DOCUMENTAZIONE DEL PROGETTO DI LABORATORIO DI APPLICAZIONI MOBILI ANDROID

CashFlowApp

GIOVANNI MARIA SAVOCA – 970094

GIOVANNIMARIA.SAVOCA@STUDIO.UNIBO.IT

INFORMATICA PER IL MANAGEMENT 2023

Introduzione al progetto

CashFlowApp è un'applicazione mobile sviluppata per aiutare gli utenti a gestire il flusso di cassa personale, consentendo di registrare le transazioni finanziarie, monitorare le spese e le entrate, visualizzare grafici statistici, e semplificare la gestione delle proprie finanze. L'applicazione offre una serie di funzionalità utili per tenere traccia delle finanze personali, rendendo più semplice il controllo delle spese e il monitoraggio delle entrate. Le transazioni vengono organizzate all'interno di un "Account," ciascuna assegnata a una specifica categoria. Le categorie predefinite includono: "Cibo e Bevande," "Shopping," "Casa," "Trasporti," "Svago," "Comunicazione e PC," "Stipendio," "Regali," e "Altro."

Installazione e Prerequisiti

Prerequisiti:

Prima di procedere all'installazione di CashFlowApp, assicurati di soddisfare i seguenti requisiti:

- Android Studio con le seguenti versioni:
  - Gradle Version: 8.0.
  - Android Gradle Plugin Version: 8.1.2.
- Un dispositivo Android con sistema operativo Android 11+.
- Connessione Internet (necessaria per alcune funzionalità come il riconoscimento del testo da una foto).
- Servizi di localizzazione attivi.
- Autorizzazioni per l'utilizzo della fotocamera e l'accesso alla galleria.

Installazione:

Per installare CashFlowApp, segui i seguenti passi:

- Collega il tuo dispositivo Android al tuo computer utilizzando un cavo USB.
- Avvia Android Studio.
- Apri il progetto CashFlowApp in Android Studio.

- Seleziona il tuo dispositivo Android come target di installazione o utilizza un Virtual Device, un simulatore di un telefono Android disponibile sul tuo computer.
- Fai clic su "Run" o "Build and Run" per installare l'applicazione sul tuo dispositivo.

Funzionalità principali

CashFlowApp offre una serie di funzionalità principali per aiutarti a gestire le tue finanze personali:

1. Registrazione di Transazioni:
Registra le tue transazioni finanziarie, inclusi dettagli come l'importo, la categoria, l'account di destinazione e la data. (NewTransactionFragment.java)

2. Creazione di Account:
Crea account personalizzati per organizzare le tue transazioni. (NewAccountFragment.java)

3. Categorie di Transazioni:
Le transazioni sono categorizzate in base a categorie predefinite come "FoodAndDrinks", "Shopping", "House", "Transport", ecc. (CategoriesEnum.java)

4. Grafici Statistici:
Visualizza grafici statistici temporali sulle spese ed entrate con la possibilità di selezionare le date di inizio e fine desiderate. (StatisticsFragment.java)

5. Grafici a Torta e a Barre:
Genera grafici a torta e a barre per visualizzare la distribuzione delle spese ed entrate in base alle categorie. (Income_expense.java)

6. Esportazione CSV:
Esporta i dati delle transazioni in un file CSV che può essere salvato o condiviso con altri. (StatisticsFragment.java)

7. Visualizzazione su Google Maps:
Visualizza la posizione delle transazioni su Google Maps, utilizzando i dati di localizzazione delle città in cui sono state registrate. (MapFragment.java)

8. Dettagli delle Transazioni:
Visualizza i dettagli delle transazioni e apporta eventuali modifiche quando necessario. (AccountDetailFragment.java - TransactionListAdapter)

9. Modifica Account:
Cambia il nome dell'account o elimina gli account che non ti servono più. (EditTransactionFragment.java)

10. Riconoscimento del Testo da Foto:
Carica una foto di uno scontrino e utilizza il riconoscimento ottico dei caratteri (OCR) di Google per estrarre il testo e aggiungere i dettagli della transazione. (OCRManager.java)

Struttura del progetto

CashFlowApp è strutturato in modo da garantire una gestione efficace dei dati finanziari e una facile interazione dell'utente. La struttura principale dell'applicazione comprende:

- Un'attività principale che include un footer e un header, creando una struttura di base per l'app.
- Fragment per le viste parziali che compongono le diverse schermate dell'app.
- Classi di dati come Account, Transazioni, Città e Categorie per gestire le informazioni finanziarie in modo organizzato.
- Classi di utilità come OCRManager per il riconoscimento del testo da foto e JsonReadWrite per l'uso di JSON per il salvataggio dei dati e la generazione di file CSV per l'esportazione.
- Un'interfaccia grafica utente intuitiva e responsive che semplifica l'interazione con l'applicazione.
- All'interno di *./res/layout* ci sono tutti i file xml che strutturano l'interfaccia grafica dell'applicazione.

