Ashborn Bank

Relazione Progetto Programmazione Dispositivi Mobili

Marius Dumitru Berinde, Giorgio Perna

October 3, 2024

Contents

1 Gruppo

Informazioni del gruppo di laboratorio

Nome del gruppo Full Metal Dev Componenti

- Marius Dumitru Berinde
- Giorgio Perna

2 Date

- **Data di sottomissione proposta progetto: 15/05/2024**
- **Data di accettazione proposta progetto: 15/05/2024**

3 Descrizione

Intendiamo sviluppare un applicazione di internet banking in cui sono presenti le funzionalità elencate nella tabella obbiettivi, lo stile dell'applicazione è simile a quello dell'app bancaria di Intesa San Paolo da cui abbiamo preso ispirazione, per la sua semplicità di utilizzo.

Vedi la tabella obbiettivi per ulteriori dettagli.

4 Definizioni

Termine	**Definizione**
IBAN	International Bank Account Number
App	applicazione
info	informazioni
MAV	Mediante Avviso
MVVM	Model View View Model
PIN	Personal Identification Number
qr code	Quick Response Code

Table 1: Tabella Definizioni

5 Contesto del progetto

5.1 Situazione attuale

Negli store attuali ogni banca ha almeno un'applicazione dedicata (quindi sarebbe inutile provare ad elencarle). Abbiamo guardato principalmente le app di Intesa SanPaolo, e Fineco, per via della loro semplicità di utilizzo.

Durante la fase esplorativa ci siamo resi conto che ci sono alcune operazioni sono scomode e poco intuitive da trovare ed effettuare quando serve, in particolare senza andare a cercare su internet come effettuarle, e dove cercare le voci corrispondenti all'interno dell'app, Eg. il disconoscimento di un'operazione sebbene fattibile, richiede spesso di chiamare l'assistenza, e di cercare su internet come effettuarla (perché molto nascosta e poco chiara all'interno dell'app), dovrebbe quindi essere più semplice da trovare e la procedura più guidata.

5.2 Benefici e creazione di valore

Vogliamo migliorare l'esperienza degli utenti di un'app bancaria semplificando l'utilizzo alcune funzionalità, replicando la semplicità di utilizzo, che altre app già presenti sul mercato offrono. (vedi obiettivi del progetto).

5.3 Obiettivi del progetto

Obiettivo	Descrizione
Operazione di disconoscimento di un'operazione (molto semplificata)	Per farlo introdurremo un pulsante extra che consente di effettuare direttamente l'operazione, aprendo una schermata dedicata, senza dover navigare in molteplici voci di menù.
Accesso tramite codice numerico	Gli accessi successivi al primo vengono fatti tramite codice PIN.
Bonifico, Mav	Possibilità di eseguire le operazioni descritte.
Gestione avvisi	Possibilità di eseguire le operazioni descritte.

Table 2: Tabella Obiettivi

6 Profilo del progetto

6.1 Analisi requisiti

Requisiti funzionali

- Per registrarsi la prima volta l'utente deve fornire Nome, Cognome, Data di nascita e Codice cliente, che verranno memorizzati in un database locale per poterli riusare.
- Ad ogni login viene richiesto il PIN per poter accedere (si assume che venga fornito in filiale).
- L'app impedisce di interagire senza la connessione ad internet (neanche di fare il login).
- L'app richiede di inserire nuovamente il PIN qualora non si interagisca con l'app per una quantità di tempo sopra una soglia stabilita, o se viene messa in background per più di una certa quantità di tempo.
- L'app consente di effettuare al suo interno le seguenti operazioni bancarie: pagamento di un bonifico, pagamento di un MAV
- il pagamento di un MAV deve poter essere gestito con la lettura del qr code dalla fotocamera, o permettere di inserire i dati manualmente

Requisiti non funzionali

- l'app deve essere responsiva e performante
- l'app deve essere intuitiva e semplice da usare

6.2 Profilo della soluzione

Le seguenti schermate servono per mostrare possibili interazioni dell'utente con l'app. Esistono due versioni del login:

