**DOCUMENTAZIONE DEL PROGETTO DI   
LABORATORIO DI APPLICAZIONI MOBILI ANDROID**

**GIOVANNI MARIA SAVOCA – 970094 –** [**GIOVANNIMARIA.SAVOCA@STUDIO.UNIBO.IT**](mailto:GIOVANNIMARIA.SAVOCA@STUDIO.UNIBO.IT)

**INFORMATICA PER IL MANAGEMENT 2023**

**Librerie utilizzate:**

com.google.code.gson:gson:2.8.9

Descrizione: Gson è una libreria per la conversione di oggetti Java in formato JSON e viceversa. È ampiamente utilizzata per analizzare le risposte JSON dai servizi web o per serializzare oggetti Java in formato JSON. Utilizzo tipico: Parsing di dati JSON da API web o formattazione di oggetti Java in JSON.

com.google.android.gms:play-services-vision:20.1.3

Descrizione: Questa libreria offre funzionalità di visione e riconoscimento di immagini. Include strumenti per il rilevamento di oggetti, il riconoscimento ottico dei caratteri (OCR) e altre funzionalità legate alla visione computazionale. Utilizzo tipico: Riconoscimento di oggetti o testo nelle immagini.

com.google.android.gms:play-services-location:21.0.1

Descrizione: Questa libreria fornisce servizi di localizzazione per le app Android, inclusi il tracciamento della posizione GPS e altre funzionalità legate alla posizione. Utilizzo tipico: Accesso alla posizione dell'utente o definizione di aree geografiche specifiche per attivare azioni.

com.github.PhilJay:MPAndroidChart:v3.1.0

Descrizione: MPAndroidChart è una libreria per la creazione di grafici personalizzati nelle tue app Android. Supporta vari tipi di grafici, come grafici a barre, a torta, a dispersione e molti altri. Utilizzo tipico: Visualizzazione dei dati in forma grafica attraverso vari tipi di grafici.

com.google.firebase:firebase-ml-vision:24.1.0

Descrizione: Firebase ML Vision è una libreria di Firebase che fornisce funzionalità di riconoscimento di immagini, testo e altri elementi utilizzando modelli di machine learning pre-addestrati. Utilizzo tipico: Riconoscimento di oggetti, testo o etichette nelle immagini.

com.google.firebase:firebase-bom:32.4.0

Descrizione: Firebase Bill of Materials (BoM) consente di gestire le versioni delle librerie Firebase in modo coerente nel tuo progetto. È uno strumento che semplifica la gestione delle versioni delle librerie Firebase. Utilizzo tipico: Gestione delle versioni delle librerie Firebase nel tuo progetto.

com.theartofdev.edmodo:android-image-cropper:2.8.0

Descrizione: Questa libreria consente di aggiungere funzionalità di ritaglio delle immagini alle tue app Android. Gli utenti possono ritagliare e modificare le immagini all'interno dell'app. Utilizzo tipico: Consentire agli utenti di ritagliare e modificare le immagini all'interno dell'app.

com.google.android.gms:play-services-maps:18.2.0

Descrizione: Questa libreria offre funzionalità di mappatura e visualizzazione di mappe interattive nelle tue app Android. Puoi visualizzare mappe, posizionare marcatori e molto altro. Utilizzo tipico: Visualizzazione di mappe e integrazione di funzionalità legate alle mappe nell'app.

**Classe Account**

La classe Account rappresenta un account finanziario e tiene traccia del nome dell'account, del saldo e delle transazioni associate. L'account può essere utilizzato per tenere traccia di entrate e uscite finanziarie.

Costruttori

Account(): Crea un account con un nome predefinito "Accountempty" e un saldo iniziale di 0.

Account(String name): Crea un account con un nome specificato e un saldo iniziale di 0.

Account(String name, ArrayList<Transactions> listTrans): Crea un account con un nome specificato e una lista di transazioni fornita come parametro.

