Transaction Wallet With Connectivity And Synchronization Between Local Database And External REST API Custom Server

CIONT CRISTIAN

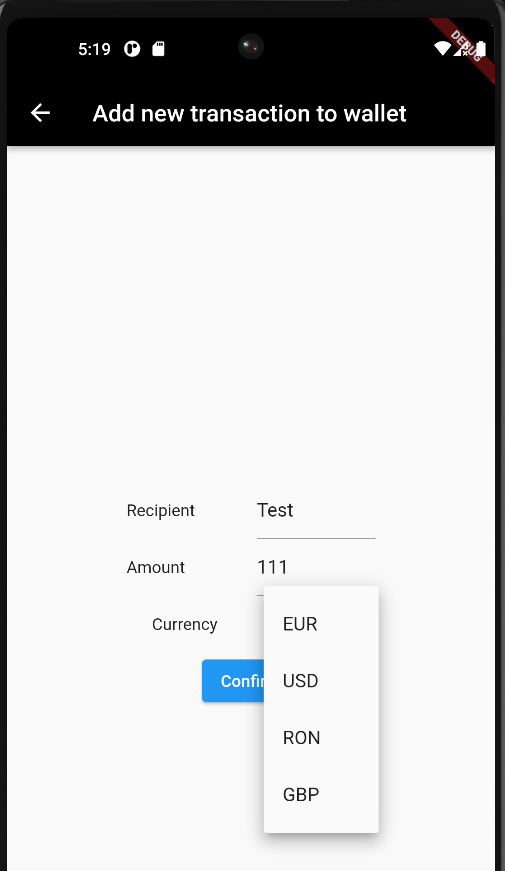
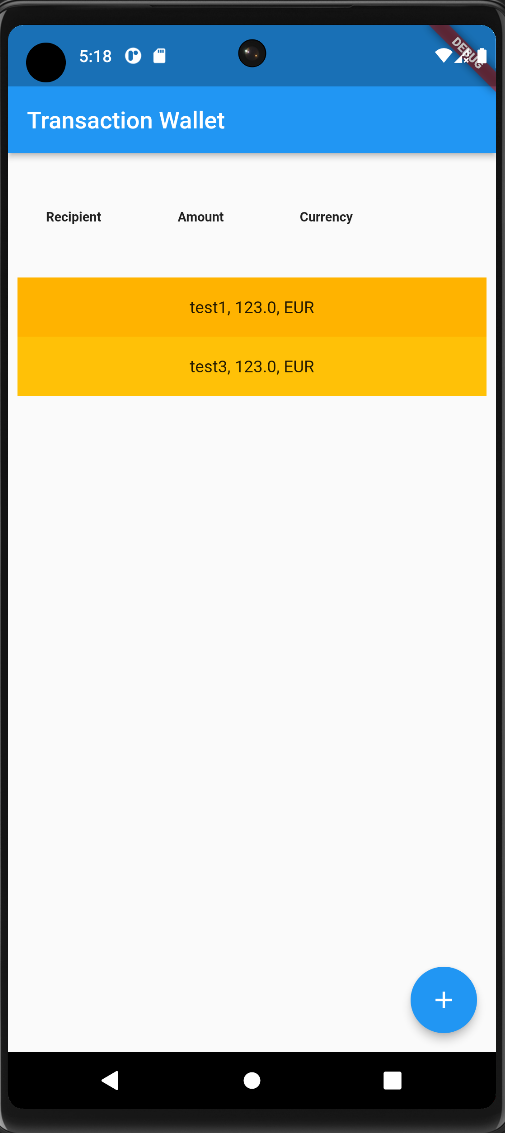
Elemente de M-Business

1. Prezentare

Aplicatia permite utilizatorilor sa isi salveze tranzactiile efectuate indiferent de starea conexiunii la Internet. In cazul in care nu exista o conexiune la Internet, toate tranzactiile vor fi salvate intr-o baza de date locala SQFLite. In momentul in care dispozitivul mobil se conecteaza la WiFi sau date mobile, aplicatia isi va sincroniza datele stocate cu o baza de date externa PostgreSQL dintr-un server extern construit folosind Spring Boot prin servicii de tip REST.

Flutter este un framework open source de la Google care permite crearea de aplicatii Android, compilate nativ, multi-platforma (web, mobile, embedded). Este un framework rapid, care permite recompilarea aplicatiilor in timp ce ruleaza cu ajutorul functiei Hot Reload.

Spring Boot este un framework din familia Java Spring, care permite crearea rapida de aplicatii, cu un server Tomcat integrat, cu configurari minime si care permite utilizarea conceptului de “Loosely Coupled Code” cu ajutorul injectarii de dependinte.