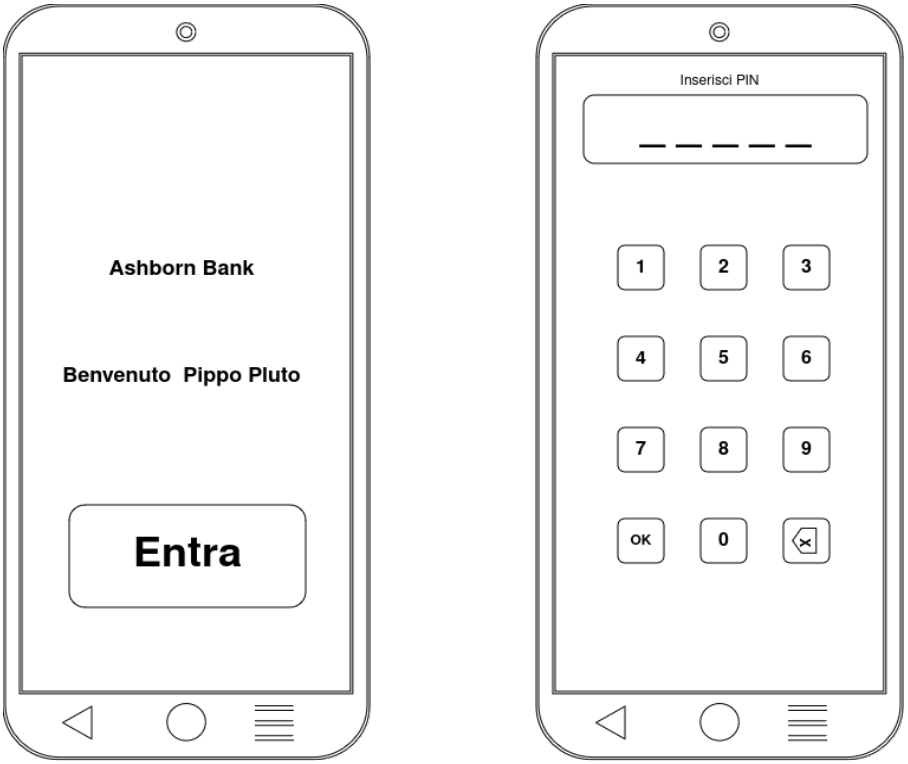

Figure 1: Login utente conosciuto

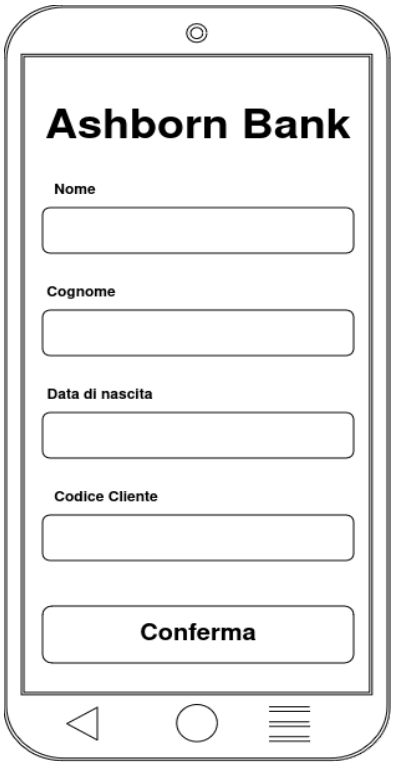

Figure 2: Schermata che l'utente deve compilare per il primo login

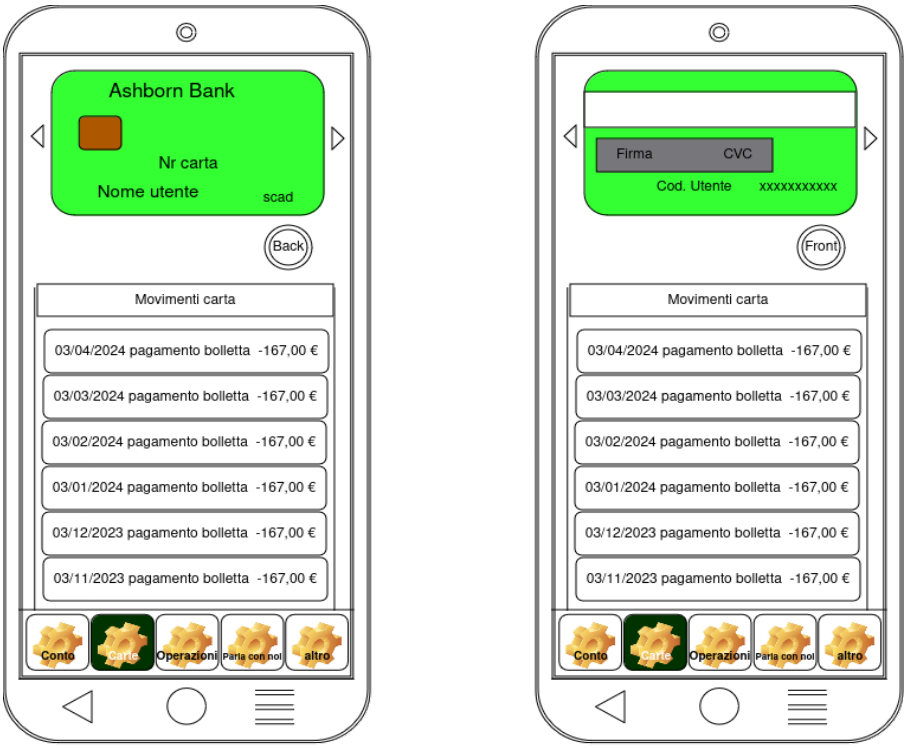

Figure 3: Dettagli conto corrente

Figure 4: Dettagli Conto Corrente
Premendo l'icona dell'occhio si mostra/nasconde il saldo corrente.
Premendo l'icona share l'IBAN viene copiata nella clipboard del sistema, o mandata tramite le app disponibili

L'interfaccia grafica è creata usando jetpack compose mentre useremo il paradigma MVVM per la suddivisione delle responsabilità di gestione dei dati e delle interazioni con questi ultimi.

