης, ένα κουμπί σχεδιασμού υλικού (MaterialButton). Αυτό το κουμπί δίνει τη δυνατότητα στους χρήστες να αποστέλλουν τη φόρμα και να πραγματοποιούν ενημερώσεις στις λεπτομέρειες του βιβλίου μέσα στην εφαρμογή.**

**fragment\_delete\_book.xml:**

**Το αρχείο fragment\_delete\_book.xml, περιγράφει τη διεπαφή χρήστη μιας φόρμας που χρησιμοποιείται για τη διαγραφή εγγραφών βιβλίου. Υιοθετεί ένα RelativeLayout ως βασική διάταξη, επιτρέποντας την ευέλικτη διάταξη των θυγατρικών προβολών του.**

**Μέσα στο RelativeLayout περικλείεται ένα CardView που κοσμείται με στρογγυλεμένες γωνίες και μια λεπτή ανύψωση, προσδίδοντας μια εμφάνιση που μοιάζει με κάρτα. Αυτό το CardView χρησιμεύει ως δοχείο για τα στοιχεία της φόρμας.**

**Ενσωματωμένο μέσα στο CardView είναι ένα κατακόρυφο LinearLayout σχεδιασμένο να φιλοξενεί διάφορα στοιχεία, όπως ένα ΤextView και ένα Spinner. Η προβολή κειμένου ζητά από το χρήστη να επιλέξει ένα βιβλίο, ενώ το περιστρεφόμενο επιτρέπει την επιλογή μιας υπάρχουσας καταχώρησης βιβλίου. Επιπλέον, παρέχεται ένα κουμπί σχεδιασμού υλικού (MaterialButton) για την έναρξη της διαδικασίας διαγραφής.**

**InsertBookFragment.java:**

**Η κλάση "InsertBookFragment" είναι υπεύθυνη για το χειρισμό της εισαγωγής νέων εγγραφών βιβλίων στη βάση δεδομένων της βιβλιοθήκης.**

**Κατά τη δημιουργία, το fragment διογκώνει τη διάταξή του και αρχικοποιεί διάφορα στοιχεία του UI, όπως πεδία EditText για την εισαγωγή λεπτομερειών του βιβλίου, ένα Spinner για την επιλογή του τμήματος της βιβλιοθήκης και ένα Button για την ενεργοποίηση της διαδικασίας εισαγωγής.**

**Το fragment γεμίζει το Spinner με τα ονόματα των διαθέσιμων τμημάτων της βιβλιοθήκης που ανακτώνται από τη βάση δεδομένων χρησιμοποιώντας την περίπτωση της βάσης δεδομένων Room της κλάσης Drawer. Ρυθμίζει ένα ArrayAdapter για την εμφάνιση των ονομάτων των τμημάτων στο αναπτυσσόμενο μενού του Spinner.**

**Όταν ο χρήστης κάνει κλικ στο κουμπί εισαγωγής, καλείται η μέθοδος insertBook(). Αυτή η μέθοδος ανακτά τις τιμές εισόδου από τα πεδία EditText, τις επικυρώνει και στη συνέχεια δημιουργεί ένα νέο αντικείμενο Book με τα παρεχόμενα στοιχεία. Διασφαλίζει ότι όλα τα απαιτούμενα πεδία είναι συμπληρωμένα και ότι έχει επιλεγεί ένα έγκυρο τμήμα πριν προχωρήσει στην εισαγωγή.**

**Εάν η εισαγωγή είναι επιτυχής, εμφανίζεται ένα μήνυμα επιτυχίας χρησιμοποιώντας μια ειδοποίηση Toast. Εάν προκύψει σφάλμα κατά την εισαγωγή, εμφανίζεται ένα μήνυμα σφάλματος για να ειδοποιηθεί ο χρήστης.**

**UpdateBookFragment.java:**

**Η κλάση "UpdateBookFragment" διευκολύνει την ενημέρωση υφιστάμενων εγγραφών βιβλίων στη βάση δεδομένων της βιβλιοθήκης.**

**Κατά τη δημιουργία του, το fragment φουσκώνει τη διάταξή του και αρχικοποιεί διάφορα στοιχεία του UI, όπως πεδία EditText για την επεξεργασία των στοιχείων του βιβλίου, Spinners για την επιλογή του βιβλίου και του τμήματος της βιβλιοθήκης προς ενημέρωση και ένα Button για την ενεργοποίηση της διαδικασίας ενημέρωσης.**

**Το fragment αρχικοποιεί τα Spinners με τους τίτλους των υπαρχόντων βιβλίων και τα ονόματα των τμημάτων της βιβλιοθήκης που ανακτώνται από τη βάση δεδομένων χρησιμοποιώντας την περίπτωση της βάσης δεδομένων Room της κλάσης Drawer.**

**Όταν ο χρήστης κάνει κλικ στο κουμπί ενημέρωσης, καλείται η μέθοδος updateBook(). Αυτή η μέθοδος ανακτά τις τιμές εισόδου από τα πεδία EditText και τα Spinners, τις επικυρώνει και στη συνέχεια δημιουργεί ένα νέο αντικείμενο Book με τα ενημερωμένα στοιχεία. Διασφαλίζει ότι ένα βιβλίο και ένα τμήμα είναι επιλεγμένα πριν προχωρήσει στην ενημέρωση.**

**Εάν η ενημέρωση είναι επιτυχής, εμφανίζεται ένα μήνυμα επιτυχίας χρησιμοποιώντας μια ειδοποίηση Toast. Εάν προκύψει σφάλμα κατά τη διαδικασία ενημέρωσης, εμφανίζεται ένα μήνυμα σφάλματος για να ειδοποιήσει τον χρήστη.**

**DeleteBookFragment.java:**

**Η κλάση "DeleteBookFragment" παρέχει λειτουργικότητα για τη διαγραφή υφιστάμενων εγγραφών βιβλίων από τη βάση δεδομένων της βιβλιοθήκης.**

**Κατά τη δημιουργία, το fragment διογκώνει τη διάταξή του και αρχικοποιεί ένα Spinner για την εμφάνιση των τίτλων των υπαρχόντων βιβλίων. Οι χρήστες μπορούν να επιλέξουν ένα βιβλίο από το Spinner, και με το πάτημα του κουμπιού delete, το επιλεγμένο βιβλίο διαγράφεται από τη βάση δεδομένων.**

**Το fragment αρχικοποιεί το Spinner με τους τίτλους των υπαρχόντων βιβλίων που ανακτώνται από τη βάση δεδομένων χρησιμοποιώντας την περίπτωση της βάσης δεδομένων Room της κλάσης Drawer. Όταν πατηθεί το κουμπί διαγραφής, ενεργοποιείται η μέθοδος onClick(), η οποία ανακτά το αναγνωριστικό του επιλεγμένου βιβλίου από το Spinner, δημιουργεί ένα νέο αντικείμενο Book με το αναγνωριστικό και στη συνέχεια καλεί τη μέθοδο deleteBook() του BookDao για να διαγράψει το βιβλίο από τη βάση δεδομένων.**

**Εάν η διαγραφή είναι επιτυχής, εμφανίζεται ένα μήνυμα επιτυχίας χρησιμοποιώντας μια ειδοποίηση Toast. Εάν προκύψει σφάλμα κατά τη διαδικασία διαγραφής, εμφανίζεται ένα μήνυμα σφάλματος για να ειδοποιηθεί ο χρήστης.**

## **ΑΡΧΕΙΑ ΓΙΑ ΔΙΑΧΕΙΡΗΣΗ ΜΕΛΩΝ**

**fragment\_insert\_member.xml:**

**Το τμήμα XML fragment\_insert\_member.xml, ορίζει τη διεπαφή χρήστη μιας φόρμας που χρησιμοποιείται για την εισαγωγή στοιχείων μέλους . Χρησιμοποιεί ένα RelativeLayout ως βασική διάταξη, παρέχοντας ευελιξία στην τοποθέτηση των θυγατρικών προβολών του.**

**Μέσα στο RelativeLayout περιέχεται ένα CardView με στρογγυλεμένες γωνίες και ελαφρά ανύψωση, δημιουργώντας μια εμφάνιση που μοιάζει με κάρτα. Αυτό το CardView λειτουργεί ως περιέκτης για τα στοιχεία της φόρμας.**

**Ενσωματωμένο μέσα στο CardView είναι ένα κατακόρυφο LinearLayout σχεδιασμένο να φιλοξενεί διάφορα στοιχεία, συμπεριλαμβανομένων πεδίων κειμένου και ενός κουμπιού. Το TextView προτρέπει τον χρήστη να συμπληρώσει τα στοιχεία του μέλους, ενώ τα πεδία κειμένου που είναι τυλιγμένα σε TextInputLayout παρέχουν είσοδο για το όνομα και τη διεύθυνση ηλεκτρονικού ταχυδρομείου του μέλους. Επιπλέον, παρέχεται ένα κουμπί σχεδιασμού υλικού (MaterialButton) για την έναρξη της διαδικασίας εισαγωγής μέλους.**

**fragment\_update\_member.xml:**

**Το αρχείο fragment\_update\_member.xml, περιγράφει τη διεπαφή χρήστη μιας φόρμας που χρησιμοποιείται για την ενημέρωση των στοιχείων των μελών . Υιοθετεί ένα RelativeLayout ως βασική διάταξη, επιτρέποντας την ευέλικτη διάταξη των θυγατρικών προβολών του.**

**Μέσα στο RelativeLayout περικλείεται ένα CardView με στρογγυλεμένες γωνίες και διακριτική ανύψωση, δημιουργώντας μια παρουσίαση που μοιάζει με κάρτα. Αυτό το CardView χρησιμεύει ως δοχείο για τα στοιχεία της φόρμας.**

**Ενσωματωμένο μέσα στο CardView είναι ένα κάθετο LinearLayout προσαρμοσμένο για να φιλοξενεί διάφορα στοιχεία, όπως πεδία κειμένου και ένα κουμπί. Ενα TextView προτρέπει τον χρήστη να επιλέξει ένα μέλος, ακολουθούμενη από ένα widget Spinner για την επιλογή ενός υπάρχοντος μέλους. Στη συνέχεια, υπάρχει ένα TextView που δίνει οδηγίες στους χρήστες να εισάγουν νέα στοιχεία, ακολουθούμενη από πεδία εισαγωγής κειμένου τυλιγμένα σε TextInputLayout για την ενημέρωση του ονόματος και της διεύθυνσης ηλεκτρονικού ταχυδρομείου του μέλους. Τέλος, περιλαμβάνεται ένα κουμπί Material Design (MaterialButton) για την έναρξη της διαδικασίας ενημέρωσης του μέλους.**

**fragment\_delete\_member.xml:**

**Το αρχείο fragment\_delete\_member.xml, περιγράφει τη διεπαφή χρήστη μιας φόρμας που χρησιμοποιείται για τη διαγραφή καταχωρήσεων μελών . Χρησιμοποιεί ένα RelativeLayout ως βασική διάταξη, παρέχοντας ευελιξία στη διάταξη των θυγατρικών προβολών του.**

**Μέσα στο RelativeLayout περιέχεται ένα CardView με στρογγυλεμένες γωνίες και ένα διακριτικό εφέ ανύψωσης, με αποτέλεσμα μια εμφάνιση που μοιάζει με κάρτα. Αυτό το CardView χρησιμεύει ως δοχείο για τα στοιχεία της φόρμας.**