Metodi principali

getName(): Restituisce il nome dell'account.

getBalance(): Restituisce il saldo dell'account con un massimo di due posizioni decimali.

getListTrans(): Restituisce la lista delle transazioni associate a questo account.

addTransaction(Transactions transaction): Aggiunge una transazione alla lista e aggiorna il saldo.

removeTransaction(Transactions transaction): Rimuove una transazione dalla lista e aggiorna il saldo.

editTransaction(Transactions transactionOriginal, Transactions newTrans): Modifica una transazione esistente nella lista e aggiorna il saldo.

updateBalance(): Aggiorna il saldo dell'account in base alle transazioni correnti.

toString(): Restituisce una rappresentazione testuale dell'account.

Enum CategoriesEnum

L'enum CategoriesEnum rappresenta le categorie possibili per le transazioni finanziarie. Queste categorie possono includere "Cibo e Bevande", "Shopping", "Casa", "Trasporti", "Svago e Intrattenimento", "Comunicazione e PC", "Stipendio", "Regali" e "Altro".

**Classe City**

La classe City rappresenta una città con un nome, una latitudine e una longitudine.

Costruttori

City(): Crea una città predefinita con il nome "Bologna" e le coordinate latitudine e longitudine predefinite.

City(String nameCity, double latitude, double longitude): Crea una città con un nome specificato e le coordinate specificate.

Metodi principali

getNameCity(): Restituisce il nome della città.

getLatitude(): Restituisce la latitudine della città.

getLongitude(): Restituisce la longitudine della città.

toString(): Restituisce una rappresentazione testuale della città.

**Classe Transactions**

La classe Transactions rappresenta una transazione finanziaria, con informazioni su tipo di transazione (entrata o uscita), importo, data, città e categoria.

Costruttori

Transactions(): Crea una transazione predefinita con importo minimo, data attuale e città predefinita.

Transactions(Boolean income, double amount, Calendar date, City city, CategoriesEnum category): Crea una transazione con i dettagli specificati.

Metodi principali

getCategory(): Restituisce la categoria della transazione.

isIncome(): Verifica se la transazione è un'entrata.

getAmount(): Restituisce l'importo della transazione, positivo per le entrate e negativo per le uscite.

getAmountValue(): Restituisce l'importo della transazione come valore assoluto.

getDate(): Restituisce la data della transazione.

getCity(): Restituisce la città associata alla transazione.

setIncome(Boolean income): Imposta il tipo di transazione (entrata o uscita).

setAmount(double amount): Imposta l'importo della transazione.

setDate(Calendar date): Imposta la data della transazione.

setCity(City city): Imposta la città associata alla transazione.

setCategory(CategoriesEnum category): Imposta la categoria della transazione.

toString(): Restituisce una rappresentazione testuale della transazione.

printOnApp(): Restituisce una rappresentazione testuale formattata per l'applicazione.

**Classe MainActivity**

La classe MainActivity rappresenta l'attività principale dell'app CashFlow. Questa attività è responsabile di inizializzare l'app, gestire le autorizzazioni, leggere e scrivere dati JSON, e visualizzare il fragment principale.

Metodi principali

onCreate(Bundle savedInstanceState): Metodo chiamato quando l'attività viene creata. Inizializza l'app, legge i dati dai file JSON e imposta il pulsante "Home" per visualizzare il fragment principale.

onRequestPermissionsResult(int requestCode, String[] permissions, int[] grantResults): Metodo chiamato quando viene richiesta un'autorizzazione. Gestisce la risposta all'autorizzazione da parte dell'utente.

loadFragment(Fragment fragment): Carica un fragment nell'attività.

**Classe Test**

La classe Test è una classe di test utilizzata per creare una lista di account con transazioni di esempio.

Metodi principali

Test(): Costruttore che crea una lista di account e transazioni di esempio e stampa i dettagli degli account.

getList(): Restituisce la lista di account creata nel costruttore.

**Classe JsonReadWrite**

La classe JsonReadWrite gestisce la lettura e la scrittura di dati in formato JSON. È responsabile di convertire oggetti in formato JSON e viceversa.