La struttura del progetto è progettata per massimizzare la facilità d'uso, consentendo agli utenti di registrare e monitorare le transazioni finanziarie in modo efficiente e accurato.

Quando scarichiamo l'applicazione sul nostro cellulare l'applicazione crea un file json dove salverà tutte le informazioni degli accounts. Se all'inizio da un errore di lettura e/o scrittura del file JSON si deve modificare il codice in MainActivity.java alla riga 34.

```
33        // accounts = jsonReadWrite.readAccountsFromJson(MainActivity.this);
34        accounts = null;
```

Bisogna commentare la riga 33 ed eseguire per una sola volta la riga 34. Il problema è perché non esiste nessun file nel nuovo dispositivo.

Se non viene letto nulla dal Json file, la classe Test.java inizializza 2 nuovi account "Bank" e "Cash" con una transazione INCOME (entrata) di 1000.00€.

Librerie Utilizzate:

Durante lo sviluppo di CashFlowApp, sono state utilizzate diverse librerie per ampliare le funzionalità dell'app:

Gson (com.google.code.gson:gson:2.8.9):

Utilizzata per la conversione di oggetti Java in formato JSON e viceversa. È fondamentale per analizzare risposte JSON da servizi web e per la serializzazione di oggetti Java in formato JSON.

Google Play Services Vision (com.google.android.gms:play-services-vision:20.1.3):

Fornisce funzionalità di visione e riconoscimento di immagini, inclusi il rilevamento di oggetti e il riconoscimento ottico dei caratteri (OCR).

Google Play Services Location (com.google.android.gms:play-services-location:21.0.1):

Offre servizi di localizzazione per consentire all'app di accedere alla posizione dell'utente e attivare azioni basate sulla posizione.

MPAndroidChart (com.github.PhilJay:MPAndroidChart:v3.1.0):

Utilizzata per la creazione di grafici personalizzati nell'app, compresi grafici a barre, a torta, a dispersione e altro ancora.

Firebase ML Vision (com.google.firebase:firebase-ml-vision:24.1.0):

Integrata per fornire funzionalità di riconoscimento di oggetti, testo e etichette nelle immagini utilizzando modelli di machine learning pre-addestrati.

Firebase Bill of Materials (BoM) (com.google.firebase:firebase-bom:32.4.0):

Utilizzata per la gestione coerente delle versioni delle librerie Firebase all'interno del progetto.

Android Image Cropper (com.theartofdev.edmodo:android-image-cropper:2.8.0):

Aggiunta per consentire agli utenti di ritagliare e modificare le immagini all'interno dell'app.

Google Play Services Maps (com.google.android.gms:play-services-maps:18.2.0):

Utilizzata per la visualizzazione di mappe interattive e funzionalità di mappatura all'interno dell'app.

Queste librerie ampliano la funzionalità di CashFlowApp, consentendo agli utenti di sfruttare al meglio le diverse caratteristiche offerte dall'applicazione.

ALCUNE SPECIFICHE CHE POSSONO ESSERE IMPLEMENTATE PER ESTENDERE IL PROGETTO

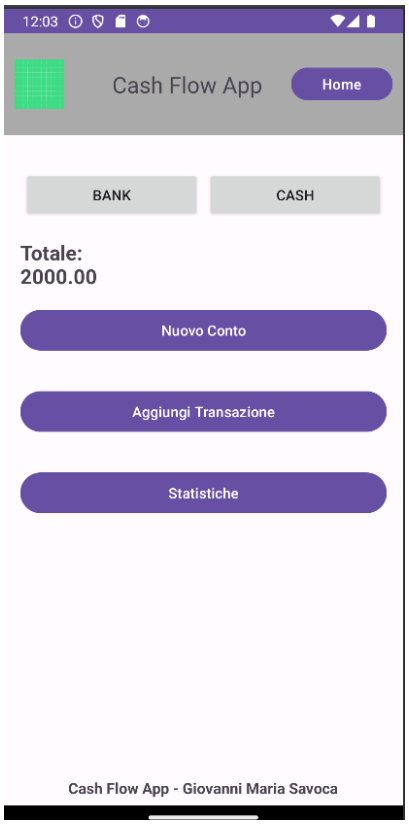
- Database Online con la scelta di profilo per ogni utente.
- Archivio di immagini scattate
- Pianificazione di future transazioni e pagamenti periodici
- Piano d'accumulo
- Tasso di valuta e valuta per ogni conto
- Conti condivisi con più utenti