7 Vincoli e assunti

- L'utente ha già creato almeno un conto bancario andando di persona in una delle filiali della banca, e possiede quindi già un codice cliente. Deve solo attivare il suo account (non deve crearne uno nuovo).ù
- L'app funziona solo se l'utente è connesso a internet e consente di usare un solo dispositivo per volta.
- Per l'operazione di Mav leggiamo un qr code creato da noi, altrimenti facciamo immettere i dati necessari manualmente all'utente.
- Gestiamo solo conti personali (senza conti aziendali)
- I pagamenti sono fittizi, (quelli con la carta sono simulati da un servizio)
- La cancellazione dell'operazione implica il rimborso della somma dell'operazione
- Non gestiamo carte prepagate

7.1 Vincoli temporali

- **7 Settembre** Sessione d'esame in cui avverrà la discussione del progetto

7.2 Vincoli tecnologici

- Supportiamo l'applicazione a partire da Android 10+ da api 29 a 35.
- Sviluppo del codice fatto interamente in Kotlin.
- Uso di JetPack Compose per realizzare l'interfaccia utente.
- Uso di Coroutines e Live Data, e Flow per il recupero delle informazioni da mostrare all'utente.
- Uso di Room per interagire con il database locale.
- Non supportiamo IOS (non abbiamo un dispositivo Apple)
- Il codice deve rispettare il pattern MVVM

8 Design

Abbiamo suddiviso i file del sistema nelle seguenti cartelle

- **dao**: contiene il dao che si occupa di interagire direttamente con il database
- **data**: contiene tutte le data class che usiamo come formato di dati per interagire per le operazioni in stile crud con il database.
- **database**: contiene i file per la gestione del database vero e proprio tranne i file di asset usati per inizializzarlo
- **model**: contiene il datastore manager, che si occupa di gestire la lettura e scrittura delle preferenze in locale
- **repository**: contiene i repository che i view model usano per interagire con il database sono divisi in modo tematico: per i conti `ContiRepository`, per le carte `CardRepository`, per gli avvisi `AvvisiRepository`, per le operazioni `OperationRepository`, per gli utenti `OfflineUserRepository`.
- **view**: contiene tutte schermate dell'app.
- **viewModel**: contiene i view model delle view.

in più per raggruppare i file per tipo di scopo in view e viewModel abbiamo creato le seguenti cartelle:

- **login**: sono i file necessari per fare login
- **operazioni**: contiene i file necessari per gestire le varie operazioni

8.1 Sezioni app

8.1.1 Autenticazione

L'utente attiva localmente il proprio account(i dati vengono forniti dalla banca), mediante `Registrazione` , questa schermata si connette al database mediante l'uso del suo view-model, verificando la validità delle informazioni fornite dall'utente, e memorizzando parte di queste in locale. Per farlo usa il repository `OfflineUserRepository` e il dao `AshbornDao` , mentre per la scrittura in locale utilizza la classe `DataStoreManager` che sfrutta la tecnologia dei Datastore. I dati richiesti per completare la procedura di attivazione dell'account sono i seguenti: data di nascita, nome, cognome, codice cliente. In caso di errore fornisce indicazioni su cosa si è sbagliato, in caso di registrazione con successo, effettua la navigazione alla pagina di benvenuto.

8.2 Gestione dei conti

All'interno delle pagine principali, Il tab Conti è quello che consente di vedere le operazioni svolte con il conto attuale, mostrando la data di esecuzione, il motivo del pagamento e l'importo della transazione. Le operazioni vengono ricaricate ogni volta che si naviga verso questo tab oppure vengono modificate anche in locale oltre che in remoto in caso di revoca di una di queste. Permette di ordinare le transazioni per data, importo, o descrizione, inoltre cliccando sopra la transazione, si ha la possibilità di navigare verso la schermata di dettagli dell'operazione, dove è possibile vedere meglio questi ultimi, ed è possibile inoltre: condividere l'operazione, e revocarla ove possibile, o effettuare una richiesta di disconoscimento qualora questa sia frutto di frode, se non è possibile revocarla perché è già eseguita. Consente di vedere i conti disponibili tra quelli posseduti, e di cambiare, il conto mostrato se se ne possiede più di uno; mostra che non se ne possiedono al momento, qualora non se ne abbia nessuno al momento. Consente di vedere i dati del conto mostrato incluso il saldo e di nasconderlo/mostrarlo premendo sull'apposita icona.