**Ενσωματωμένο μέσα στο CardView είναι ένα κατακόρυφο LinearLayout σχεδιασμένο να φιλοξενεί διάφορα στοιχεία, όπως TextViews, ένα Spinner και ένα κουμπί. Ενα TextView προτρέπει τον χρήστη να επιλέξει ένα μέλος, ακολουθούμενη από ένα widget Spinner που επιτρέπει στους χρήστες να επιλέξουν ένα υπάρχον μέλος. Επιπλέον, περιλαμβάνεται ένα κουμπί Material Design (MaterialButton) για να διευκολύνει τη διαδικασία διαγραφής μέλους.**

**InsertMemberFragment.java:**

**Η κλάση "InsertMemberFragment" διευκολύνει την εισαγωγή νέων εγγραφών μελών στη βάση δεδομένων της βιβλιοθήκης. Κατά τη δημιουργία, το τμήμα διογκώνει τη διάταξή του, η οποία περιλαμβάνει πεδία εισόδου για την εισαγωγή του ονόματος και της διεύθυνσης ηλεκτρονικού ταχυδρομείου του μέλους, καθώς και ένα κουμπί για την εισαγωγή του μέλους στη βάση δεδομένων.**

**Όταν γίνεται κλικ στο κουμπί εισαγωγής, ενεργοποιείται η μέθοδος onClick(). Η μέθοδος ελέγχει πρώτα αν τα πεδία όνομα και email είναι συμπληρωμένα. Εάν κάποιο από τα πεδία είναι κενό, ένα μήνυμα Toast προτρέπει τον χρήστη να συμπληρώσει όλα τα πεδία. Διαφορετικά, η μέθοδος ανακτά τις τιμές του ονόματος και του email που έχουν εισαχθεί, δημιουργεί ένα νέο αντικείμενο Member, θέτει τα χαρακτηριστικά του αναλόγως και το εισάγει στη βάση δεδομένων χρησιμοποιώντας την περίπτωση της βάσης δεδομένων Room της κλάσης Drawer.**

**Εάν η εισαγωγή είναι επιτυχής, εμφανίζεται ένα μήνυμα επιτυχίας χρησιμοποιώντας μια ειδοποίηση Toast. Εάν προκύψει σφάλμα κατά τη διαδικασία εισαγωγής, εμφανίζεται ένα μήνυμα σφάλματος για να ειδοποιήσει τον χρήστη.**

**UpdateMemberFragment.java:**

**Η κλάση "UpdateMemberFragment" παρέχει λειτουργικότητα για την ενημέρωση υφιστάμενων εγγραφών μελών στη βάση δεδομένων της βιβλιοθήκης. Αυτό το fragment διογκώνει τη διάταξή του, το οποίο περιέχει ένα spinner για την επιλογή του μέλους προς ενημέρωση, πεδία εισαγωγής για την τροποποίηση του ονόματος και της διεύθυνσης ηλεκτρονικού ταχυδρομείου του μέλους, μαζί με ένα κουμπί για την έναρξη της διαδικασίας ενημέρωσης.**

**Με το πάτημα του κουμπιού ενημέρωσης, καλείται η μέθοδος onClick(). Αρχικά, ελέγχει αν τα πεδία όνομα και email είναι συμπληρωμένα. Εάν κάποιο από τα πεδία είναι κενό, ένα μήνυμα Toast προτρέπει τον χρήστη να συμπληρώσει όλα τα πεδία. Διαφορετικά, η μέθοδος ανακτά τις εισαχθείσες τιμές ονόματος και ηλεκτρονικού ταχυδρομείου, καθώς και το αναγνωριστικό του επιλεγμένου μέλους από το spinner.**

**Στη συνέχεια, δημιουργεί ένα νέο αντικείμενο Member με τα ενημερωμένα στοιχεία, συμπεριλαμβανομένου του ανακτηθέντος αναγνωριστικού μέλους. Στη συνέχεια, επιχειρεί να ενημερώσει το μέλος στη βάση δεδομένων χρησιμοποιώντας την περίπτωση της βάσης δεδομένων Room της κλάσης Drawer. Εάν η ενημέρωση είναι επιτυχής, εμφανίζεται ένα μήνυμα επιτυχίας χρησιμοποιώντας μια ειδοποίηση Toast. Εάν προκύψει σφάλμα κατά τη διάρκεια της διαδικασίας ενημέρωσης, εμφανίζεται ένα μήνυμα σφάλματος για να ειδοποιήσει τον χρήστη.**

**DeleteMemberFragment.java:**

**Η κλάση "DeleteMemberFragment" διευκολύνει τη διαγραφή εγγραφών μελών από τη βάση δεδομένων της βιβλιοθήκης. Αυτό το τμήμα περιέχει ένα spinner για την επιλογή του μέλους προς διαγραφή και ένα κουμπί για την έναρξη της διαδικασίας διαγραφής.**

**Κατά τον πληθωρισμό της διάταξής του, το θραύσμα αρχικοποιεί το spinner συμπληρώνοντάς το με τα ονόματα των υπαρχόντων μελών που ανακτώνται από τη βάση δεδομένων. Διατηρεί επίσης έναν κατάλογο αναγνωριστικών μελών που αντιστοιχούν στα επιλεγμένα ονόματα μελών στο περιστρεφόμενο.**

**Όταν γίνεται κλικ στο κουμπί διαγραφής, καλείται η μέθοδος onClick(). Ανακτά τη θέση του επιλεγμένου μέλους από το spinner και εξάγει το αντίστοιχο αναγνωριστικό μέλους από τη λίστα αναγνωριστικών μελών. Στη συνέχεια, δημιουργεί ένα νέο αντικείμενο Member με το ανακτημένο αναγνωριστικό μέλους και καλεί τη μέθοδο deleteMember() από το παράδειγμα της βάσης δεδομένων Room για να διαγράψει την εγγραφή μέλους.**

**Εάν η διαγραφή είναι επιτυχής, εμφανίζεται ένα μήνυμα επιτυχίας χρησιμοποιώντας μια ειδοποίηση Toast. Εάν προκύψει σφάλμα κατά τη διαδικασία διαγραφής, εμφανίζεται ένα μήνυμα σφάλματος για να ειδοποιηθεί ο χρήστης.**

## **ΑΡΧΕΙΑ ΓΙΑ ΔΙΑΧΕΙΡΗΣΗ ΤΜΗΜΑΤΩΝ ΒΙΒΛΙΟΘΗΚΗς**

**fragment\_insert\_section.xml:**

**Το αρχείο fragment\_insert\_section.xml, ορίζει τη διεπαφή χρήστη μιας φόρμας που χρησιμοποιείται για την εισαγωγή λεπτομερειών τμήματος . Χρησιμοποιεί ένα RelativeLayout ως βασική διάταξη, παρέχοντας ευελιξία στην τοποθέτηση των θυγατρικών προβολών του.**

**Μέσα στο RelativeLayout περιέχεται ένα CardView με στρογγυλεμένες γωνίες και ελαφρά ανύψωση, δημιουργώντας μια εμφάνιση που μοιάζει με κάρτα. Αυτό το CardView λειτουργεί ως περιέκτης για τα στοιχεία της φόρμας.**

**Ενσωματωμένο μέσα στο CardView είναι ένα κατακόρυφο LinearLayout σχεδιασμένο να φιλοξενεί διάφορα στοιχεία, συμπεριλαμβανομένων πεδίων κειμένου και ενός κουμπιού. Ενα TextView προτρέπει τον χρήστη να συμπληρώσει τις λεπτομέρειες του τμήματος, ενώ τα πεδία κειμένου που είναι τυλιγμένα σε TextInputLayout παρέχουν είσοδο για το όνομα και τη θέση του τμήματος. Επιπλέον, παρέχεται ένα κουμπί σχεδιασμού υλικού (MaterialButton) για την έναρξη της διαδικασίας εισαγωγής τμήματος.**

**fragment\_update\_section.xml:**

**Το αρχείο fragment\_update\_section.xml, ορίζει τη διεπαφή χρήστη μιας φόρμας που χρησιμοποιείται για την ενημέρωση των στοιχείων της ενότητας σε μια εφαρμογή Android. Χρησιμοποιεί ένα RelativeLayout ως βασική διάταξη, επιτρέποντας την ευέλικτη τοποθέτηση των θυγατρικών προβολών του.**

**Εντός του RelativeLayout, υπάρχει ένα CardView που ενθυλακώνει τα στοιχεία της φόρμας. Αυτό το CardView έχει στρογγυλεμένες γωνίες και μια ελαφρά ανύψωση, δίνοντάς του μια εμφάνιση που μοιάζει με κάρτα.**

**Μέσα στο CardView, ένα κάθετο LinearLayout τοποθετεί διάφορα στοιχεία, συμπεριλαμβανομένων των πεδίων κειμένου και ενός κουμπιού. Ένα TextView προτρέπει τον χρήστη να επιλέξει ένα τμήμα, ακολουθούμενο από ένα widget Spinner για την επιλογή ενός υπάρχοντος τμήματος. Παρακάτω, υπάρχουν TextInputLayouts για την εισαγωγή του ενημερωμένου ονόματος και της θέσης του τμήματος. Κάθε TextInputLayout παρέχει ένα πεδίο εισαγωγής με κυμαινόμενες ετικέτες, τηρώντας τις οδηγίες του Material Design. Τέλος, ένα κουμπί MaterialButton επιτρέπει στους χρήστες να υποβάλουν τις ενημερωμένες λεπτομέρειες του τμήματος.**

**fragment\_delete\_section.xml:**

**Το αρχείο fragment\_delete\_section.xml, ορίζει τη διεπαφή χρήστη μιας φόρμας που χρησιμοποιείται για τη διαγραφή στοιχείων τμήματος. Χρησιμοποιεί ένα RelativeLayout ως βασική διάταξη, προσφέροντας ευελιξία στη διάταξη των θυγατρικών προβολών του.**

**Μέσα στο RelativeLayout περικλείεται ένα CardView που χρησιμεύει ως περιέκτης για τα στοιχεία της φόρμας. Αυτό το CardView παρουσιάζει στρογγυλεμένες γωνίες και μια διακριτική ανύψωση, ενισχύοντας την οπτική του ελκυστικότητα.**

**Ενσωματωμένο μέσα στο CardView είναι ένα κάθετο LinearLayout υπεύθυνο για την οργάνωση των διαφόρων στοιχείων, συμπεριλαμβανομένων των προβολών κειμένου, ενός περιστροφικού μηχανισμού και ενός κουμπιού. Ένα TextView προτρέπει τον χρήστη να επιλέξει το τμήμα που πρόκειται να διαγραφεί, ακολουθούμενο από ένα widget Spinner που παρουσιάζει τις επιλογές των υφιστάμενων τμημάτων. Παρακάτω, παρέχεται ένα MaterialButton για την έναρξη της διαδικασίας διαγραφής.**

**InsertSectionFragment.java:**

**Η κλάση InsertSectionFragment είναι υπεύθυνη για την προσθήκη νέων τμημάτων στη βάση δεδομένων της βιβλιοθήκης. Παρέχει μια διεπαφή με πεδία εισαγωγής για το όνομα και τη θέση του τμήματος, μαζί με ένα κουμπί για την υποβολή της εισαγωγής.**

**Κατά τον πληθωρισμό της διάταξής του, το θραύσμα αρχικοποιεί τα πεδία EditText για την καταγραφή του ονόματος και της θέσης του τμήματος. Ρυθμίζει επίσης έναν ακροατή κλικ για το κουμπί εισαγωγής.**

**Όταν γίνεται κλικ στο κουμπί εισαγωγής, καλείται η μέθοδος onClick(). Αρχικά επικυρώνει αν τα πεδία όνομα τμήματος και τοποθεσία είναι συμπληρωμένα. Εάν κάποιο από αυτά είναι κενό, ένα μήνυμα Toast προτρέπει τον χρήστη να συμπληρώσει όλα τα πεδία.**