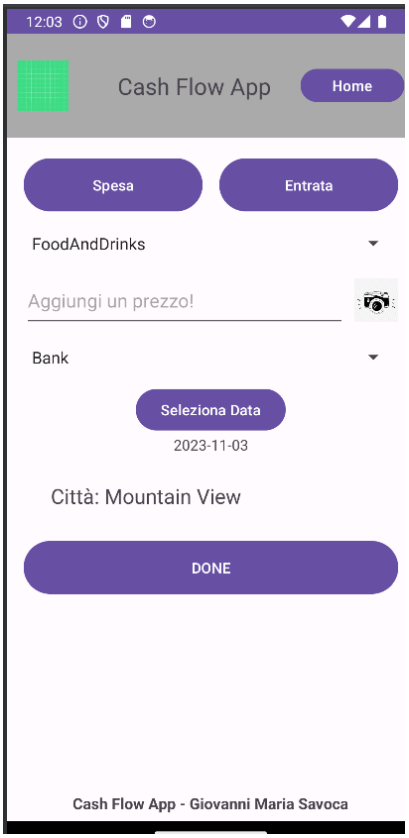
Ecco alcune ulteriori funzionalità che potrebbero essere implementate per arricchire il tuo progetto di applicazione mobile CashFlow:

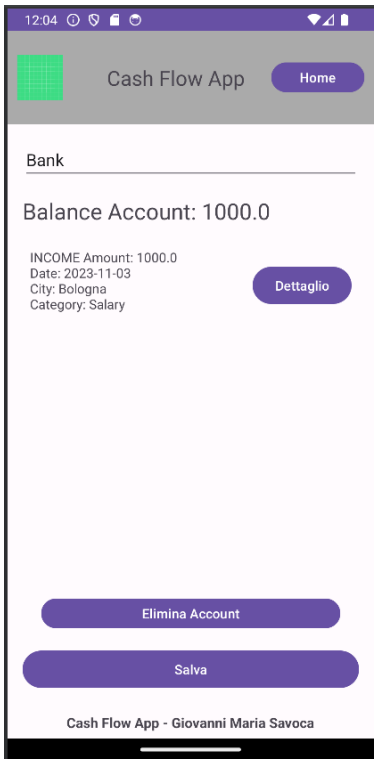
1. Database Online con Profili Utente: Implementa un sistema di autenticazione utente e un database online per consentire agli utenti di creare e gestire i propri profili. Ogni utente dovrebbe poter accedere ai propri dati finanziari in modo sicuro. Puoi utilizzare servizi come

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. Backup ed export dei dati: Aggiungere la possibilità di eseguire il backup dei dati e di esportare le transazioni in CSV, per scopi di archiviazione o condivisione.


12:03
Cash Flow App
Home
BANK
CASH
Totale:
2000.00
Nuovo Conto
Aggiungi Transazione
Statistiche
Cash Flow App - Giovanni Maria Savoca


12:03
Cash Flow App
Home
Spesa
Entrata
FoodAndDrinks
Aggiungi un prezzo!
Bank
Seleziona Data
2023-11-03
Città: Mountain View
DONE
Cash Flow App - Giovanni Maria Savoca


12:04
Cash Flow App
Home
Bank
Balance Account: 1000.0
INCOME Amount: 1000.0
Date: 2023-11-03
City: Bologna
Category: Salary
Dettaglio
Elimina Account
Salva
Cash Flow App - Giovanni Maria Savoca


12:04
Cash Flow App
Home
Seleziona Data Inizio
Seleziona Data Fine
28-10-2023
04-11-2023
Genera Grafico
2,000
1,600
1,200
800
400
0
29-10
30-10
31-10
01-11
02-11
03-11
04-11
Cash Flow App - Giovanni Maria Savoca

