Relazione Progetto

Alessandro Arrighi

July 2024

Laboratorio di Programmazione di Sistemi Mobili
Corso di Tecnologie dei Sistemi Informatici
Alma Mater Studiorum - Università di Bologna

Contents

1 Introduzione

Il presente progetto ha come scopo quello di realizzare una app nativa per Android, al fine di gestire lo storage di credenziali per servizi utente. Nello specifico l'utente ha la possibilità di effettuare le 4 operazioni CRUD (Create, Read, Update, Delete) per le proprie credenziali. Il sistema gestisce oltre all'archiviazione dei dati in locale, anche la connessione con un server esterno per il salvataggio ed il recupero delle credenziali da remoto.

2 Struttura del Progetto

L'architettura utilizzata per questo progetto segue i principi di progettazione della clean architecture. Sono stati divisi i sorgenti in due package principali: uno per la gestione dell'interfaccia grafica e l'altro per la gestione dei dati. La struttura architetturale di un progetto, che segue i principi della clean architecture, può utilizzare un terzo gruppo chiamato domain. Quest'ultimo gestisce classi per raggruppare le stesse funzionalità utilizzate da più ViewModel. Esso non è obbligatorio poiché dipende dalla complessità del progetto. In questo specifico caso non è stato utilizzato poiché andrebbe ad appesantire la sua struttura.

I dati sono stati gestiti creando sia le classi necessarie per l'archiviazione in locale sia quelle per la comunicazione con il server remoto. L'archiviazione in locale viene gestita utilizzando Android Room, che gestisce un database SQL Lite. La classe ServiceCredentialDAO è costituita da metodi per effettuare query al database, con le quali ricevere o inviare dati. L'archiviazione dei dati da remoto avviene utilizzando le api fornite dal server. La classe responsabile della comunicazione con il server istanzia una connessione attraverso la libreria Retrofit. Grazie alla libreria Moshi i dati sono convertiti da JSON a classe Kotlin. Nella fase successiva la connessione, è possibile creare l'interfaccia che fornisce i metodi per le chiamate api. Infine, è stata implementata la classe ServiceCredentialRepository. Il suo scopo è quello di fornire le api alla UI per l'accesso ai dati. In questo modo, la gestione dei dati locali e remoti è delegata al Repository. Così facendo le classi di UI hanno un utilizzo semplificato dei dati durante l'intero ciclo di vita del software.

Nella parte di UI, l'oggetto ViewModel viene utilizzato per recuperare i dati dal repository e rappresentarli nell'interfaccia attraverso le activity. Tutti i passaggi gestiti dal ViewModel sono effettuati all'interno di threads di background, in modo che l'utente possa utilizzare l'applicazione anche quando queste funzionalità sono in esecuzione. Vi sono presenti due activity: una è la MainActivity, che dispone i dati a video utilizzando una RecilcerView. La seconda activity permette di inserire o modificare le credenziali. Questa viene chiamata al click di un bottone presente nella MainActivity e i dati vengono scambiati tra loro utilizzando un oggetto Intent.

3 Punti di Forza

Le scelte architetturali del progetto garantiscono una buona separazione del codice, dividendo le logiche relative ai dati con quelle relative l'interfaccia grafica. Così facendo la lettura del codice risulta essere facilitata e avere una migliore manutenibilità, in previsione di aggiornamenti futuri. Essendo un codice modulare, è possibile riutilizzarlo, riducendo la duplicazione che potrebbe portare ad errori implementativi.

4 Possibili Miglioramenti

La gestione delle dipendenze dei dati locali e dati da remoto dovrebbe essere gestita in modo più coerente poiché i dati locali hanno la priorità rispetto ai dati da remoto.

La sincronizzazione dei dati avviene in modo "macchinoso", poiché questi vengono aggiornati con il server solo all'avvio dell'applicazione.