**Εάν και τα δύο πεδία είναι συμπληρωμένα, η μέθοδος ανακτά το κείμενο που έχει εισαχθεί στα πεδία EditText, δημιουργεί ένα νέο αντικείμενο LibrarySection με το όνομα και τη θέση που δόθηκε και το εισάγει στη βάση δεδομένων χρησιμοποιώντας τη μέθοδο sectionDao().insertSection().**

**Εάν η εισαγωγή είναι επιτυχής, εμφανίζεται ένα μήνυμα επιτυχίας χρησιμοποιώντας μια ειδοποίηση Toast. Εάν προκύψει σφάλμα κατά τη διαδικασία εισαγωγής, εμφανίζεται ένα μήνυμα σφάλματος για να ειδοποιηθεί ο χρήστης.**

**UpdateSectionFragment.java:**

**Η κλάση UpdateSectionFragment διευκολύνει την τροποποίηση υφιστάμενων τμημάτων στη βάση δεδομένων της βιβλιοθήκης. Περιλαμβάνει πεδία εισαγωγής για την ενημέρωση του ονόματος και της θέσης του τμήματος, μαζί με ένα spinner για την επιλογή του τμήματος προς ενημέρωση και ένα κουμπί για την υποβολή των αλλαγών.**

**Κατά τον πληθωρισμό της διάταξής του, το fragment αρχικοποιεί τα πεδία EditText για την καταγραφή του ενημερωμένου ονόματος και της θέσης του τμήματος. Ρυθμίζει επίσης ένα spinner για να επιτρέπει στους χρήστες να επιλέγουν το τμήμα που επιθυμούν να ενημερώσουν. Το spinner συμπληρώνεται με τα ονόματα όλων των υφιστάμενων τμημάτων που ανακτήθηκαν από τη βάση δεδομένων.**

**Όταν γίνεται κλικ στο κουμπί ενημέρωσης, ενεργοποιείται η μέθοδος onClick(). Αρχικά ελέγχει αν τα πεδία όνομα τμήματος και τοποθεσία είναι συμπληρωμένα. Εάν κάποιο από αυτά είναι κενό, ένα μήνυμα toast προτρέπει τον χρήστη να συμπληρώσει όλα τα πεδία.**

**Εάν και τα δύο πεδία είναι συμπληρωμένα, η μέθοδος ανακτά το κείμενο που έχει εισαχθεί στα πεδία EditText, μαζί με το αναγνωριστικό του επιλεγμένου τμήματος από το spinner. Στη συνέχεια δημιουργεί ένα νέο αντικείμενο LibrarySection με τα ενημερωμένα στοιχεία και ενημερώνει το αντίστοιχο τμήμα στη βάση δεδομένων χρησιμοποιώντας τη μέθοδο sectionDao().updateSection().**

**Εάν η λειτουργία ενημέρωσης είναι επιτυχής, εμφανίζεται ένα μήνυμα επιτυχίας χρησιμοποιώντας μια ειδοποίηση Toast. Εάν προκύψει κάποιο σφάλμα κατά τη διαδικασία ενημέρωσης, εμφανίζεται ένα μήνυμα σφάλματος για να ειδοποιήσει τον χρήστη.**

**DeleteSectionFragment.java:**

**Η κλάση DeleteSectionFragment επιτρέπει στους χρήστες να αφαιρούν υπάρχοντα τμήματα από τη βάση δεδομένων της βιβλιοθήκης. Διαθέτει ένα περιστρεφόμενο κουμπί για την επιλογή του τμήματος προς διαγραφή και ένα κουμπί για την επιβεβαίωση της διαγραφής.**

**Κατά τον πληθωρισμό της διάταξής του, το fragment αρχικοποιεί το spinner για να το γεμίσει με τα ονόματα όλων των υπαρχόντων τμημάτων που ανακτήθηκαν από τη βάση δεδομένων.**

**Όταν πατηθεί το κουμπί διαγραφής, ενεργοποιείται η μέθοδος onClick(). Αρχικά ανακτά το αναγνωριστικό του επιλεγμένου τμήματος από το spinner. Στη συνέχεια, δημιουργεί ένα νέο αντικείμενο LibrarySection που αντιπροσωπεύει το τμήμα που πρόκειται να διαγραφεί και καλεί τη μέθοδο sectionDao().deleteSection() για να το αφαιρέσει από τη βάση δεδομένων.**

**Εάν η λειτουργία διαγραφής είναι επιτυχής, εμφανίζεται ένα μήνυμα επιτυχίας χρησιμοποιώντας μια ειδοποίηση Toast. Εάν προκύψει οποιοδήποτε σφάλμα κατά τη διαδικασία διαγραφής, εμφανίζεται ένα μήνυμα σφάλματος για να ειδοποιήσει τον χρήστη.**

## **ΑΡΧΕΙΑ ΓΙΑ ΔΙΑΧΕΙΡΗΣΗ ΔΑΝΕΙΩΝ**

**fragment\_insert\_loan.xml:**

**Το αρχείο fragment\_insert\_loan.xml, ορίζει τη διεπαφή χρήστη μιας φόρμας που χρησιμοποιείται για την εισαγωγή στοιχείων δανείου.**

**Χρησιμοποιεί ένα RelativeLayout ως βασική διάταξη, παρέχοντας ευελιξία στη διάταξη των θυγατρικών προβολών του.**

**Μέσα στο RelativeLayout περικλείεται ένα CardView που χρησιμεύει ως περιέκτης για τα στοιχεία της φόρμας. Αυτό το CardView παρουσιάζει στρογγυλεμένες γωνίες και μια διακριτική ανύψωση, ενισχύοντας την οπτική του ελκυστικότητα.**

**Ενσωματωμένο μέσα στο CardView είναι ένα κάθετο LinearLayout υπεύθυνο για την οργάνωση διαφόρων στοιχείων, συμπεριλαμβανομένων TextViews, Spinners και πεδίων εισαγωγής κειμένου. προτρέπουν τον χρήστη να επιλέξει ένα βιβλίο, ένα μέλος και να συμπληρώσει λεπτομέρειες του δανείου, όπως η ημερομηνία λήξης. Παρέχονται spinners για την επιλογή του βιβλίου και του μέλους, ενώ ένα πεδίο εισαγωγής κειμένου τυλιγμένο σε ένα TextInputLayout χρησιμοποιείται για την εισαγωγή της ημερομηνίας λήξης.**

**Επιπλέον, παρέχεται ένα κουμπί MaterialButton για την έναρξη της διαδικασίας εισαγωγής δανείου, βελτιώνοντας την εμπειρία του χρήστη προσφέροντας έναν απλό τρόπο για την υποβολή των στοιχείων του δανείου.**

**Εξήγηση ειδικού κώδικα:**

private void showDatePickerDialog() {  
 // Get current date  
 Calendar calendar = Calendar.*getInstance*();  
 int year = calendar.get(Calendar.*YEAR*);  
 int month = calendar.get(Calendar.*MONTH*);  
 int dayOfMonth = calendar.get(Calendar.*DAY\_OF\_MONTH*);  
  
 // Create DatePickerDialog and set minimum date  
 DatePickerDialog datePickerDialog = new DatePickerDialog(requireContext(), new DatePickerDialog.OnDateSetListener() {  
 @Override  
 public void onDateSet(DatePicker view, int year, int monthOfYear, int dayOfMonth) {  
 // Set the selected date to the EditText  
 duedatetext.setText(String.*format*(Locale.*getDefault*(), "%02d/%02d/%d", dayOfMonth, monthOfYear + 1, year));  
 }  
 }, year, month, dayOfMonth);  
  
 // Set minimum date  
 datePickerDialog.getDatePicker().setMinDate(System.*currentTimeMillis*() - 1000); // Disable past dates  
 datePickerDialog.show();  
}

**1. Λήψη τρέχουσας ημερομηνίας:**

* **Η μέθοδος ξεκινά με τη λήψη της τρέχουσας ημερομηνίας χρησιμοποιώντας τη μέθοδο `Calendar.getInstance()`. Αυτό δημιουργεί μια υπόσταση `Calendar` που αντιπροσωπεύει την τρέχουσα ημερομηνία και ώρα.**
* **Ανακτά τα στοιχεία του έτους, του μήνα και της ημέρας του μήνα από την περίπτωση `Calendar` χρησιμοποιώντας τη μέθοδο `get()`.**

**2. Δημιουργία DatePickerDialog:**

* **Στη συνέχεια, δημιουργείται ένα `DatePickerDialog`. Αυτός ο διάλογος επιτρέπει στο χρήστη να επιλέξει μια ημερομηνία από ένα γραφικό περιβάλλον τύπου ημερολογίου.**
* **Ο κατασκευαστής του `DatePickerDialog` λαμβάνει διάφορες παραμέτρους:**
* **Το πλαίσιο (`requireContext()` σε αυτή την περίπτωση) χρησιμοποιείται για την ανάκτηση του κατάλληλου θέματος και των πόρων για τον διάλογο.**
* **Παρέχεται ένας `OnDateSetListener`, ο οποίος καλείται όταν ο χρήστης ορίζει την ημερομηνία στο διάλογο. Μέσα στη μέθοδο `onDateSet()` αυτού του ακροατή, η επιλεγμένη ημερομηνία μορφοποιείται και ορίζεται στο `EditText` (`duedatetext`).**
* **Το έτος, ο μήνας και η ημέρα του μήνα που ανακτήθηκαν νωρίτερα χρησιμοποιούνται για την αρχικοποίηση της αρχικής ημερομηνίας που εμφανίζεται στο παράθυρο διαλόγου.**

**3. Ορισμός ελάχιστης ημερομηνίας:**

* **Πριν από την εμφάνιση του διαλόγου, η ελάχιστη επιλέξιμη ημερομηνία ορίζεται με τη μέθοδο `setMinDate()` του `DatePicker` μέσα στο διάλογο.**
* **Η μέθοδος `System.currentTimeMillis() - 1000` υπολογίζει την τρέχουσα ώρα μείον ένα δευτερόλεπτο, διασφαλίζοντας ότι οι ημερομηνίες πριν από την τρέχουσα ημερομηνία είναι απενεργοποιημένες.**
* **Αυτό εμποδίζει τον χρήστη να επιλέξει παρελθοντικές ημερομηνίες στο πλαίσιο του διαλόγου.**

**4. Εμφάνιση διαλόγου:**

* **Τέλος, καλείται η μέθοδος `show()` στο `DatePickerDialog` για να εμφανιστεί στον χρήστη.**

**Συνοπτικά, αυτή η μέθοδος δημιουργεί ένα παράθυρο διαλόγου επιλογής ημερομηνίας που επιτρέπει στο χρήστη να επιλέξει μια ημερομηνία, διασφαλίζοντας ότι μόνο μελλοντικές ημερομηνίες είναι επιλέξιμες. Μόλις επιλεγεί μια ημερομηνία, μορφοποιεί και θέτει την επιλεγμένη ημερομηνία στο καθορισμένο `EditText`.**

private boolean isValidDateFormat(String date) {  
 try {  
 // Parse the date using the specified format (dd/MM/yyyy)  
 SimpleDateFormat sdf = new SimpleDateFormat("dd/MM/yyyy", Locale.*getDefault*());  
 sdf.setLenient(false);  
 sdf.parse(date);  
 return true; // If parsing is successful, return true  
 } catch (ParseException e) {  
 return false; // If parsing fails, return false  
 }  
 }

**1. Μπλοκ Try-Catch:**

* **Η μέθοδος ενθυλακώνει τη λειτουργικότητά της μέσα σε ένα μπλοκ try-catch. Αυτή η δομή επιτρέπει το χειρισμό πιθανών εξαιρέσεων που μπορεί να προκύψουν κατά τη διαδικασία ανάλυσης.**