8.3 Gestione delle carte

All'interno delle pagine principali, Il tab Carte è quello che consente di vedere le operazioni svolte con la carta attuale, mostrando la data di esecuzione, il motivo del pagamento e l'importo della transazione. Permette di ordinare le transazioni per data, importo, o descrizione, inoltre cliccando sopra la transazione, si ha la possibilità di navigare verso la schermata di dettagli dell'operazione, dove è possibile vedere meglio questi ultimi, ed è possibile inoltre: condividere l'operazione e effettuare una richiesta di disconoscimento qualora il movimento sia frutto di frode. Consente di vedere le carte disponibili tra quelle possedute, e di cambiare, la carta mostrata se se ne possiede più di una; mostra che non se ne possiedono al momento, qualora non se ne abbia nessuna. Consente di vedere i dati della carta mostrata.

8.4 Operazioni

All'interno delle pagine principali, il tab operazioni è quello che consente di effettuare operazioni bancarie, al momento quelle supportate sono effettuare un bonifico, e pagare un mav.

8.4.1 Pagamento MAV

Il pagamento di un MAV è una delle operazioni supportate, può essere eseguito in due modi: compilando i campi a mano, o scannerizzando un codice QR (generato appositamente da noi), la prima schermata consente di scegliere la modalità. L'inserimento manuale viene gestito da `MavManuale`. Una volta letto il codice QR il risultato viene serializzato e mandato tramite la navigazione alla scermata del MAV manuale, dove l'utente può modificare da quale conto intende fare il pagamento se ne ha più di uno. Una volta confermato il MAV i campi vengono validati, e in caso di errori viene segnalato il problema, altrimenti l'operazione viene serializzata e mandata alla schermata di riepilogo, che la prende come parametro dopo che questa viene deserializzata. Se nel riepilogo l'utente è soddisfatto, può richiedere la conferma del pagamento, a seguito del quale viene richiesto il PIN tramite la schermata apposita, a cui l'operazione viene mandata in forma serializzata.

8.4.2 Bonifico

Il Bonifico è una delle operazioni supportate, che può essere effettuato con le seguenti modalità: normale e istantaneo. Entrambe richiedono all'utente di inserire negli appositi campi, le seguenti informazioni: beneficiario, iban ,importo, causale, data di esecuzione, e scelta del conto corrente da cui fare l'operazione. La versione istantanea differisce da quella normale per le seguenti caratteristiche: la data di esecuzione deve essere quella odierna, il costo di commissione applicato è più elevato, inoltre non è possibile revocare l'operazione, in quanto viene effettuata immediatamente invece di essere segnata come pendente . Un'operazione segnata come pendente (ad esempio il bonifico normale) verrà poi effettuata successivamente da un apposito servizio (che simula il

back-end), che si occuperà di completarla dal lato del ricevente, mentre dal lato dell'esecutore verrà detratto l'importo dell'operazione immediatamente dalle risorse disponibili, che potrà poi venire restituito in caso di revoca dell'operazione. Per memorizzare l'utente attivo sull'app usiamo un datastore manager che memorizza localmente nome,cognome e codice cliente, inoltre memorizziamo il numero di tentativi sbagliati del pin e il tempo rimasto del timer, che si attiva quando il numero di tentavi errati è maggiore di 3, per gestire i tentavi errati.

8.5 Parla con noi

Contiene i dati per contattare la banca.

8.6 Altro

All'interno delle pagine principali, il tab altro è quello che consente di visualizzare le comunicazioni della banca, permette di andare alla pagina che contiene una versione espansa e più facilmente leggibile

8.6.1 Dettagli avvisi

Contiene la versione espansa di un avviso, e consente di condividerlo con il pulsante di share o di eliminare l'avviso in questione.

9 Implementazione

9.1 Autenticazione

9.1.1 Welcome

La prima schermata visualizzata dall'utente, contiene un pulsante e una scritta con il nome dell'utente preso tramite le preferenze locali da datastoremanager usando il viewmodel. Se le preferenze non sono impostate il pulsante porta alla schermata di registrazione altrimenti a quella di richiesta del PIN. Infine abbiamo usato un LaunchedEffect per gestire la richiesta del permesso nelle notifiche.

9.1.2 Richiesta PIN

Sono presenti due casi d'uso: l'autenticazione durante il login e l'approvazione di un'operazione bancaria (nel secondo caso questa viene deserializzata durante la navigazione e passata per parametro). Per essere un PIN legittimo è richiesto che sia composto da 8 cifre, e che il suo hash nel database corrisponda. In caso di errore indica qual è il problema. La gestione degli errori si attiva quando si commetto più di due errori consecutivi facendo fare attesa attiva dell'utente e si resetta quando si introduce il PIN corretto. L'attesa aumenta in modo proporzionale al numero di tentativi fatti e funziona anche se si spegne e riaccende l'applicazione (questa cosa è possibile perché il numero di tentativi è memorizzato come preferenza locale) inoltre se si prova a mettere in background o a uscire dall'app il timer viene fatto ripartire. Abbiamo usato i LauchedEffect per gestire eventi di navigazione in `AskPIN` bastati sul parametro .