Metodi principali

JsonReadWrite(String fileName): Costruttore che inizializza l'oggetto con il nome del file JSON.

JsonReadWrite(ArrayList<Account> accounts, String fileName): Costruttore che inizializza l'oggetto con una lista di account e il nome del file JSON.

setList(ArrayList<Account> accounts, Context context): Imposta la lista di account e salva i dati in formato JSON.

saveToJson(Context context): Salva i dati in formato JSON in un file.

accountsToJson(): Converte la lista di account in una stringa JSON.

readAccountsFromJson(Context context): Legge i dati JSON dal file e restituisce una lista di account.

readJsonFromFile(Context context): Legge il contenuto del file JSON.

**Classe HomeFragment**

La classe HomeFragment rappresenta il fragment principale dell'app CashFlow. Mostra una griglia di account e calcola il saldo totale degli account.

Metodi principali

onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState): Crea la vista del fragment, carica la griglia degli account e gestisce i clic sui pulsanti.

onResume(): Calcola il saldo totale degli account e aggiorna la vista.

getSubtotal(): Calcola il saldo totale degli account.

openNewAccountFragment(): Apre il fragment per creare un nuovo account.

openTransactionFragment(): Apre il fragment per aggiungere una nuova transazione.

openStatisticsFragment(): Apre il fragment per visualizzare le statistiche degli account.

**Classe AccountDetailsFragment**

La classe AccountDetailsFragment rappresenta un fragment utilizzato per visualizzare i dettagli di un account finanziario, tra cui il nome dell'account, il saldo, le transazioni associate e le opzioni per la modifica e l'eliminazione dell'account o delle transazioni. Questa classe è parte dell'applicazione CashFlow.

Costruttore:

AccountDetailsFragment(Account account): Crea un nuovo AccountDetailsFragment associato a un account finanziario specificato. Questo costruttore richiede un oggetto Account come parametro.

Metodi principali:

onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState): Questo metodo è chiamato quando il fragment viene creato e inizializza l'interfaccia utente per visualizzare i dettagli dell'account, le transazioni e i pulsanti per l'aggiornamento o l'eliminazione dell'account.

changeName(): Questo metodo gestisce la modifica del nome dell'account e l'aggiornamento dei dati nel file JSON.

doesAccountExist(ArrayList<Account> accounts, String name): Verifica se esiste già un account con lo stesso nome nella lista di account.

findAccountIndex(ArrayList<Account> accounts, String oldName): Restituisce l'indice dell'account nella lista in base al nome specificato.

showDeleteConfirmationDialog(): Visualizza un dialog di conferma per l'eliminazione dell'account.

deleteAccount(): Gestisce l'eliminazione dell'account e l'aggiornamento dei dati JSON.

TransactionListAdapter: Una classe interna per gestire la visualizzazione delle transazioni associate a questo account in un elenco.

**Classe EditTransactionFragment**

La classe EditTransactionFragment rappresenta un fragment utilizzato per modificare i dettagli di una transazione finanziaria, tra cui il tipo di transazione (entrata o uscita), l'importo, la data, la categoria e la città associata. Questa classe è parte dell'applicazione CashFlow.

Costruttore:

EditTransactionFragment(Transactions transaction, Account account): Crea un nuovo EditTransactionFragment associato a una transazione finanziaria e a un account specificati. Questo costruttore richiede un oggetto Transactions e un oggetto Account come parametri.

Metodi principali:

onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState): Questo metodo è chiamato quando il fragment viene creato e inizializza l'interfaccia utente per la modifica dei dettagli della transazione, inclusi il tipo, l'importo, la data, la categoria, la città e il salvataggio dei dati.

setExpense(): Imposta il tipo di transazione come "uscita".

setIncome(): Imposta il tipo di transazione come "entrata".

showDatePickerDialog(): Mostra un dialog per la selezione della data.

deleteTransaction(): Gestisce l'eliminazione della transazione e l'aggiornamento dei dati JSON.

updateTransaction(): Gestisce l'aggiornamento dei dettagli della transazione e l'aggiornamento dei dati JSON.

openCamera(): Apre la fotocamera per catturare un'immagine.

openGallery(): Apre la galleria per selezionare un'immagine.

onActivityResult(int requestCode, int resultCode, Intent data): Gestisce l'output dell'acquisizione dell'immagine e del cropping dell'immagine.

showDeleteConfirmationDialog(): Visualizza un dialog di conferma per l'eliminazione della transazione.