**2. Ανάλυση ημερομηνίας:**

* **Εντός του μπλοκ try, η μέθοδος επιχειρεί να αναλύσει την παρεχόμενη συμβολοσειρά `date` χρησιμοποιώντας ένα αντικείμενο `SimpleDateFormat` (`sdf`).**
* **Αυτό το αντικείμενο διαμορφώνεται με το πρότυπο `"dd/MM/yyyy"`, το οποίο υποδηλώνει την αναμενόμενη μορφή της συμβολοσειράς ημερομηνίας: ημέρα, μήνας και έτος χωρισμένα με κάθετους.**
* **Με τη ρύθμιση `setLenient(false)` στο αντικείμενο `SimpleDateFormat`, διασφαλίζεται ότι η διαδικασία ανάλυσης τηρεί αυστηρά την καθορισμένη μορφή. Οποιαδήποτε απόκλιση από αυτή τη μορφή θα οδηγήσει σε `ParseException`.**

**3. Τιμές επιστροφής:**

* **Εάν η διαδικασία ανάλυσης πετύχει χωρίς να αντιμετωπίσει καμία εξαίρεση, η μέθοδος επιστρέφει την τιμή `true`. Αυτό υποδηλώνει ότι η παρεχόμενη συμβολοσειρά ημερομηνίας ταιριάζει με την καθορισμένη μορφή.**
* **Εάν κατά την προσπάθεια ανάλυσης εμφανιστεί `ParseException` (δηλ. η συμβολοσειρά ημερομηνίας εισόδου δεν ταιριάζει με την αναμενόμενη μορφή), το μπλοκ catch χειρίζεται αυτή την εξαίρεση. Σε τέτοιες περιπτώσεις, η μέθοδος επιστρέφει `false`, υποδεικνύοντας ότι η παρεχόμενη συμβολοσειρά ημερομηνίας δεν έχει τη σωστή μορφή.**

**Συνοπτικά, αυτή η μέθοδος λειτουργεί ως επικυρωτής για συμβολοσειρές ημερομηνιών, διασφαλίζοντας ότι τηρούν μια συγκεκριμένη μορφή (`dd/MM/yyyy`). Ο πρωταρχικός της στόχος είναι να παρέχει έναν αξιόπιστο έλεγχο για τη συνέπεια της μορφής ημερομηνίας, επιτρέποντας στις εφαρμογές να επιβάλλουν τυποποιημένη εισαγωγή ημερομηνίας σε διάφορες αλληλεπιδράσεις με τον χρήστη.**

loanCollectionRef  
 .document()  
 .set(loanData, SetOptions.*merge*())  
 .addOnSuccessListener(new OnSuccessListener<Void>() {  
 @Override  
 public void onSuccess(Void aVoid) {  
 Toast.*makeText*(getActivity(), "Loan added successfully", Toast.*LENGTH\_SHORT*).show();  
 populateSpinner();  
 duedatetext.setText("");  
 duedatetext.clearFocus();  
 }  
 })  
 .addOnFailureListener(new OnFailureListener() {  
 @Override  
 public void onFailure(@NonNull Exception e) {  
 Toast.*makeText*(getActivity(), "Failed to add loan: " + e.getMessage(), Toast.*LENGTH\_SHORT*).show();  
 }  
 });

* **Η `SetOptions.merge()` χρησιμοποιείται κατά την εγγραφή δεδομένων σε ένα έγγραφο Firestore.**
* **Ενημερώνει ένα υπάρχον έγγραφο συγχωνεύοντας τα νέα δεδομένα με το υπάρχον έγγραφο.**
* **Τα πεδία που καθορίζονται στα νέα δεδομένα ενημερώνονται ή προστίθενται στο έγγραφο, ενώ τα υπάρχοντα πεδία που δεν αναφέρονται στα νέα δεδομένα παραμένουν αμετάβλητα.**
* **Εάν το έγγραφο δεν υπάρχει, δημιουργεί ένα νέο έγγραφο με τα καθορισμένα δεδομένα.**

**Στην ουσία, επιτρέπει την ενημέρωση συγκεκριμένων πεδίων ενός εγγράφου χωρίς να αντικαθιστά ολόκληρο το έγγραφο.**

**fragment\_update\_loan.xml:**

**Το αρχείο fragment\_update\_loan.xml, περιγράφει τη διεπαφή χρήστη μιας φόρμας προσαρμοσμένης για την ενημέρωση των στοιχείων του δανείου. Υιοθετεί ένα RelativeLayout ως βασική διάταξη, παρέχοντας ευελιξία στη διάταξη των θυγατρικών προβολών του.**

**Μέσα στο RelativeLayout περιέχεται ένα CardView, προσεκτικά σχεδιασμένο με στρογγυλεμένες γωνίες και ένα διακριτικό εφέ ανύψωσης για να προσδώσει μια γυαλισμένη, μοντέρνα εμφάνιση. Εξυπηρετώντας ως δοχείο για τα στοιχεία της φόρμας, αυτό το CardView παρέχει μια συνεκτική οπτική παρουσίαση.**

**Ενσωματωμένο μέσα στο CardView βρίσκεται ένα κάθετο LinearLayout, σχολαστικά σχεδιασμένο για να οργανώνει διάφορα στοιχεία, όπως TextViews, περιστρεφόμενα και πεδία εισαγωγής κειμένου. Τα TextViews προτρέπουν τον χρήστη να επιλέξει ένα δάνειο και να συμπληρώσει τις ενημερωμένες λεπτομέρειες, στην περίπτωση μας η ημερομηνία λήξης. Παρέχονται spinners για την επιλογή δανείου, ενώ ένα πεδίο εισαγωγής κειμένου τυλιγμένο σε ένα TextInputLayout διευκολύνει την εισαγωγή της ενημερωμένης ημερομηνίας λήξης.**

**Επιπλέον, στο κάτω μέρος της διάταξης βρίσκεται ένα κουμπί MaterialButton, το οποίο χρησιμεύει ως διαισθητικό μέσο για την έναρξη της διαδικασίας ενημέρωσης του δανείου. Η τοποθέτησή του μέσα σε ένα LinearLayout εξασφαλίζει τη σωστή ευθυγράμμιση και απόσταση, συμβάλλοντας σε μια απρόσκοπτη εμπειρία χρήστη.**

**fragment\_delete\_loan.xml:**

**Το αρχείο fragment\_delete\_loan.xml, περιγράφει τη διεπαφή χρήστη για τη διαγραφή εγγραφών δανείων . Αξιοποιεί ένα RelativeLayout ως βασική διάταξη, παρέχοντας προσαρμοστικότητα στη διάταξη των θυγατρικών προβολών του.**

**Μέσα στο RelativeLayout περικλείεται ένα CardView, προσεκτικά σχεδιασμένο με στρογγυλεμένες γωνίες και ένα διακριτικό εφέ ανύψωσης, που ενισχύει την οπτική του ελκυστικότητα. Αυτό το CardView χρησιμεύει ως δοχείο για τα στοιχεία της φόρμας, προσφέροντας μια συνεκτική δομή διάταξης.**

**Ενσωματωμένο μέσα στο CardView είναι ένα κάθετο LinearLayout, σχολαστικά σχεδιασμένο για να οργανώνει διάφορα στοιχεία, συμπεριλαμβανομένου ενώς TextView , ενός spinner και ενός κουμπιού. Η προβολή κειμένου ζητά από το χρήστη να επιλέξει την εγγραφή δανείου που πρόκειται να διαγραφεί, ενώ το spinner παρέχει επιλογές για την επιλογή δανείου.**

**Επιπλέον, ένα MaterialButton βρίσκεται στο κάτω μέρος της διάταξης, διευκολύνοντας τη διαδικασία διαγραφής. Η τοποθέτησή του μέσα σε ένα LinearLayout εξασφαλίζει τη σωστή ευθυγράμμιση και απόσταση, συμβάλλοντας σε μια απρόσκοπτη εμπειρία χρήστη.**

**InsertLoanFragment.java:**

**Η κλάση InsertLoanFragment διευκολύνει την προσθήκη νέων εγγραφών δανεισμού στη βάση δεδομένων της βιβλιοθήκης. Περιλαμβάνει spinners για την επιλογή ενός βιβλίου και ενός μέλους, μαζί με τον προσδιορισμό της ημερομηνίας λήξης του δανείου χρησιμοποιώντας ένα πεδίο EditText. Επιπλέον, διαθέτει ένα κουμπί για την υποβολή των στοιχείων του δανείου.**

**Με τον πληθωρισμό της διάταξής του, το fragment αρχικοποιεί τα spinners για την επιλογή ενός βιβλίου και ενός μέλους. Ρυθμίζει επίσης έναν ακροατή κλικ για το κουμπί εισαγωγής.**

**Η μέθοδος populateSpinner() ανακτά όλα τα βιβλία από τη βάση δεδομένων και φιλτράρει αυτά που έχουν ήδη δανειστεί. Στη συνέχεια γεμίζει το spinner βιβλίου με τους τίτλους των διαθέσιμων βιβλίων και το spinner μέλους με τα ονόματα όλων των μελών.**

**Η μέθοδος showDatePickerDialog() εμφανίζει ένα DatePickerDialog όταν το EditText ημερομηνία λήξης αποκτά εστίαση, επιτρέποντας στο χρήστη να επιλέξει μια ημερομηνία λήξης για το δάνειο.**

**Όταν γίνεται κλικ στο κουμπί εισαγωγής, ενεργοποιείται η μέθοδος insertLoan(). Ανακτά το επιλεγμένο αναγνωριστικό βιβλίου, το αναγνωριστικό μέλους και την ημερομηνία λήξης από τα spinners και το πεδίο EditText. Στη συνέχεια, δημιουργεί ένα νέο αντικείμενο δανείου με αυτές τις λεπτομέρειες και το προσθέτει στη βάση δεδομένων Firestore.**

**Εάν η λειτουργία εισαγωγής είναι επιτυχής, εμφανίζεται ένα μήνυμα επιτυχίας με τη χρήση μιας ειδοποίησης Toast και το βιβλίο με την κράτηση αφαιρείται από το spinner για να αντικατοπτρίζει την ενημέρωση. Εάν προκύψει κάποιο σφάλμα κατά τη διαδικασία εισαγωγής, εμφανίζεται ένα μήνυμα σφάλματος για να ειδοποιηθεί ο χρήστης.**

**UpdateLoanFragment.java:**

**Η κλάση UpdateLoanFragment είναι υπεύθυνη για την ενημέρωση των υφιστάμενων καταχωρίσεων δανείων στη βάση δεδομένων της βιβλιοθήκης. Παρέχει ένα spinner για την επιλογή του δανείου προς ενημέρωση και ένα πεδίο EditText για τον καθορισμό της νέας ημερομηνίας λήξης του δανείου. Επιπλέον, διαθέτει ένα κουμπί για την υποβολή των ενημερώσεων.**

**Κατά τον πληθωρισμό της διάταξής του, το fragment αρχικοποιεί το spinner δανείου και το EditText της ημερομηνίας λήξης. Ρυθμίζει επίσης έναν ακροατή κλικ για το κουμπί ενημέρωσης.**

**Η μέθοδος populateSpinners() ανακτά όλες τις υπάρχουσες καταχωρήσεις δανείων από τη βάση δεδομένων Firestore και γεμίζει το spinner δανείων με τους αντίστοιχους τίτλους βιβλίων και τα ονόματα των μελών. Κάθε καταχώρηση δανείου στο spinner εμφανίζεται ως "Book Title for Member Name".**