EditTransactionFragment

- + expenseButton : Button
- + incomeButton : Button
- + categorySpinner : Spinner
- + numberEditText : EditText
- + cameraButton : ImageView
- + ocrManager : OCRManager
- + REQUEST_IMAGE_PICK : integer
- + accountSpinner : Spinner
- + dateButton : Button
- + selectedTimeTextView : TextView
- + locationEditText : Calendar
- + doneButton : Button
- + deleteButton : Button
- + transactionOriginal : Transactions
- + accountOriginal : Account
- + categories : string [*]
- + jsonReadWrite : JsonReadWrite
- + accounts : string [*]
- + originalTransactionIndex : integer
- + originalAccountIndex : integer

- +EditTransactionFragment(transaction: Transactions, account: Account)
- +onCreateView(inflater: LayoutInflater, container: ViewGroup, savedInstanceState: Bundle): View
- + onActivityResult(requestCode: int, resultCode: int, data: Intent)
- +openCamera()
- +openGallery()
- +setExpense()
- +setIncome()
- +deleteTransaction()
- +updateTransaction()
- + showDatePickerDialog()

NewTransactionFragment

- + expenseButton : Button
- + incomeButton : Button
- + doneButton : Button
- + categorySpinner : Spinner
- + numberEditText : EditText
- + accountSpinner : Spinner
- + selectedDate : Calendar
- + isDateSelected : boolean
- + dateButton : Button
- + locationEditText : EditText
- + jsonReadWrite : JsonReadWrite
- + accounts : Account [*]
- + categories : string [*]
- + cameraButton : ImageView
- + selectedTimeTextView : TextView
- + cityPosition : City
- + ocrManager : OCRManager
- + REQUEST_IMAGE_PICK : integer

- +NewTransactionFragment(accounts: ArrayList<Account>, cityPosition: City)
- +onCreateView(inflater: LayoutInflater, container: ViewGroup, savedInstanceState: Bundle): View
- +onCreateView(inflater: LayoutInflater, container: ViewGroup, savedInstanceState: Bundle): View
- +openCamera()
- +openGallery()
- +saveTransaction()
- +showDatePickerDialog()

AccountDetailsFragment

- + account : Account
- + jsonReadWrite : JsonReadWrite
- + nameEditText : EditText
- + balanceTextView : TextView
- + transactionsRecyclerView : RecyclerView
- + saveButton : Button
- + deleteButton : Button

- +AccountDetailsFragment(account: Account)
- +onCreateView(inflater: LayoutInflater, container: ViewGroup, savedInstanceState: Bundle): View
- +changeName()
- +doesAccountExist(accounts: ArrayList<Account>, name: String): boolean
- +findAccountIndex(accounts: ArrayList<Account>, oldName: String): int
- +showDeleteConfirmationDialog()
- +deleteAccount() |
- +TransactionListAdapter

NewAccountFragment

- + jsonReadWrite : JsonReadWrite
- + editName : EditText
- + accounts : string [*]

- +NewAccountFragment(accounts: ArrayList<Account>)
- +onCreateView(inflater: LayoutInflater, container: ViewGroup, savedInstanceState: Bundle): View

HomeFragment

- + accounts : string [*]
- + subtotalText : string
- + myTextView : TextView
- + posizione : Posizione
- + city : City

- +HomeFragment(accounts: ArrayList<Account>)
- +onCreateView(inflater: LayoutInflater, container: ViewGroup, savedInstanceState: Bundle): View
- +onResume(): void
- +getSubtotal(): String
- +openNewAccountFragment(): void
- +openNewAccountFragment(): void
- +openNewAccountFragment(): void

MainActivity

- + btnHome : Button
- + accounts : Account [*]
- + jsonReadWrite : JsonReadWrite
- + PERMISSION_REQUEST_CODE : integer

- +onCreate(savedInstanceState: Bundle): void
- +onRequestPermissionsResult(requestCode: int, permissions: String[], grantResults: int[]): v...
- +loadFragment(fragment: Fragment): void

StatisticsFragment

- + accounts : Account [*]
- + btnLineChart : Button
- + google_maps : Button
- + incomeButton : Button
- + expenseButton : Button

- +StatisticsFragment(accounts: ArrayList<Account>)
- +onCreateView(inflater: LayoutInflater, container: ViewGroup, savedInstanceState: Bundle): Vi...