**Classe NewTransactionFragment**

La classe NewTransactionFragment rappresenta un fragment utilizzato per l'aggiunta di nuove transazioni finanziarie nell'applicazione CashFlow.

Costruttore:

NewTransactionFragment(ArrayList<Account> accounts, City cityPosition): Crea un nuovo NewTransactionFragment associato a un elenco di account e alla posizione di una città specificati. Questo costruttore richiede una lista di oggetti Account e un oggetto City come parametri.

Metodi principali:

onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState): Questo metodo è chiamato quando il fragment viene creato e inizializza l'interfaccia utente per l'aggiunta di nuove transazioni finanziarie. Questa include la selezione del tipo di transazione (entrata o uscita), l'inserimento dell'importo, la scelta della data, la selezione della categoria, la selezione dell'account, l'acquisizione di dati sulla posizione, il caricamento di un'immagine dalla fotocamera o dalla galleria e il salvataggio dei dati.

openCamera(): Apre l'applicazione della fotocamera per catturare un'immagine.

openGallery(): Apre l'applicazione della galleria per selezionare un'immagine.

onActivityResult(int requestCode, int resultCode, Intent data): Gestisce l'output dell'acquisizione dell'immagine e del ritaglio dell'immagine. Dopo aver catturato o selezionato un'immagine, questa viene ritagliata utilizzando il modulo di ritaglio "android-image-cropper". Il testo risultante dall'immagine viene elaborato da un OCRManager.

saveTransaction(): Gestisce il salvataggio di una nuova transazione. Questo metodo esegue varie verifiche sui dati inseriti (tipo di transazione, importo, data) e, se tutti i dati sono validi, salva la transazione nell'account selezionato. La data e l'importo vengono estrapolati dai campi dell'interfaccia utente. La transazione viene quindi aggiunta all'account corrispondente e l'account viene aggiornato con il nuovo saldo. Infine, i dati vengono salvati in un file JSON.

showDatePickerDialog(): Mostra un dialog per la selezione della data da parte dell'utente.

La classe NewTransactionFragment è progettata per consentire all'utente di inserire nuove transazioni finanziarie nel sistema CashFlow, inclusa la possibilità di acquisire l'importo direttamente da un'immagine e di selezionare la data da un calendario. Prima di utilizzare questa classe, assicurati che le risorse e le dipendenze esterne siano correttamente configurate nell'applicazione.

**Classe Posizione**

La classe Posizione è utilizzata nell'applicazione CashFlow per acquisire la posizione geografica del dispositivo dell'utente, determinare la città corrispondente e restituire i dati della città attraverso un callback.

Costruttore:

Posizione(Context context): Crea un nuovo oggetto Posizione associato a un contesto specificato. Questo costruttore richiede un oggetto Context come parametro.

Metodi principali:

requestDeviceLocation(DeviceLocationCallback callback): Richiede la posizione del dispositivo e restituisce i dati della città corrispondente tramite un callback. Prima di recuperare la posizione, verifica se l'applicazione ha l'autorizzazione per accedere alla posizione. Se l'autorizzazione è concessa, viene utilizzato il servizio di localizzazione Fused Location Provider per ottenere l'ultima posizione nota del dispositivo. Successivamente, la posizione viene convertita in una città utilizzando il servizio Geocoder e i dati vengono restituiti attraverso il callback.

interface DeviceLocationCallback: Questa interfaccia definisce due metodi che devono essere implementati:

onLocationFetched(City city): Chiamato quando la posizione è stata acquisita con successo e i dati della città sono disponibili.

onLocationFetchFailed(Exception e): Chiamato quando si verifica un errore durante il recupero della posizione.