**Όταν η ημερομηνία λήξης EditText αποκτήσει εστίαση, η μέθοδος showDatePickerDialog() εμφανίζει ένα DatePickerDialog για να επιτρέψει στο χρήστη να επιλέξει μια νέα ημερομηνία λήξης για το δάνειο.**

**Όταν γίνεται κλικ στο κουμπί ενημέρωσης, ενεργοποιείται η μέθοδος updateLoan(). Ανακτά το αναγνωριστικό της επιλεγμένης καταχώρησης δανείου από το spinner και τη νέα ημερομηνία λήξης από το πεδίο EditText. Στη συνέχεια, ενημερώνει την ημερομηνία λήξης της επιλεγμένης καταχώρησης δανείου στη βάση δεδομένων Firestore.**

**Εάν η λειτουργία ενημέρωσης είναι επιτυχής, εμφανίζεται ένα μήνυμα επιτυχίας με τη χρήση μιας ειδοποίησης Toast και ανανεώνεται το δάνειο spinner για να αντικατοπτρίζει τις αλλαγές. Εάν προκύψει κάποιο σφάλμα κατά τη διαδικασία ενημέρωσης, εμφανίζεται ένα μήνυμα σφάλματος για να ειδοποιήσει τον χρήστη.**

**DeleteLoanFragment.java:**

**Το DeleteLoanFragment είναι υπεύθυνο για τη διαγραφή των υφιστάμενων καταχωρήσεων δανείων από τη βάση δεδομένων της βιβλιοθήκης. Αποτελείται από ένα spinner για την επιλογή του δανείου προς διαγραφή και ένα κουμπί για την εκτέλεση της λειτουργίας διαγραφής.**

**Κατά τον πληθωρισμό της διάταξής του, το fragment αρχικοποιεί το spinner του δανείου και το κουμπί διαγραφής. Ρυθμίζει επίσης έναν ακροατή κλικ για το κουμπί διαγραφής.**

**Η μέθοδος populateSpinner() ανακτά όλες τις υπάρχουσες καταχωρήσεις δανείων από τη βάση δεδομένων Firestore και γεμίζει το δάνειο spinner με τους αντίστοιχους τίτλους βιβλίων και τα ονόματα των μελών. Κάθε καταχώρηση δανείου στο spinner εμφανίζεται ως "Book Title for Member Name".**

**Όταν γίνεται κλικ στο κουμπί διαγραφής, ενεργοποιείται η μέθοδος deleteLoan(). Ανακτά το αναγνωριστικό της επιλεγμένης καταχώρησης δανείου από το spinner και προχωρά στη διαγραφή του αντίστοιχου εγγράφου από τη βάση δεδομένων Firestore.**

**Εάν η λειτουργία διαγραφής είναι επιτυχής, εμφανίζεται ένα μήνυμα επιτυχίας με τη χρήση μιας ειδοποίησης toast και το spinner του δανείου ανανεώνεται για να αντικατοπτρίζει τις αλλαγές. Εάν προκύψει κάποιο σφάλμα κατά τη διαδικασία διαγραφής, εμφανίζεται ένα μήνυμα σφάλματος για να ειδοποιήσει τον χρήστη.**

## **ΑΡΧΕΙΑ ΓΙΑ ΒΑΣΗ ΔΕΔΟΜΕΝΩΝ**

**Book.java:**

Η κλάση «Book.java» είναι ζωτικής σημασίας για τη διαχείριση οντοτήτων βιβλίων εντός της βάσης δεδομένων της εφαρμογής βιβλιοθήκης. Είναι διακοσμημένο με σχολιασμούς όπως "@Entity", που δηλώνει την κατάστασή του ως οντότητα βάσης δεδομένων και "@PrimaryKey", που υποδεικνύει το πεδίο πρωτεύοντος κλειδιού. Μέσα στην τάξη, διάφορα πεδία καταγράφουν βασικές λεπτομέρειες του βιβλίου:

* 'id': χρησιμεύει ως πρωτεύον κλειδί, που δημιουργείται αυτόματα για κάθε οντότητα βιβλίου.
* 'title': αποθηκεύει τον τίτλο του βιβλίου.
* ‘author’: κατέχει το όνομα του συγγραφέα του βιβλίου.
* 'pages': παρακολουθεί τον αριθμό των σελίδων του βιβλίου.
* 'sectionId': λειτουργεί ως ξένο κλειδί, συνδέοντας κάθε βιβλίο με το αντίστοιχο τμήμα βιβλιοθήκης.

Παρέχονται μέθοδοι accessor και mutator (getters και setters) για κάθε πεδίο, επιτρέποντας την ανάκτηση και τροποποίηση των χαρακτηριστικών του βιβλίου. Αυτή η δομημένη προσέγγιση διασφαλίζει τον αποτελεσματικό χειρισμό των δεδομένων βιβλίων, διευκολύνοντας λειτουργίες όπως η δημιουργία, η ανάγνωση, η ενημέρωση και η διαγραφή αρχείων βιβλίων. Ο σχολιασμός «@ForeignKey» διασφαλίζει την ακεραιότητα της σχέσης μεταξύ βιβλίων και τμημάτων βιβλιοθήκης.

**BookDao.java:**

**Η διεπαφή «BookDao.java» ορίζει το αντικείμενο πρόσβασης δεδομένων (DAO) που είναι υπεύθυνο για την αλληλεπίδραση με την οντότητα «Βιβλίο» στη βάση δεδομένων αίθουσας της εφαρμογής βιβλιοθήκης. Σχολιασμένο με '@Dao', περιγράφει διάφορες μεθόδους για την εκτέλεση λειτουργιών CRUD (Δημιουργία, Ανάγνωση, Ενημέρωση, Διαγραφή) σε οντότητες βιβλίων.**

**Η διεπαφή δηλώνει μεθόδους σχολιασμένες με «@Insert», «@Update» και «@Delete» για την εισαγωγή, ενημέρωση και διαγραφή εγγραφών βιβλίων, αντίστοιχα. Κάθε μέθοδος λαμβάνει ένα αντικείμενο «Βιβλίο» ως παράμετρο, επιτρέποντας το χειρισμό μεμονωμένων εγγραφών βιβλίων στη βάση δεδομένων.**

**Επιπλέον, η διασύνδεση περιλαμβάνει διάφορες μεθόδους ερωτήματος σχολιασμένες με '@Query'. Η μέθοδος 'getAllBooks()' ανακτά όλα τα βιβλία που είναι αποθηκευμένα στη βάση δεδομένων, επιστρέφοντας μια λίστα αντικειμένων 'Book'. Ομοίως, το 'getBooksAbove300Pages()' ανακτά βιβλία με περισσότερες από 300 σελίδες, παρέχοντας μια φιλτραρισμένη λίστα με βάση τα καθορισμένα κριτήρια.**

**Άλλες μέθοδοι ερωτήματος περιλαμβάνουν το 'getBookTitle(int i)', το οποίο ανακτά τον τίτλο ενός βιβλίου με το αναγνωριστικό του και το 'getBookById(int i)', το οποίο ανακτά ολόκληρη την οντότητα βιβλίου με βάση το αναγνωριστικό του.**

**LibraryDatabase.java:**

**Η κλάση LibraryDatabase.java ορίζει τη δομή της βάσης δεδομένων για την εφαρμογή βιβλιοθήκης. Καθορίζει οντότητες όπως Book, Member και LibrarySection και παρέχει πρόσβαση σε αντίστοιχα αντικείμενα πρόσβασης σε δεδομένα (DAOs) για λειτουργίες CRUD. Αυτή η κλάση χρησιμεύει ως το κεντρικό σημείο πρόσβασης στη βάση δεδομένων SQLite, περιγράφοντας το σχήμα και την έκδοση, ενώ αφαιρεί τις πολυπλοκότητες της SQL μέσω της αρχιτεκτονικής του Room.**

**LibrarySection.java:**

**Η κλάση "LibrarySection.java" είναι απαραίτητη για τη διαχείριση των τμημάτων της βιβλιοθήκης εντός της βάσης δεδομένων της εφαρμογής. Διακοσμημένο με σχολιασμούς όπως "@Entity" και "@PrimaryKey", υποδεικνύει ότι κάθε εμφάνιση αυτής της κλάσης θα αποθηκευτεί ως εγγραφή στον πίνακα "library\_sections" της βάσης δεδομένων.**

* **‘id’: Αντιπροσωπεύει το πρωτεύον κλειδί της οντότητας ενότητας βιβλιοθήκης. Σχολιάζεται με "@PrimaryKey" και έχει ρυθμιστεί ώστε να δημιουργεί αυτόματα τιμές.**
* **‘name’: Αποθηκεύει το όνομα του τμήματος βιβλιοθήκης.**
* **‘location’: Αποθηκεύει τη θέση του τμήματος βιβλιοθήκης.**
* **‘bookCount’: Αντιπροσωπεύει τον αριθμό των βιβλίων στην ενότητα βιβλιοθήκης.**

**Οι μέθοδοι Getter και setter παρέχονται για κάθε πεδίο για να διασφαλιστεί η πρόσβαση και η τροποποίηση των χαρακτηριστικών του τμήματος βιβλιοθήκης. Με αυτή την καθορισμένη δομή, η εφαρμογή μπορεί να διαχειριστεί και να αποθηκεύσει αποτελεσματικά πληροφορίες σχετικά με τα διάφορα τμήματα της βιβλιοθήκης στη βάση δεδομένων.**

**LibrarySectionDao.java:**

**Η διασύνδεση LibrarySectionDao.java ορίζει το αντικείμενο πρόσβασης δεδομένων (DAO) που είναι υπεύθυνο για την αλληλεπίδραση με την οντότητα LibrarySection στη βάση δεδομένων Room της εφαρμογής βιβλιοθήκης. Σχολιασμένο με @Dao, περιγράφει διάφορες μεθόδους για την εκτέλεση λειτουργιών CRUD (Δημιουργία, Ανάγνωση, Ενημέρωση, Διαγραφή) σε οντότητες ενότητας βιβλιοθήκης.**

**Η διασύνδεση δηλώνει μεθόδους σχολιασμένες με @Insert, @Update και @Delete για την εισαγωγή, ενημέρωση και διαγραφή εγγραφών ενότητας βιβλιοθήκης, αντίστοιχα. Κάθε μέθοδος λαμβάνει ένα αντικείμενο LibrarySection ως παράμετρο, επιτρέποντας το χειρισμό μεμονωμένων καταχωρήσεων ενότητας βιβλιοθήκης στη βάση δεδομένων.**

**Επιπλέον, η διασύνδεση περιλαμβάνει διάφορες μεθόδους ερωτήματος σχολιασμένες με @Query. Η μέθοδος getAllSections() ανακτά όλες τις ενότητες βιβλιοθήκης που είναι αποθηκευμένες στη βάση δεδομένων, επιστρέφοντας μια λίστα αντικειμένων LibrarySection. Ομοίως, η μέθοδος getAllSectionsWithBookCount() λαμβάνει όλες τις ενότητες βιβλιοθήκης μαζί με τον αριθμό των βιβλίων σε κάθε ενότητα.**

**Member.java:**

**Η κλάση «Member.java» είναι ζωτικής σημασίας για τη διαχείριση οντοτήτων μελών εντός της βάσης δεδομένων της εφαρμογής βιβλιοθήκης. Σχολιασμένο με «@Entity», υποδηλώνει τον ρόλο του ως οντότητας βάσης δεδομένων.**

**Μέσα στην τάξη, διάφορα πεδία καταγράφουν βασικά στοιχεία των μελών:**

* **'id': χρησιμεύει ως πρωτεύον κλειδί, που δημιουργείται αυτόματα για κάθε οντότητα μέλος.**
* **'name': αποθηκεύει το όνομα του μέλους.**
* **'email': διατηρεί τη διεύθυνση ηλεκτρονικού ταχυδρομείου του μέλους.**

**Για κάθε πεδίο παρέχονται μέθοδοι accessor και mutator (getters και setters), διευκολύνοντας την ανάκτηση και τροποποίηση των χαρακτηριστικών των μελών. Αυτή η δομημένη προσέγγιση διασφαλίζει τον αποτελεσματικό χειρισμό των δεδομένων των μελών, επιτρέποντας λειτουργίες όπως η δημιουργία, η ανάγνωση, η ενημέρωση και η διαγραφή εγγραφών μελών.**

**MemberDao.java:**

Η διασύνδεση MemberDao ορίζει το αντικείμενο πρόσβασης δεδομένων (DAO) που είναι υπεύθυνο για την αλληλεπίδραση με την οντότητα μέλος στη βάση δεδομένων αίθουσας της εφαρμογής βιβλιοθήκης. Σχολιασμένο με @Dao, περιγράφει διάφορες μεθόδους για την εκτέλεση λειτουργιών CRUD (Δημιουργία, Ανάγνωση, Ενημέρωση, Διαγραφή) σε οντότητες μέλη.