9.1.3 Registrazione

L'utente riempie i campi presenti, e tramite la classe `Validatore` ne viene controllata la correttezza del formato, in paritcolar modo devono soddisfare i seguenti vincoli:

- data nascita: i controlli vengono fatti mediante il CustomDatePickerDialog,
- nome e cognome: verifica che il campo abbia lunghezza inserita compresa tra 2 e 20 e che sia alfanumerico
- codice cliente: verifica che il campo sia alfanumerico di 9 caratteri

Inoltre verifica che l'utente sia un cliente della banca usando la funzione **auth** e in caso positivo memorizza i campi nome,cognome e codice cliente come preferenza locale usando il DatastoreManager. Abbiamo usato un LaunchedEffect per gestiere la scrittura delle preferenze locali Per prendere lo stato di navigazione alla richiesta di autenticazione fatta tramite una coroutine usiamo un Livedata.

9.2 Gestione Conti

I conti selezionabili vengono gestiti mediante il composable **Conti** mentre lo scaricamento delle operazioni viene affidato a **ContiViewModel** e **DettagliOperazione**. Quest'ultima dà la possibilità di condividere e cancellare o disconoscere l'operazione.

9.3 Gestione Carte

Le carte selezionabili vengono gestiti mediante il composable **Carte** mentre lo scaricamento delle operazioni viene affidato a **CarteViewModel** mentre dettagliOperazione serve per gestire l'operazione.

9.4 Operzioni

9.4.1 Bonifico

L'utente riempie i campi presenti, e tramite il view model apposito controlla la correttezza dei campi inseriti , in paritcolar modo devono soddisfare i seguenti vincoli, che vengono controllati tramite la funzione **gestisciErroriBonifico**:

- data : i controlli vengono fatti mediante il CustomDatePickerDialog
- beneficiario: il campo deve essere alfanumerico con lunghezza compresa tra 2 e 100 caratteri
- IBAN : verifica che il campo abbia un lunghezza compresa tra 17 e 34 caratteri e usa l'algoritmo di controllo mod 97 per il controllo della plausibilità
- importo : verifica che il campo sia minore di 15000.0 e che abbia al massimo 2 cifre decimali
- causale : controlla che campo abbia lunghezza minore di 300 caratteri

I bonifici non istantanei (e i MAV che hanno come data di esecuzione una data futura), vengono memorizzati nel database con lo stato PENDING. Il giorno dell'esecuzione verranno completati dal servizio **DbWorker**

9.4.2 MAV

L'utente riempie i campi presenti, e tramite il view model apposito controlla la correttezza dei campi inseriti , in paritcolar modo devono soddisfare i seguenti vincoli:

- data : i controlli vengono fatti mediante il CustomDatePickerDialog,

Nel caso della lettura del QR code, questa viene fatta usando la libreria **libs.google.code.scanner** tramite la classe `BarCodeScanner` nel viewmodel, che viene acceduto tramite i composable `MavQRCode e ScanBarcode`

9.4.3 QR code

Il formato che abbiamo usato per il QR code che leggiamo è il seguente: `iban | descrizione | importo in centesimi`. Una volta letto il codice QR il risultato viene convertito in un'operazione, e serializzato per essere mandato come parametro di navigazione alla schermata di mav manuale, che riempirà in automatico i campi necessari.

NOTA BENE
Per via del supporto alle API questa feature è supportata a partire da android 13, dalla 10 alla 12 è disponibile soltanto la versione ad inserimento manuale.

9.5 Altro

Questa sezione viene usata per vedere i dettagli dell'utente attuale,vedere gli avvisi della banca e dare la possibilità di cambiare mediante il pulsante "Elimina Smartphone"

9.6 CustomDatePickerDialog

Le date vengono gestite mediante il composable `CustomDatePickerDialog`, che implementa dei controlli sulle date utilizzabili mediante il parametro useCase, questo evita di dover controllare la validità dell'input, perché non è possibile che l'utente inserisca un valore di data non valido per il caso d'uso, e aiuta l'utente a sapere quali date sono disponibili. In particolare gestiamo i seguenti casi:

- **NASCITA**: fa in modo che possano essere selezionate solo le date di nascita dei maggiorenni rispetto alla data attuale.
- **MAV**: una data è valida se appartiene al range [data odierna , data odierna+ 18 mesi] escludendo i sabati e le domeniche.
- **BONIFICO**: una data è valida se appartiene al range data odierna , [data odierna + 12 mesi] escludendo i sabati e le domeniche.