La classe Posizione è fondamentale per ottenere informazioni sulla posizione del dispositivo, che è utilizzata per fornire dettagli sulla città in cui viene effettuata una transazione finanziaria.

**Classe OCRManager**

La classe OCRManager è parte dell'applicazione CashFlow ed è responsabile dell'elaborazione di immagini contenenti testo per l'estrazione del testo. Questa classe utilizza il servizio Firebase ML Vision per riconoscere il testo presente in un'immagine e restituirlo tramite un callback.

Costruttore:

OCRManager(Context context): Crea un nuovo oggetto OCRManager associato a un contesto specificato. Questo costruttore richiede un oggetto Context come parametro.

Metodi principali:

processImage(Uri imageUri, OCRListener listener): Elabora un'immagine contenente testo, identificata tramite un'URI, per l'estrazione del testo. Questo metodo utilizza il servizio Firebase ML Vision per riconoscere il testo nell'immagine. Una volta completato il processo di riconoscimento del testo, i dati estratti vengono restituiti tramite un callback.

interface OCRListener: Questa interfaccia definisce due metodi che devono essere implementati:

onTextRecognized(String text): Chiamato quando il testo nell'immagine è stato riconosciuto con successo e restituito.

onFailure(Exception e): Chiamato quando si verifica un errore durante il riconoscimento del testo nell'immagine.

La classe OCRManager è fondamentale per consentire all'utente di acquisire automaticamente l'importo da un'immagine di uno scontrino o di un documento, semplificando l'aggiunta di nuove transazioni finanziarie nell'applicazione CashFlow.

**Classe StatisticsFragment**

La classe StatisticsFragment è parte dell'applicazione CashFlow e rappresenta un fragment utilizzato per la visualizzazione delle statistiche relative agli account finanziari. Queste statistiche includono la visualizzazione di grafici a linee, la mappa delle transazioni finanziarie, nonché le entrate e le uscite totali.

Costruttore:

StatisticsFragment(ArrayList<Account> accounts): Crea un nuovo oggetto StatisticsFragment associato a un elenco di account finanziari specificati. Questo costruttore richiede un'ArrayList di oggetti Account come parametro.

Metodi principali:

onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState): Questo metodo è chiamato quando il fragment viene creato e inizializza l'interfaccia utente per la visualizzazione delle statistiche. Vengono creati pulsanti per accedere alle diverse visualizzazioni statistiche, come il grafico a linee, la mappa delle transazioni, le entrate e le uscite totali.

Il fragment consente all'utente di selezionare una visualizzazione statistica specifica facendo clic sui pulsanti corrispondenti. La classe utilizza il concetto di fragment replacement per sostituire il fragment corrente con il fragment desiderato quando si fa clic su un pulsante.

**Classe MapFragment**

La classe MapFragment è parte dell'applicazione CashFlow ed è un fragment utilizzato per visualizzare una mappa con marcatori che rappresentano le città associate alle transazioni finanziarie degli account dell'utente.

Costruttore:

MapFragment(ArrayList<Account> accounts): Crea un nuovo oggetto MapFragment associato a un elenco di account finanziari specificati. Questo costruttore richiede un'ArrayList di oggetti Account come parametro.

Metodi principali:

onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState): Questo metodo è chiamato quando il fragment viene creato e inizializza l'interfaccia utente per la visualizzazione della mappa delle transazioni finanziarie. Utilizza la libreria Google Maps Android per mostrare una mappa interattiva.

onMapReady(GoogleMap googleMap): Questo metodo viene chiamato quando la mappa è pronta per essere utilizzata. Aggiunge i marcatori sulla mappa corrispondenti alle città delle transazioni finanziarie presenti negli account. Inoltre, muove la telecamera verso una posizione adeguata sulla mappa.

getMarkerData(): Restituisce un elenco di oggetti MarkerOptions che rappresentano i marcatori sulla mappa. Oltre ai marcatori delle città delle transazioni finanziarie, aggiunge un marcatore di test per scopi di test.

getCitiesFromAccounts(): Estrae le città associate alle transazioni finanziarie dagli account finanziari. Restituisce un elenco di oggetti MarkerOptions contenenti i dati delle città.