Παρόμοια με τη διασύνδεση BookDao, το MemberDao περιλαμβάνει μεθόδους σχολιασμένες με @Insert, @Update και @Delete για εισαγωγή, ενημέρωση και διαγραφή εγγραφών μελών, αντίστοιχα. Κάθε μέθοδος λαμβάνει ένα αντικείμενο μέλους ως παράμετρο, επιτρέποντας το χειρισμό μεμονωμένων καταχωρήσεων μελών στη βάση δεδομένων.

Επιπλέον, η διασύνδεση περιλαμβάνει μεθόδους ερωτήματος σχολιασμένες με @Query. Η μέθοδος getAllMembers() ανακτά όλα τα μέλη που είναι αποθηκευμένα στη βάση δεδομένων, επιστρέφοντας μια λίστα αντικειμένων μελών.

Μια άλλη μέθοδος ερωτήματος, getMemberName(Integer memberId), λαμβάνει το όνομα ενός μέλους που έχει λάβει το αναγνωριστικό του. Αυτή η μέθοδος χρησιμοποιεί ένα παραμετροποιημένο ερώτημα για την ανάκτηση του ονόματος που αντιστοιχεί στο παρεχόμενο αναγνωριστικό μέλους.

## **ΑΡΧΕΙΟ ΓΙΑ NOTIFICATION**

**DueDateCheckService.java:**

**Η υπηρεσία DueDateCheckService λειτουργεί ως βασική υπηρεσία στο παρασκήνιο της εφαρμογής της βιβλιοθήκης, επιβλέποντας σχολαστικά τις ημερομηνίες λήξης των δανείων και διασφαλίζοντας ότι οι admins ενημερώνονται αμέσως για τις επικείμενες προθεσμίες. Κατά την αρχικοποίηση, η υπηρεσία ενορχηστρώνει μια ScheduledExecutorService, η οποία έχει ρυθμιστεί σχολαστικά ώστε να πραγματοποιεί σαρώσεις δανείων σε τακτά χρονικά διαστήματα, διασφαλίζοντας έτσι ότι καμία προθεσμία δεν ξεφεύγει από την προσοχή της.**

**Κατά τη διάρκεια κάθε κύκλου σάρωσης, η υπηρεσία πραγματοποιεί επιμελώς ερωτήματα στη βάση δεδομένων Firestore, εξάγοντας σχολαστικά ζωτικής σημασίας στοιχεία δανείων, όπως ημερομηνίες λήξης, αναγνωριστικά μελών και αναγνωριστικά βιβλίων. Οπλισμένη με αυτές τις πληροφορίες, αναλαμβάνει το κρίσιμο έργο του υπολογισμού της χρονικής διαφοράς μεταξύ της ημερομηνίας λήξης κάθε δανείου και της τρέχουσας ώρας του συστήματος. Αυτός ο σχολαστικός υπολογισμός επιτρέπει στην υπηρεσία να εντοπίζει δάνεια με προθεσμίες που πλησιάζουν εντός των επόμενων δύο ημερών, σηματοδοτώντας την επικείμενη ανάγκη προσοχής του χρήστη.**

**Με τον εντοπισμό των δανείων που πλησιάζουν στην ημερομηνία λήξης τους, η υπηρεσία αναλαμβάνει αμέσως δράση, συντάσσοντας και αποστέλλοντας προσεγμένες ειδοποιήσεις στους αντίστοιχους χρήστες. Αυτές οι ειδοποιήσεις, κομψά σχεδιασμένες με τη χρήση του NotificationCompat.Builder, εμπλουτίζονται με σχετικές λεπτομέρειες, όπως ο τίτλος του βιβλίου και το όνομα του δανειολήπτη, προωθώντας τη σαφήνεια και διευκολύνοντας την ταχεία ανάληψη δράσης. Αξιοποιώντας τη δύναμη του NotificationManagerCompat, η υπηρεσία εξασφαλίζει την απρόσκοπτη παράδοση των ειδοποιήσεων στις συσκευές των χρηστών, ενισχύοντας έτσι τα κανάλια επικοινωνίας και βελτιώνοντας την εμπειρία των χρηστών.**

**Εξήγηση ειδικού κώδικα:**

package com.example.libraryapp.notifications;  
  
import android.Manifest;  
import android.app.NotificationChannel;  
import android.app.NotificationManager;  
import android.app.Service;  
import android.content.Context;  
import android.content.Intent;  
import android.content.pm.PackageManager;  
import android.os.Build;  
import android.os.IBinder;  
  
import androidx.annotation.Nullable;  
import androidx.core.app.ActivityCompat;  
import androidx.core.app.NotificationCompat;  
import androidx.core.app.NotificationManagerCompat;  
  
import com.example.libraryapp.Drawer;  
import com.example.libraryapp.R;  
import com.google.firebase.firestore.FirebaseFirestore;  
import com.google.firebase.firestore.QueryDocumentSnapshot;  
  
import java.text.ParseException;  
import java.text.SimpleDateFormat;  
import java.util.Date;  
import java.util.Locale;  
import java.util.concurrent.Executors;  
import java.util.concurrent.ScheduledExecutorService;  
import java.util.concurrent.TimeUnit;  
  
public class DueDateCheckService extends Service {  
 private ScheduledExecutorService executor;  
 private static final String *CHANNEL\_ID* = "LibraryAppNotifications";  
  
 @Override  
 public void onCreate() {  
 super.onCreate();  
 }  
  
 @Nullable  
 @Override  
 public IBinder onBind(Intent intent) {  
 return null;  
 }  
  
 @Override  
 public int onStartCommand(Intent intent, int flags, int startId) {  
 startLoanScanning();  
 return *START\_STICKY*;  
 }  
  
 private void startLoanScanning() {  
 executor = Executors.*newSingleThreadScheduledExecutor*();  
 executor.scheduleAtFixedRate(this::startDueDateCheck, 0, 2, TimeUnit.*HOURS*);  
 }  
  
 private void startDueDateCheck() {  
 FirebaseFirestore db = FirebaseFirestore.*getInstance*();  
 db.collection("Loan")  
 .get()  
 .addOnCompleteListener(task -> {  
 if (task.isSuccessful()) {  
 for (QueryDocumentSnapshot document : task.getResult()) {  
 // Assuming dueDate is stored as a string in Firestore  
 String dueDateString = document.getString("dueDate");  
 Integer memberId = document.getLong("memberId").intValue();  
 Integer bookId = document.getLong("bookId").intValue();  
  
 // Retrieve book title and member name from local database  
 String bookTitle = Drawer.*database*.bookDao().getBookTitle(bookId);  
 String memberName = Drawer.*database*.memberDao().getMemberName(memberId);  
  
 if (dueDateString != null) {  
 try {  
 SimpleDateFormat dateFormat = new SimpleDateFormat("dd/MM/yyyy", Locale.*getDefault*());  
 Date dueDate = dateFormat.parse(dueDateString);  
 long currentTimeMillis = System.*currentTimeMillis*();  
 long dueDateTimeMillis = dueDate.getTime();  
 long timeDiffMillis = dueDateTimeMillis - currentTimeMillis;  
 int daysRemaining = (int) TimeUnit.*MILLISECONDS*.toDays(timeDiffMillis);  
 if (daysRemaining <= 2) {  
 // Send notification with different contexts  
 sendNotification(getApplicationContext(), bookTitle, memberName);  
 // You can call sendNotification() multiple times with different contexts  
 }  
 } catch (ParseException e) {  
 e.printStackTrace();  
 }  
 }  
 }  
 }  
 });  
 }  
  
 private void sendNotification(Context context, String bookTitle, String memberName) {  
 createNotificationChannel(context);  
  
 NotificationCompat.Builder builder = new NotificationCompat.Builder(context, *CHANNEL\_ID*)  
 .setSmallIcon(R.mipmap.*book\_icon*)  
 .setContentTitle("Loan Due Soon")  
 .setContentText("The book with title " + bookTitle + " should be returned within the next 2 days by " + memberName)  
 .setPriority(NotificationCompat.*PRIORITY\_DEFAULT*);  
  
 NotificationManagerCompat notificationManager = NotificationManagerCompat.*from*(context);  
 if (ActivityCompat.*checkSelfPermission*(this, Manifest.permission.*POST\_NOTIFICATIONS*) == PackageManager.*PERMISSION\_GRANTED*) {  
 // Use unique notification IDs for each notification to ensure they are displayed separately  
 int notificationId = generateNotificationId();  
 notificationManager.notify(notificationId, builder.build());  
 }  
 }  
  
 private int generateNotificationId() {  
 // Generate unique notification IDs using a random number generator or other method  
 return (int) System.*currentTimeMillis*();  
 }  
  
  
 private void createNotificationChannel(Context context) {  
 if (Build.VERSION.*SDK\_INT* >= Build.VERSION\_CODES.*O*) {  
 CharSequence name = "LibraryApp Notifications";  
 String description = "Notification channel for LibraryApp";  
 int importance = NotificationManager.*IMPORTANCE\_DEFAULT*;  
 NotificationChannel channel = new NotificationChannel(*CHANNEL\_ID*, name, importance);  
 channel.setDescription(description);  
  
 NotificationManager notificationManager = context.getSystemService(NotificationManager.class);  
 notificationManager.createNotificationChannel(channel);  
 }  
 }  
  
 @Override  
 public void onDestroy() {  
 super.onDestroy();  
 if (executor != null && !executor.isShutdown()) {  
 executor.shutdown();  
 }  
 }  
}

**Αυτός ο κώδικας ορίζει μια υπηρεσία υποβάθρου με όνομα `DueDateCheckService`, υπεύθυνη για τον περιοδικό έλεγχο των ημερομηνιών λήξης των δανείων που είναι αποθηκευμένα σε μια βάση δεδομένων Firestore και την αποστολή ειδοποιήσεων για να υπενθυμίζει στους admins τις επικείμενες ημερομηνίες λήξης.**