9.7 Navigazione

La navigazione dell'app viene fatta mediante `AppNavigazione2` e il seguente grafico mostra come è strutturata. La route conti fa navigare verso `Pagine` dove è possibile scegliere quali schermate vedere. Da questa schermata possono essere caricati direttamente i composable di Conti, Carte, Operazioni, ParlaConNoi e Altro ed è la schermata principale dell'app. Le routes login e PIN usano entrambe parametri di navigazione. In particolare login ha un parametro opzionale che viene usato per la gestione nel rientro dal background mentre pin ha un parametro obbligatorio (per la route non per la schermata) che viene usando quando si compie un bonifico o un MAV, questo parametro è sempre deserializzato nella navigazione prima di essere passato al composable AskPIN. La serializzazione/deserializzazione è effettuata trasformando il parametro in una stringa in formato JSON al fine di mandare dati (di solito un'operazione) tra le varie schermate. Il parametro in questione viene usato per fare in modo che la schermata appaia in modo diverso in base al valore assegnato al parametro. Le altre routes che hanno come parametro operation sono:

- dettagli-operazione
- mav-manuale
- riepilogo-operazione

Una nozione onorevole va alla route `mav-manuale` infatti il suo parametro è sia opzionale che serializzato

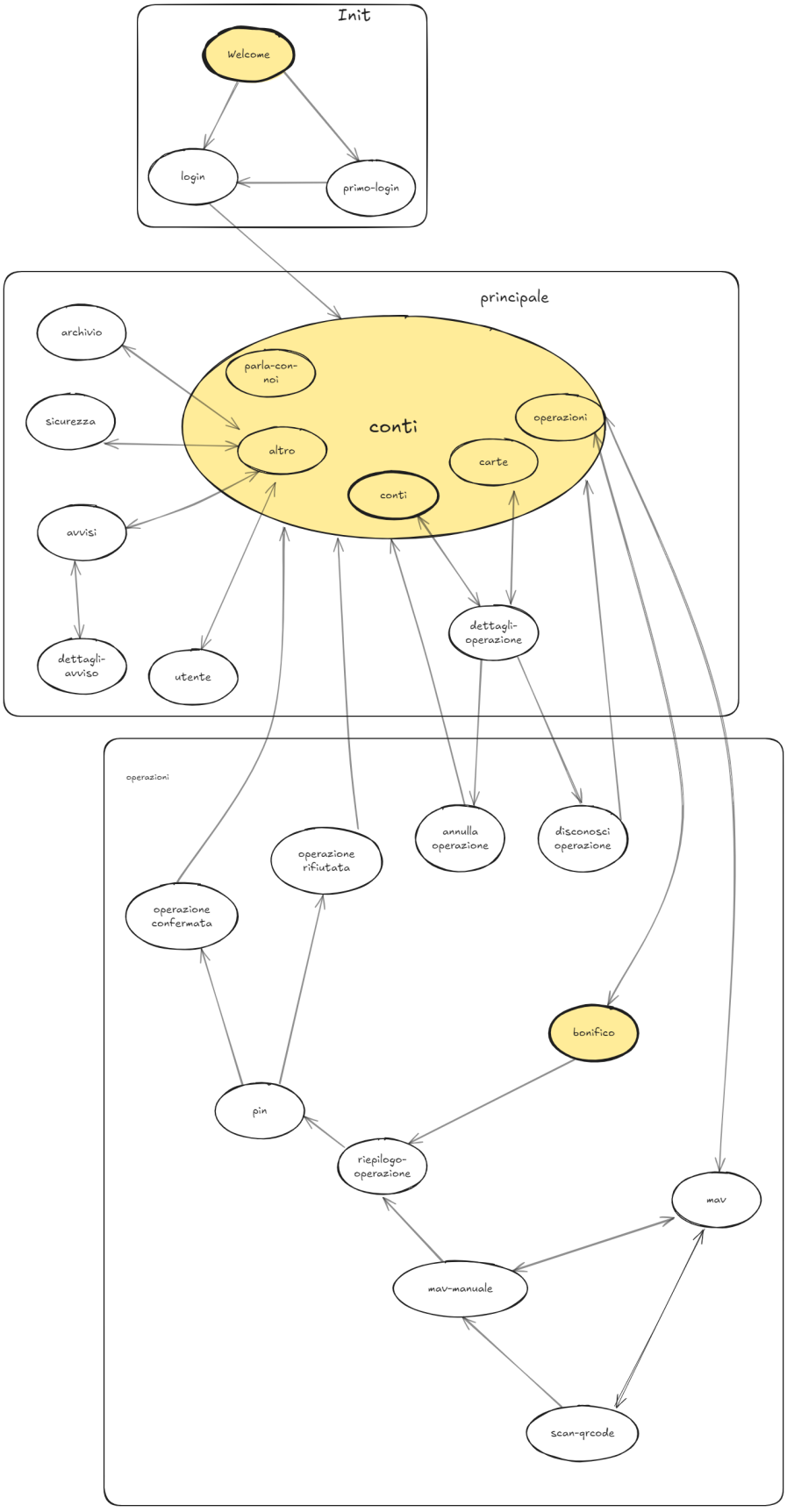

Figure 5: Albero di navigazione
I nodi in giallo rappresentano la destinazione di partenza dell'albero

9.8 Organizzazione del database

Il database è organizzato nelle seguenti tabelle:

- **users**: gestione dei dati degli utenti
- **operations**: gestione dei dati di tutte le operazioni bancarie
- **conti**: gestione dei conti degli utenti
- **carte**: gestione delle carte degli utenti associate ad un conto
- **avvisi** : gestione degli avvisi

Le date vengono memorizzate sotto forma di stringa come timestamp, che va poi convertita usando LocalDateTime, per renderla usabile. Per creare la comunicazione tra database (rappresentao dal dao) con i vari viewModels usiamo i flow.