La classe MapFragment consente all'utente di visualizzare geograficamente le posizioni delle transazioni finanziarie sugli account finanziari sulla mappa. I marcatori mostrano il nome delle città associate alle transazioni.

**Classe Line\_chart**

La classe Line\_chart è parte dell'applicazione CashFlow e rappresenta un fragment utilizzato per visualizzare un grafico a linee delle transazioni finanziarie in base alle date di inizio e fine selezionate dall'utente.

Costruttore:

Line\_chart(ArrayList<Account> accounts): Crea un nuovo oggetto Line\_chart associato a un elenco di account finanziari specificati. Questo costruttore richiede un'ArrayList di oggetti Account come parametro.

Metodi principali:

onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState): Questo metodo è chiamato quando il fragment viene creato e inizializza l'interfaccia utente per la visualizzazione del grafico a linee. L'interfaccia utente contiene pulsanti per aprire il selettore di date di inizio e fine, nonché un pulsante per generare il grafico. Inoltre, vengono visualizzati i TextView per mostrare le date di inizio e fine selezionate dall'utente.

openStartDatePicker(): Apre un selettore di date per la selezione della data di inizio. La data di inizio deve essere precedente alla data di fine.

openEndDatePicker(): Apre un selettore di date per la selezione della data di fine. La data di fine deve essere successiva alla data di inizio.

createLineChart(): Genera il grafico a linee delle transazioni finanziarie in base alle date di inizio e fine selezionate dall'utente. Il grafico mostra il saldo giornaliero, con le date sull'asse X e il saldo sull'asse Y.

generateDataEntries(Calendar startDate, Calendar endDate): Genera le voci di dati per il grafico a linee. Calcola il saldo giornaliero in base alle transazioni finanziarie presenti negli account finanziari. Restituisce un elenco di oggetti Entry, che rappresentano le voci di dati del grafico.

isSameDay(Calendar date1, Calendar date2): Verifica se due date sono dello stesso giorno, confrontando l'anno, il mese e il giorno.

formatDateKey(Calendar date): Formatta una data in una stringa nel formato "dd-MM-yyyy".

formatDateKeyWithoutYear(Calendar date): Formatta una data in una stringa nel formato "dd-MM", senza l'anno.

La classe Line\_chart consente all'utente di selezionare un intervallo di date e generare un grafico a linee che mostra il saldo giornaliero delle transazioni finanziarie in base a tali date. Il grafico offre una visualizzazione chiara e intuitiva dei dati finanziari.

**Classe Income\_expense**

La classe Income\_expense è parte dell'applicazione CashFlow e rappresenta un fragment utilizzato per visualizzare i dati finanziari relativi ai guadagni o alle spese, in base alla selezione dell'utente. Questo fragment consente all'utente di selezionare gli account finanziari di interesse e mostra grafici a torta e a barre basati sui dati di guadagno o spesa delle categorie.

Costruttore:

Income\_expense(Boolean isIncome, ArrayList<Account> accounts): Crea un nuovo oggetto Income\_expense associato a un valore booleano (true per i guadagni, false per le spese) e a un elenco di account finanziari specificati. Questo costruttore richiede un booleano e un'ArrayList di oggetti Account come parametri.