* **“Αρχικοποίηση της υπηρεσίας”:**
  + **Η `DueDateCheckService` επεκτείνει την κλάση `Service` του Android, δημιουργώντας ένα κανάλι ειδοποιήσεων στη μέθοδο `onCreate()`, η οποία είναι απαραίτητη για την εμφάνιση ειδοποιήσεων σε συσκευές Android που εκτελούν επίπεδο API 26 (Android 8.0) ή υψηλότερο.**
* **“Προγραμματισμένη εκτέλεση”:**
  + **Η μέθοδος `startLoanScanning()` δημιουργεί μια `ScheduledExecutorService` για την περιοδική εκτέλεση της μεθόδου `startDueDateCheck()` κάθε 2 ώρες.**
  + **Η μέθοδος `startDueDateCheck()` πραγματοποιεί αναζήτηση στη βάση δεδομένων Firestore για έγγραφα δανείων, επαναλαμβάνοντας το καθένα για να επιθεωρήσει τις ημερομηνίες λήξης.**
* **“Έλεγχος προθεσμίας”:**
  + **Για κάθε έγγραφο δανείου, ανακτά την ημερομηνία λήξης, το αναγνωριστικό μέλους και το αναγνωριστικό βιβλίου.**
  + **Φέρνει τον αντίστοιχο τίτλο του βιβλίου και το όνομα του μέλους από την τοπική βάση δεδομένων.**
  + **Εάν η ημερομηνία λήξης είναι εντός 2 ημερών από την τρέχουσα ημερομηνία, ενεργοποιεί μια ειδοποίηση χρησιμοποιώντας την `sendNotification()`.**
* **“Χειρισμός ειδοποιήσεων”:**
  + **Η μέθοδος `sendNotification()` κατασκευάζει και αποστέλλει μια ειδοποίηση με τίτλο που υποδεικνύει μια επικείμενη ημερομηνία λήξης.**
  + **Δημιουργεί ένα μοναδικό αναγνωριστικό ειδοποίησης για κάθε ειδοποίηση, ώστε να διασφαλίζεται η διακριτή εμφάνισή τους.**
  + **Εάν η έκδοση SDK της συσκευής είναι 26 (Oreo) ή υψηλότερη, δημιουργεί ένα κανάλι ειδοποιήσεων που είναι απαραίτητο για την εμφάνιση των ειδοποιήσεων.**
* **“Χειρισμός αδειών”:**
  + **Πριν από την αποστολή ειδοποιήσεων, η υπηρεσία επαληθεύει τη διαθεσιμότητα του δικαιώματος `POST\_NOTIFICATIONS`, το οποίο μπορεί να είναι ένα προσαρμοσμένο δικαίωμα που ορίζεται αλλού στην εφαρμογή.**
* **“Διαχείριση του κύκλου ζωής της υπηρεσίας”:**
  + **Κατά την καταστροφή της υπηρεσίας, η μέθοδος `onDestroy()` διασφαλίζει τον ορθό τερματισμό της υπηρεσίας εκτέλεσης, αποτρέποντας πιθανές διαρροές μνήμης.**

**Στην ουσία, η υπηρεσία αυτή λειτουργεί στο παρασκήνιο, παρακολουθώντας περιοδικά τις ημερομηνίες λήξης των δανείων και εκδίδοντας ειδοποιήσεις για να ενημερώνει τους χρήστες, ενισχύοντας έτσι τη χρηστικότητα της εφαρμογής της βιβλιοθήκης.**

## **ΑΡΧΕΙΑ ΓΙΑ ΕΜΦΑΝΗΣΗ ΔΕΔΟΜΕΝΩΝ**

**fragment\_search.xml:**

**Το αρχείο fragment\_search.xml, ορίζει τη διάταξη της οθόνης αναζήτησης στην εφαρμογή βιβλιοθήκης. Χρησιμοποιεί μια κατακόρυφη LinearLayout ως βασική διάταξη, επιτρέποντας μια απλή διάταξη των θυγατρικών προβολών της.**

**Στην κορυφή της διάταξης υπάρχει ένα TextView που προτρέπει τους χρήστες να επιλέξουν μια επιλογή. Αυτό το TextView είναι οριζόντια ευθυγραμμισμένο και τοποθετημένο στο επάνω μέρος της οθόνης, με ένα μικρό περιθώριο στην αρχή και στο επάνω μέρος για απόσταση. Το μέγεθος και το χρώμα του κειμένου ρυθμίζονται για βέλτιστη ορατότητα και αναγνωσιμότητα.**

**Κάτω από το TextView υπάρχει ένα στοιχείο Spinner, που επιτρέπει στους χρήστες να επιλέξουν μια επιλογή από ένα αναπτυσσόμενο μενού. Το Spinner ρυθμίζεται με τις κατάλληλες διαστάσεις και το padding, ώστε να διασφαλίζεται η συνοχή με το συνολικό σχεδιασμό της εφαρμογής.**

**Δίπλα στο Spinner βρίσκεται ένα κουμπί με την ένδειξη " Search " (Αναζήτηση), το οποίο οι χρήστες μπορούν να πατήσουν για να ξεκινήσουν τη διαδικασία αναζήτησης. Το κουμπί είναι κεντραρισμένο οριζόντια και τοποθετημένο κάτω από το Spinner με ένα περιθώριο για απόσταση. Το μέγεθος και η στοίχιση του κειμένου του έχουν ρυθμιστεί για σαφήνεια και συνέπεια.**

**Μετά το Κουμπί ακολουθεί ένα στοιχείο ScrollView, το οποίο επιτρέπει στους χρήστες να μετακινηθούν στα αποτελέσματα της αναζήτησης εάν αυτά υπερβαίνουν το διαθέσιμο χώρο της οθόνης. Μέσα στο ScrollView υπάρχει ένα TableLayout με όνομα queryResultTable, το οποίο χρησιμεύει ως περιέκτης για την εμφάνιση των αποτελεσμάτων αναζήτησης σε μορφή πίνακα. Το TableLayout είναι αρχικά άδειο, με τις γραμμές του πίνακα να προστίθενται δυναμικά με βάση τα αποτελέσματα αναζήτησης.**

**fragment\_search.xml (land):**

**Η οριζόντια έκδοση του αρχείου fragment\_search.xml, είναι βελτιστοποιημένη για οριζόντιο προσανατολισμό στις συσκευές. Διατηρεί τη λειτουργικότητα της οθόνης αναζήτησης, ενώ προσαρμόζει τη διάταξη ώστε να αξιοποιεί καλύτερα τον ευρύτερο χώρο της οθόνης που είναι διαθέσιμος σε οριζόντια λειτουργία.**

**Η διάταξη αποτελείται από ένα οριζόντιο LinearLayout ως ριζική διάταξη, χωρίζοντας την οθόνη σε δύο κύρια τμήματα: η αριστερή πλευρά περιέχει το Spinner και το κουμπί, και η δεξιά πλευρά εμφανίζει τα αποτελέσματα της αναζήτησης σε ένα TableLayout μέσα σε ένα ScrollView.**

**Στην αριστερή πλευρά, ένα κάθετο LinearLayout χρησιμοποιείται για την οργάνωση του Spinner και του Button. Ένα TextView προτρέπει τους χρήστες να επιλέξουν μια επιλογή, ακολουθούμενο από το στοιχείο Spinner για την επιλογή των κριτηρίων αναζήτησης. Ένα κουμπί με την ένδειξη " Search " (Αναζήτηση) επιτρέπει στους χρήστες να ξεκινήσουν τη διαδικασία αναζήτησης. Αυτές οι προβολές στοιβάζονται κάθετα για τη μεγιστοποίηση της χρήσης του κατακόρυφου χώρου.**

**Στη δεξιά πλευρά, ένα ScrollView τυλίγει ένα TableLayout με όνομα queryResultTable, το οποίο γεμίζει δυναμικά τις γραμμές για την εμφάνιση των αποτελεσμάτων αναζήτησης. Το ScrollView διασφαλίζει ότι οι χρήστες μπορούν να μετακινηθούν οριζόντια στα αποτελέσματα αναζήτησης, εάν αυτά υπερβαίνουν το πλάτος της οθόνης.**

**SearchFragment.java:**

**Η κλάση SearchFragment.java παίζει καθοριστικό ρόλο στην εφαρμογή της βιβλιοθήκης, παρέχοντας στους χρήστες μια ευέλικτη διεπαφή αναζήτησης για την απρόσκοπτη εξερεύνηση των πόρων της. Η κύρια λειτουργία του είναι να προσφέρει μια διαισθητική πλατφόρμα για την ανακάλυψη βιβλίων, τη διαχείριση των μελών και την πρόσβαση σε βασικές πληροφορίες της βιβλιοθήκης.**

**Κατά την αρχικοποίηση, το τεμάχιο ρυθμίζει κρίσιμα στοιχεία διεπαφής χρήστη, όπως ένα περιστρεφόμενο για την επιλογή κριτηρίων αναζήτησης, έναν πίνακα για την εμφάνιση των αποτελεσμάτων αναζήτησης και ένα κουμπί αναζήτησης για την ενεργοποίηση ερωτημάτων με βάση τις επιλογές του χρήστη. Κάθε επιλογή στο spinner αντιστοιχεί σε ένα συγκεκριμένο ερώτημα αναζήτησης, επιτρέποντας στους χρήστες να ανακτούν δεδομένα από τη βάση δεδομένων της βιβλιοθήκης προσαρμοσμένα στις ανάγκες τους.**

**Το spinner και ο πίνακας λειτουργούν σε συνδυασμό με το κουμπί αναζήτησης για να διευκολύνουν την αποτελεσματική ανάκτηση και παρουσίαση δεδομένων. Όταν ο χρήστης επιλέγει μια επιλογή από το spinner και πατάει το κουμπί αναζήτησης, το τμήμα εκτελεί το αντίστοιχο ερώτημα της βάσης δεδομένων. Αυτά τα ερωτήματα περιλαμβάνουν διάφορα κριτήρια αναζήτησης, όπως η άντληση του αριθμού των βιβλίων ανά τμήμα βιβλιοθήκης, των στοιχείων των μελών, των δανείων σε καθυστέρηση, των διαθέσιμων βιβλίων ή των ημερών που απομένουν ανά δανεισμένο βιβλίο.**

**Μετά την εκτέλεση του ερωτήματος βάσης δεδομένων, το τμήμα μορφοποιεί τα δεδομένα που ανακτώνται και συμπληρώνει τον πίνακα με τις σχετικές πληροφορίες. Κάθε γραμμή στον πίνακα αντιπροσωπεύει ένα ξεχωριστό αποτέλεσμα αναζήτησης, διασφαλίζοντας ότι οι χρήστες μπορούν εύκολα να πλοηγηθούν και να έχουν πρόσβαση στις λεπτομέρειες που χρειάζονται.**

**Η διαχείριση του κύκλου ζωής αποτελεί κρίσιμη πτυχή της λειτουργικότητας του SearchFragment. Χειρίζεται τα συμβάντα κύκλου ζωής onResume, onPause και onDestroyView για να διατηρεί συνεπή συμπεριφορά, να διασφαλίζει ομαλές μεταβάσεις στο UI και να βελτιστοποιεί τη διαχείριση των πόρων. Κατά τη διάρκεια των onResume και onPause, το fragment διαχειρίζεται την ορατότητα ενός κουμπιού αλληλογραφίας, προσαρμόζοντας την παρουσία του με βάση την αλληλεπίδραση του χρήστη με το fragment.**

**Επιπλέον, το SearchFragment ενσωματώνει ισχυρούς μηχανισμούς χειρισμού σφαλμάτων για την αξιοπρεπή διαχείριση των αποτυχιών ανάκτησης της βάσης δεδομένων. Σε περίπτωση σφάλματος, εμφανίζονται ενημερωτικά μηνύματα Toast για την καθοδήγηση των χρηστών και τη διατήρηση μιας θετικής εμπειρίας περιήγησης.**

**Εξήγηση ειδικού κώδικα (ένα από τα έξι ερωτήματα):**

private void displayDaysRemainingperLoanedBook() {  
 queryResultTable.removeAllViews();  
  
 // Add header row  
 TableRow headerRow = new TableRow(requireContext());  
 headerRow.setLayoutParams(new TableRow.LayoutParams(TableRow.LayoutParams.*MATCH\_PARENT*, TableRow.LayoutParams.*WRAP\_CONTENT*));  
  