9.9 Connessione ad internet

Per il funzionamento dell'app richiediamo di avere una connessione ad internet funzionante, (questo non è un requisito in sé per sé per il funzionamento dell'app, come l'abbiamo implementata noi, ma per come dovrebbe funzionare una vera app bancaria, a cui serve per contattare il backend, e senza internet non sarebbe normalmente possibile svolgere nessuna operazione in remoto). Per fare ciò abbiamo usato la classe `ConnectivityObserver` che si occupa di controllare che lo stato della connessione sia disponibile, e sia quindi possibile procedere, con le operazioni. Per impedire all'utente di interagire con l'app quando la connessione non è disponibile abbiamo usato il composable `ErroreConnessione` che si occupa di mettere un overlay, che mostra il messaggio di perdita della connessione, e impedisce di interagire con il resto dell'app.

9.10 Servizi in background

Per simulare il lavoro fatto sul backend usiamo dei servizi, implementati tramite delle classi che estendono `CoroutineWorker`:

- Per concludere le transazioni che si verificano verso clienti della stessa banca che non sono istantanee, (per quelli che non ne fanno parte non simuliamo l'altro lato della transazione) abbiamo implementato il servizio `DBWorker`.
- Per simulare l'utilizzo della carta di credito da parte degli utenti e mostrare la notifica dell'uso delle carte abbiamo implementato il servizio `PaymentWorker` che crea delle operazioni casuali, e le fa compiere alle carte, se l'app ha il permesso per effettuare notifiche push, viene inviata una notifica all'utente, con i dettagli dell'operazione di pagamento.

Entrambi eseguono, periodicamente ogni circa 20 minuti (potrebbe richiedere un po' più di tempo, se il sistema operativo ha bisogno di recuperare memoria, ma il compito verrà comunque portato a termine).

9.11 Share

Per implementare questa feature abbiamo creato il compose `Share` che attraverso un intent invia il contenuto trasformato in stringa e consente di memorizzarlo nella clipboard del sistema o di inviarlo usando una delle app installate disponibili adatte.

9.12 Altre funzionalità

Come richiesto dallo standard bancario abbiamo bloccato la possibilità di compiere screenshot quando l'applicazione è in foreground. Questa operazione viene fatta introducendo le seguenti istruzioni nella main activity:

```
window.setFlags(
    WindowManager.LayoutParams.FLAG_SECURE,
    WindowManager.LayoutParams.FLAG_SECURE
)
```

inoltre per uniformarci alle altre app bancarie abbiamo bloccato l'orientamento in modalità verticale inserendo il seguente settaggio nel manifest

```
android:screenOrientation="sensorPortrait"
```

Se l'utente non usa l'app per 5 minuti fa ritornare l'utente in Welcome o se rimane per lo stesso periodo in background. Nella quasi totalità dei composable abbiamo usato le String resources per i nomi dei campi.

9.13 Test

I test sono stati condotti su diversi livelli, per le funzioni logiche (principalmente quando si tratta di validare un input) abbiamo creato la classe `OperationViewModelTest` dove abbiamo applicato test driven development. I beta-test sono stati effettuati su diversi tipi di utenti con diverse fasce di età (in totale 8 persone esterne e 2 sviluppatori), che hanno fatto emergere delle criticità di user experience, che abbiamo successivamente usato come feedback per migliorare l'app. I test dei beta-tester si sono concentrati principalmente sui dispositivi Pixel 8, Pixel 7 e Medium phone con le configurazioni elencate di seguito. Durante questi test abbiamo fatto provare a ciascun utente le fasi di registrazione, effettuazione di un bonifico, e pagamento di un MAV (sia con la lettura di un qr code che inserendo manualmente i dati), revoca di un bonifico, e cancellazione di un avviso, facendo durare ogni test 5-10 minuti. Le view che hanno riscontrato maggiori problemi di usabilità sono state: Registrazione, MavManuale e Bonifico. Questo era dovuto alla poca familiarità degli utenti con questo genere di app, e al modo in cui venivano gestiti visivamente gli errori per l'utente. L'applicazione è stata testata su dispositivi virtuali con diverse versioni di Android , in particolare ci siamo accertati che funzioni correttamente con le versioni a partire da Android 13 in quanto queste richiedono il permesso per l'uso di notifiche push. Abbiamo testato l'app sui seguenti dispositivi emulati:

- Big phone AVD 35 Android 14/15
- Medium phone API 35 Android 14/15
- Pixel 8 phone API 35 Android 14/15
- Pixel 7 phone API 33 Android 13
- Pixel 6 API 31 Android 12
- Pixel 5 API 30 Android 11
- Pixel 4 API 29 Android 10
- Medium Tablet API 35 Android 14/15

Mentre durante lo sviluppo abbiamo usato come dispositivi target per l'app principalmente Medium Phone e Pixel 8, gli altri sono stati testati per assicurarsi che funzioni, ma l'interfaccia grafica è ottimizzata solo per i dispositivi target.