MapFragment

- + mMap : GoogleMap
- + mapFragment : SupportMapFragment
- + markers : string [*]
- + accounts : Account [*]

- +MapFragment(accounts: ArrayList<Account>)
- +onMapReady(googleMap: GoogleMap)
- +onCreateView(inflater: LayoutInflater, container: ViewGroup, savedInstanceState: Bundle): Vi...
- +getMarkerData(): ArrayList<MarkerOptions>
- + getCitiesFromAccounts(): ArrayList<MarkerOptions>

Line_chart

- + accounts : string [*]
- + startDate : Calendar
- + endDate : Calendar
- + startDateSelected : boolean
- + endDateSelected : boolean
- + openStartDatePickerButton : Button
- + openEndDatePickerButton : Button
- + generateChartButton : Button
- + startDateTextView : TextView
- + endDateTextView : TextView
- + lineChart : LineChart

- +Line_chart(accounts: ArrayList<Account>)
- + onCreateView(inflater: LayoutInflater, container: ViewGroup, savedInstanceState: Bundle): View
- +openStartDatePicker()
- +openEndDatePicker()
- +createLineChart()
- +generateDataEntries(startDate: Calendar, endDate: Calendar): ArrayList<Entry>
- +isSameDay(date1: Calendar, date2: Calendar): boolean
- +formatDateKey(date: Calendar): String
- +formatDateKeyWithoutYear(date: Calendar): String

Income_expense

- + isIncome : boolean
- + accounts : string [*]
- + accountsCheckBox : CheckBox
- + accountsListView : string
- + selectedAccounts : Account [*]
- + pieChart : PieChart
- + barChart : BarChart
- + title : TextView

- +Income_expense(isIncome: Boolean, accounts: ArrayList<Account>)
- +Data Type
- + updateSelectedAccounts()
- +initPieChart(selectedAccounts: ArrayList<Account>)
- +initBarChart(selectedAccounts: ArrayList<Account>)
- +getIncomeOrExpensePieData(accounts: ArrayList<Account>): List<PieEntry>
- +getIncomeOrExpenseBarData(accounts: ArrayList<Account>): List<BarEntry>
- + getCategoryColors(): int[]

City

- + nameCity : string
- + latitude : double
- + longitude : double

- +City()
- +City(nameCity, latitude, longitude)
- +getNameCity()
- +setNameCity(nameCity)
- +getLatitude()
- +setLatitude(latitude)
- +setLatitude(latitude)
- +setLatitude(latitude)
- +toString()

CategoriesEnum

- +Other
- +FoodAndDrinks
- +Shopping
- +House
- +Transport
- +LifeAndEntertainment
- +CommunicationAndPC
- +Salary
- +Gifts

Transactions

- + income : boolean
- + amount : double
- + date : Calendar
- + city : City
- + category : <no type>

- +Transactions()
- +Transactions(income, amount, date, city, catego...
- +getCategory(): CategoriesEnum
- +isIncome(): Boolean
- +getAmount(): double
- + getAmountValue(): double
- + getDate(): Calendar
- + getCity(): City
- +toString(): String
- + printOnApp(): String

Account

- + name : string
- + balance : double
- + listTrans : Transactions [*]

- +Account()
- +Account(name: String)
- +Account(name: String, listTrans: ArrayList<Transactions>)
- + getName(): String
- + setName(name: String): void
- +getBalance(): double
- + getListTrans(): ArrayList<Transactions>
- +updateBalance(): void
- +addTransaction(transaction: Transactions): boolean
- + removeTransaction(transaction: Transactions): boolean
- +editTransaction(transactionOriginal: Transactions, newTrans: Transactions): boolean
- + toString(): String

Posizione

- + fusedLocationProviderClient : FusedLocationProviderClient
- + context : Context

- +Posizione(context: Context)
- +requestDeviceLocation(callback: DeviceLocationCallback): void
- +interface DeviceLocationCallback
- +onLocationFetched(city: City): void
- +onLocationFetchFailed(e: Exception): void

OCRManager

- + context : Context

- +OCRManager(context: Context)
- +OCRManager(context: Context)
- +interface OCRListener
- +onTextRecognized(text: String): void
- +onFailure(e: Exception): void

Test

- + accounts : Account [*]

- +Test()
- +getList(): ArrayList<Account>

JsonReadWrite

- + accounts : string [*]
- + fileName : string

- +JsonReadWrite(fileName: String)
- +JsonReadWrite(accounts: ArrayList<Account>, fileName: String)
- +setList(accounts: ArrayList<Account>, context: Context): void
- +setList(accounts: ArrayList<Account>, context: Context): void
- +readAccountsFromJson(context: Context): ArrayList<Account>
- +readJsonFromFile(context: Context): String