 TextView bookTitleHeader = new TextView(requireContext());  
 bookTitleHeader.setText("Book's\nTitle");  
 bookTitleHeader.setPadding(5, 5, 5, 5);  
 bookTitleHeader.setTextSize(TypedValue.*COMPLEX\_UNIT\_SP*, 18); // Set text size  
 bookTitleHeader.setBackgroundResource(R.drawable.*table\_row\_bg*);  
 bookTitleHeader.setLayoutParams(new TableRow.LayoutParams(0, TableRow.LayoutParams.*WRAP\_CONTENT*, 1f)); // Adjust width to take full space  
 bookTitleHeader.setTypeface(null, Typeface.*BOLD*); // Make text bold  
 bookTitleHeader.setLines(2);  
 headerRow.addView(bookTitleHeader);  
  
 TextView memberNameHeader = new TextView(requireContext());  
 memberNameHeader.setText("Loaner's\nName");  
 memberNameHeader.setPadding(5, 5, 5, 5);  
 memberNameHeader.setTextSize(TypedValue.*COMPLEX\_UNIT\_SP*, 18); // Set text size  
 memberNameHeader.setBackgroundResource(R.drawable.*table\_row\_bg*);  
 memberNameHeader.setLayoutParams(new TableRow.LayoutParams(0, TableRow.LayoutParams.*WRAP\_CONTENT*, 1f)); // Adjust width to take full space  
 memberNameHeader.setTypeface(null, Typeface.*BOLD*); // Make text bold  
 memberNameHeader.setLines(2);  
 headerRow.addView(memberNameHeader);  
  
 TextView daysRemainingHeader = new TextView(requireContext());  
 daysRemainingHeader.setText("Days\nRemaining");  
 daysRemainingHeader.setPadding(5, 5, 5, 5);  
 daysRemainingHeader.setTextSize(TypedValue.*COMPLEX\_UNIT\_SP*, 18); // Set text size  
 daysRemainingHeader.setBackgroundResource(R.drawable.*table\_row\_bg*);  
 daysRemainingHeader.setLayoutParams(new TableRow.LayoutParams(0, TableRow.LayoutParams.*WRAP\_CONTENT*, 1f)); // Adjust width to take full space  
 daysRemainingHeader.setTypeface(null, Typeface.*BOLD*); // Make text bold  
 daysRemainingHeader.setLines(2);  
 headerRow.addView(daysRemainingHeader);  
  
 queryResultTable.addView(headerRow);  
  
 // Retrieve the loaned books and their due dates  
 loanCollectionRef.get()  
 .addOnSuccessListener(queryDocumentSnapshots -> {  
 for (QueryDocumentSnapshot documentSnapshot : queryDocumentSnapshots) {  
 // Get the bookId and dueDate fields from each loan document  
 Integer bookId = documentSnapshot.getLong("bookId").intValue();  
 Integer memberId = documentSnapshot.getLong("memberId").intValue();  
 String dueDateString = documentSnapshot.getString("dueDate");  
  
 Book book = Drawer.*database*.bookDao().getBookById(bookId);  
 String name = Drawer.*database*.memberDao().getMemberName(memberId);  
  
  
 // Parse due date string to Date object  
 SimpleDateFormat dateFormat = new SimpleDateFormat("dd/MM/yyyy", Locale.*getDefault*());  
 Date dueDate;  
 try {  
 dueDate = dateFormat.parse(dueDateString);  
 } catch (ParseException e) {  
 Log.*e*("TAG", "Error parsing due date: " + e.getMessage());  
 return;  
 }  
  
 // Calculate days remaining until due date  
 long currentTimeMillis = System.*currentTimeMillis*();  
 long dueDateTimeMillis = dueDate.getTime();  
 long timeDiffMillis = dueDateTimeMillis - currentTimeMillis;  
 int daysRemaining = (int) TimeUnit.*MILLISECONDS*.toDays(timeDiffMillis);  
  
 // Update daysRemaining text if negative  
 String daysRemainingText;  
 if (daysRemaining < 0) {  
 daysRemainingText = "Expired";  
 } else {  
 daysRemainingText = String.*valueOf*(daysRemaining);  
 }  
  
 if (book != null) {  
 String title = book.getTitle();  
  
 // Add row for each loaned book  
 TableRow row = new TableRow(requireContext());  
 row.setLayoutParams(new TableRow.LayoutParams(TableRow.LayoutParams.*MATCH\_PARENT*, TableRow.LayoutParams.*WRAP\_CONTENT*));  
  
 TextView bookTitleTextView = new TextView(requireContext());  
 bookTitleTextView.setText(title);  
 bookTitleTextView.setPadding(5, 5, 5, 5);  
 bookTitleTextView.setTextSize(TypedValue.*COMPLEX\_UNIT\_SP*, 18); // Set text size  
 bookTitleTextView.setBackgroundResource(R.drawable.*table\_row\_bg*);  
 bookTitleTextView.setLayoutParams(new TableRow.LayoutParams(0, TableRow.LayoutParams.*WRAP\_CONTENT*, 1f)); // Adjust width to take full space  
 row.addView(bookTitleTextView);  
  
 TextView memberNameTextView = new TextView(requireContext());  
 memberNameTextView.setText(name);  
 memberNameTextView.setPadding(5, 5, 5, 5);  
 memberNameTextView.setTextSize(TypedValue.*COMPLEX\_UNIT\_SP*, 18); // Set text size  
 memberNameTextView.setBackgroundResource(R.drawable.*table\_row\_bg*);  
 memberNameTextView.setLayoutParams(new TableRow.LayoutParams(0, TableRow.LayoutParams.*WRAP\_CONTENT*, 1f)); // Adjust width to take full space  
 row.addView(memberNameTextView);  
  
 TextView daysRemainingTextView = new TextView(requireContext());  
 daysRemainingTextView.setText(daysRemainingText);  
 daysRemainingTextView.setPadding(5, 5, 5, 5);  
 daysRemainingTextView.setTextSize(TypedValue.*COMPLEX\_UNIT\_SP*, 18); // Set text size  
 daysRemainingTextView.setBackgroundResource(R.drawable.*table\_row\_bg*);  
 daysRemainingTextView.setLayoutParams(new TableRow.LayoutParams(0, TableRow.LayoutParams.*WRAP\_CONTENT*, 1f)); // Adjust width to take full space  
 row.addView(daysRemainingTextView);  
  
 // Add row to the table  
 queryResultTable.addView(row);  
 } else {  
 Log.*e*("TAG", "Book object is null");  
 }  
 }  
 })  
 .addOnFailureListener(e -> {  
 // Handle failure  
 Toast.*makeText*(requireContext(), "Failed to retrieve loaned books: " + e.getMessage(), Toast.*LENGTH\_SHORT*).show();  
 });  
}

**Αυτός ο κώδικας είναι μια μέθοδος που έχει σχεδιαστεί για να εμφανίζει τα δανεισμένα βιβλία μαζί με τις ημέρες που απομένουν μέχρι την ημερομηνία λήξης τους. Ας αναλύσουμε τι κάνει βήμα προς βήμα:**

* **`queryResultTable.removeAllViews()`: Αυτή η γραμμή αφαιρεί όλες τις προβολές (γραμμές) από τη διάταξη του πίνακα όπου θα εμφανίζονται τα δανεισμένα βιβλία.**
* **Στη συνέχεια, ο κώδικας δημιουργεί μια γραμμή κεφαλίδας για τον πίνακα, με στήλες για τον τίτλο του βιβλίου, το όνομα του δανειστή και τις ημέρες που απομένουν.**
* **Μετά τη δημιουργία της γραμμής επικεφαλίδας, η μέθοδος ανακτά τα δανεισμένα βιβλία και τις ημερομηνίες λήξης τους από μια βάση δεδομένων Firestore. Διεξάγει ερωτήματα στην `loanCollectionRef` για τα έγγραφα δανεισμού.**
* **Για κάθε έγγραφο δανεισμού που ανακτάται:**
  + **Εξάγει το αναγνωριστικό βιβλίου, το αναγνωριστικό μέλους και την ημερομηνία λήξης.**
  + **Αντλεί το αντικείμενο βιβλίου και το όνομα μέλους που αντιστοιχούν στα αναγνωριστικά.**
  + **Αναλύει τη συμβολοσειρά της ημερομηνίας λήξης σε αντικείμενο `Date`.**
  + **Υπολογίζει τον αριθμό των ημερών που απομένουν μέχρι την ημερομηνία λήξης βρίσκοντας τη διαφορά μεταξύ της τρέχουσας ώρας και της ημερομηνίας λήξης.**
  + **Προετοιμάζει το κείμενο που θα εμφανιστεί για τις ημέρες που απομένουν, επισημαίνοντας το ως "Expired" (Λήξη) εάν το βιβλίο είναι ληξιπρόθεσμο.**
* **Για κάθε δανεισμένο βιβλίο, δημιουργεί μια νέα γραμμή στη διάταξη του πίνακα και τη συμπληρώνει με τον τίτλο του βιβλίου, το όνομα του μέλους και τις ημέρες που απομένουν.**
* **Η μέθοδος χειρίζεται τυχόν αποτυχίες που προκύπτουν κατά τη διαδικασία αναζήτησης στη βάση δεδομένων, εμφανίζοντας ένα μήνυμα σφάλματος σε ένα τοστ.**

**Συνολικά, αυτός ο κώδικας συμπληρώνει δυναμικά μια διάταξη πίνακα με πληροφορίες δανεισμένων βιβλίων και ενημερώνει τις ημέρες που απομένουν σε πραγματικό χρόνο, αντικατοπτρίζοντας τυχόν αλλαγές στις ημερομηνίες λήξης.**

**Ο περισσότερος κώδικας από αυτόν επαναλαμβάνεται και στα άλλα 5 ερωτήματα το μόνο που αλλάζει είναι τα δεδομένα που παίρνονται και πως χρησιμοποιούνται.**

# **εΙΚΟΝΕς ΑΠΟ ΤΑ ΜΕΡΟΙ ΤΗΣ ΕΦΑΡΜΟΓΗΣ**

Εικόνα Οθόνης Σύνδεσης:

**A screenshot of a login screen

Description automatically generated**

**Εικόνες του Drawer** **στην Αρχική Οθόνη:**

**A screenshot of a phone

Description automatically generated**

**A screenshot of a phone

Description automatically generated**

**Εικόνα του Toolbar στην Αρχική Οθόνη:**

**A black screen with a black background

Description automatically generated**

**Εικόνα του MailButton:**

**A yellow square with a blue and purple envelope

Description automatically generated**

**Εικόνα της Αρχικής Οθόνης:**

**A screenshot of a phone

Description automatically generated**

**Εικόνες από τα Fragments για την Διαχείριση Βιβλίου:**

**A screenshot of a phone

Description automatically generatedA screenshot of a bookcase

Description automatically generated**

**A screenshot of a bookcase

Description automatically generated**

**Εικόνες από τα Fragments για την Διαχείριση Μελών:**

**A screenshot of a phone

Description automatically generatedA screenshot of a login form

Description automatically generated**

**A screenshot of a room with bookshelves and a member

Description automatically generated**

**Εικόνες από τα Fragments για την Διαχείριση Τμημάτων:**

**A screenshot of a library

Description automatically generatedA screenshot of a screen

Description automatically generated**

**A screenshot of a library

Description automatically generated**

**Εικόνες από τα Fragments για την Διαχείριση Δανείων:**

**A screenshot of a loan application

Description automatically generatedA screenshot of a book store

Description automatically generated**

A screenshot of a library

Description automatically generated