10 Conclusioni

Rispetto ai requisiti originali, abbiamo scelto di non implementare il pagamento RAV in quanto ci siamo resi conto che è un tipo particolare di codice di riconoscimento di immagini. Visto che non abbiamo un backend vero ma lo simuliamo tramite dei servizi, e un database room, abbiamo ridotto i dati dell'utente, che l'app memorizza, abbiamo scelto di non sviluppare la gestione dell'archivio, e per la stessa ragione non abbiamo implementato del tutto il disconoscimento di un'operazione visto che per fare quest'ultimo bisognerebbe caricare molti documenti e verificarne la validità, che sarebbe complicato fare per via dei forti limiti del database, e del fatto che non sono facilmente controllabili in modo automatico. Abbiamo raggiunto la maggior parte degli obbiettivi proposti tuttavia la lettura del QR code è supportata dalla versione 13 in poi, visto che Android 12 si sta avvicinando alla fine del suo ciclo di vita, mentre le versioni precedenti non sono più sopportate neanche da google, e che occorrerebbe usare una libreria completamente diversa, e ormai deprecata, solo per supportare delle versioni di Android non più mantenute, e si è preferito concentrarsi su quelli attualmente maggiormente in uso. La pagina seguente mostra l'aspetto di alcune schermate per avere un idea dell'aspetto complessivo dell'app.

Figure 6: Welcome

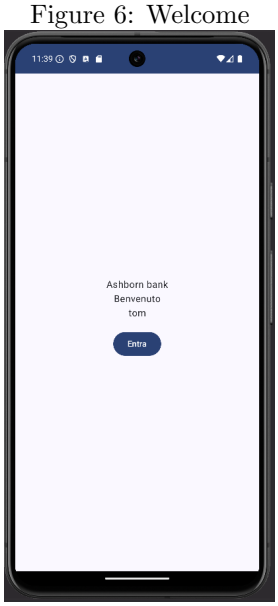

Figure 7: AskPIN

Figure 8: Registrazione

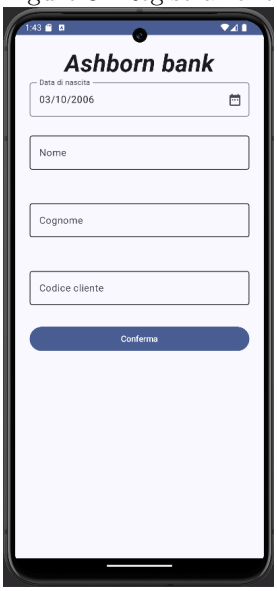

Figure 9: Carte

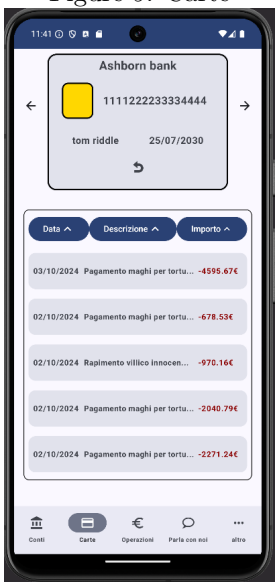

Figure 10: Conti

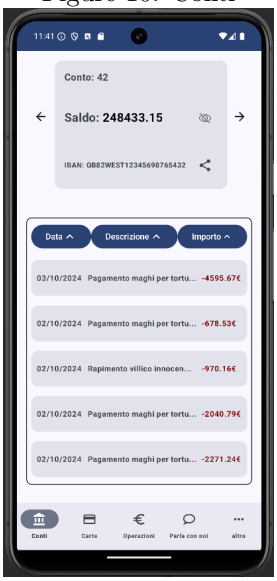

Figure 11: DettagliOperazione

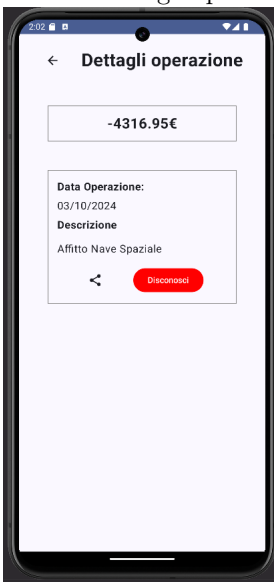

Figure 12: Altro

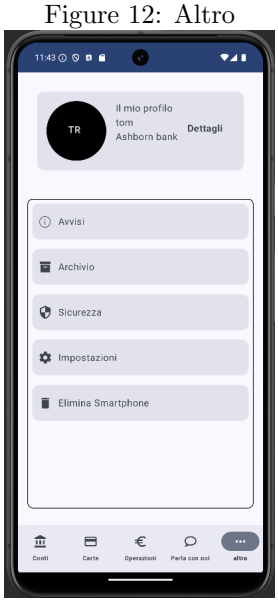

Figure 13: ParlaConNoi