Metodi principali:

onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState): Questo metodo è chiamato quando il fragment viene creato e inizializza l'interfaccia utente per la visualizzazione dei dati finanziari. L'interfaccia utente contiene un checkbox per selezionare o deselezionare tutti gli account finanziari, una ListView per elencare gli account finanziari e due grafici, uno a torta e uno a barre.

updateSelectedAccounts(): Aggiorna l'elenco degli account finanziari selezionati in base alla selezione dell'utente nella ListView.

initPieChart(ArrayList<Account> selectedAccounts): Inizializza il grafico a torta con i dati finanziari relativi agli account finanziari selezionati. Il grafico a torta mostra la suddivisione delle categorie di guadagno o spesa.

initBarChart(ArrayList<Account> selectedAccounts): Inizializza il grafico a barre con i dati finanziari relativi agli account finanziari selezionati. Il grafico a barre mostra il totale di guadagni o spese per ciascuna categoria.

getIncomeOrExpensePieData(ArrayList<Account> accounts): Restituisce un elenco di oggetti PieEntry contenenti i dati finanziari per il grafico a torta. Calcola il totale di guadagni o spese per ciascuna categoria.

getIncomeOrExpenseBarData(ArrayList<Account> accounts): Restituisce un elenco di oggetti BarEntry contenenti i dati finanziari per il grafico a barre. Calcola il totale di guadagni o spese per ciascuna categoria.

getCategoryColors(): Restituisce un array di colori corrispondenti alle categorie finanziarie. Ogni categoria è associata a un colore univoco.

La classe Income\_expense offre un'interfaccia utente intuitiva per esplorare i dati finanziari relativi ai guadagni o alle spese, consentendo all'utente di selezionare gli account di interesse. I grafici forniti consentono una chiara visualizzazione dei dati finanziari e delle categorie coinvolte.

ALCUNE SPECIFICHE CHE POSSO ESSERE IMPLEMENTATE PER ESTENDERE IL PROGETTO

* Database Online con la scelta di profilo per ogni utente.
* Archivio di immagini scattate
* Pianificazione di future transazioni e pagamenti periodici
* Piano d’accumulo
* Tasso di valuta e valuta per ogni conto
* Conti condivisi con più utenti

Ecco alcune ulteriori funzionalità che potrebbero essere implementate per arricchire il tuo progetto di applicazione mobile CashFlow:

1. Database Online con Profili Utente: Implementa un sistema di autenticazione utente e un database online per consentire agli utenti di creare e gestire i propri profili. Ogni utente dovrebbe poter accedere ai propri dati finanziari in modo sicuro. Puoi utilizzare servizi come Firebase Authentication e Firebase Realtime Database o un database SQL online per questo scopo.
2. Archivio di Immagini Scattate: Aggiungi la possibilità di archiviare immagini associate alle transazioni. Gli utenti potrebbero scattare foto di ricevute o documenti correlati alle transazioni finanziarie e associarle direttamente all'account o alla transazione.
3. Pianificazione di Future Transazioni e Pagamenti Periodici: Implementa una funzionalità che consente agli utenti di pianificare transazioni future e pagamenti periodici, come bollette mensili o rate di prestiti. Questa funzionalità dovrebbe consentire agli utenti di programmare le date e gli importi delle future transazioni.
4. Piano d'Accumulo: Aggiungi la possibilità per gli utenti di impostare obiettivi di risparmio o piani d'accumulo. L'app dovrebbe tenere traccia del progresso verso questi obiettivi e mostrare i dettagli sullo stato dei piani d'accumulo.
5. Tasso di Valuta e Valuta per Ogni Conto: Consentire agli utenti di impostare un tasso di cambio per ogni account o per transazioni specifiche. Questo è utile per coloro che gestiscono conti in valute diverse o necessitano di conversioni valutarie.
6. Conti Condivisi con Più Utenti: Implementa la possibilità di condividere un account finanziario con più utenti. Questo è utile per famiglie o gruppi di persone che condividono spese comuni.
7. Notifiche: Aggiungi un sistema di notifiche per avvisare gli utenti di scadenze di pagamento, transazioni importanti o aggiornamenti sullo stato dei loro account finanziari.
8. Backup ed export dei dati: Aggiungere la possibilità di eseguire il backup dei dati e di esportare le transazioni e gli account in diversi formati, ad esempio CSV o PDF, per scopi di archiviazione o condivisione.