1. Implementare

Aplicatia a fost dezvoltata in Android Studio, folosind limbajul Flutter, si dispune de doua ecrane:

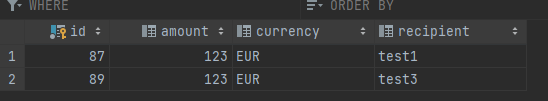
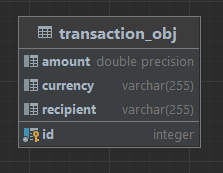
* Ecranul principal este folosit pentru majoritatea functionalitatilor. Contine lista cu

toate entitatile din baza de date, fie locala, fie externa, din care se poate ajunge printr-un click lung la afisarea unui Pop-up care serveste ca si confirmare pentru stergerea entitatii selectate. De asemenea, in partea de jos a ecranului este prezent si un buton pentru adaugarea unei noi entitati;

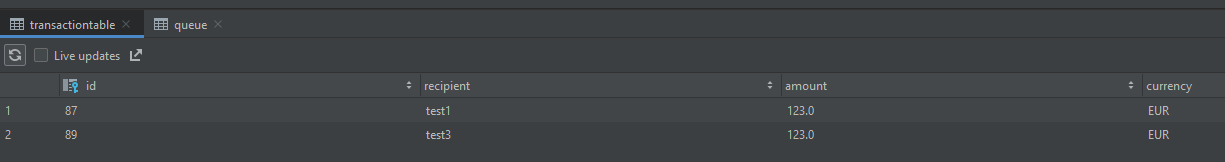
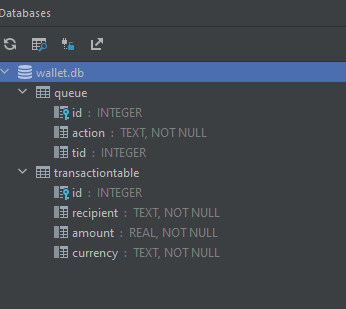
* Ecranul pentru adaugarea unei noi tranzactii, alaturi de diferite inputs care trebuie introduse;

Serverul a fost dezvoltat in IntelliJ, folosind limbajul Java, cu ajutorul framework-ului Spring Boot si dispune de o baza de date PostgreSQL si un serviciu REST care expune endpoint-uri de operatii CRUD pentru a fi consumate de clientul de Flutter.

Cele doua baze de date, respectiv din client si server vor fi mereu sincronizate in momentul in care dispozitivul mobil se conecteaza la Internet. Toate operatiunile efectuate offline vor fi asezate intr-un queue care le va trimite la server cand se detecteaza o conexiune la internet.

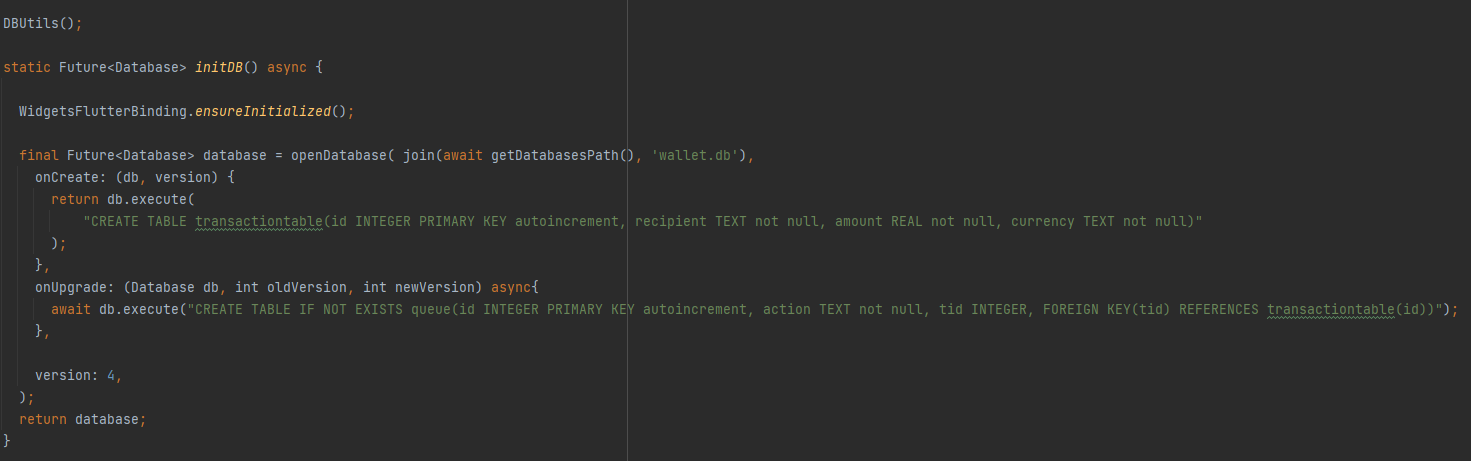


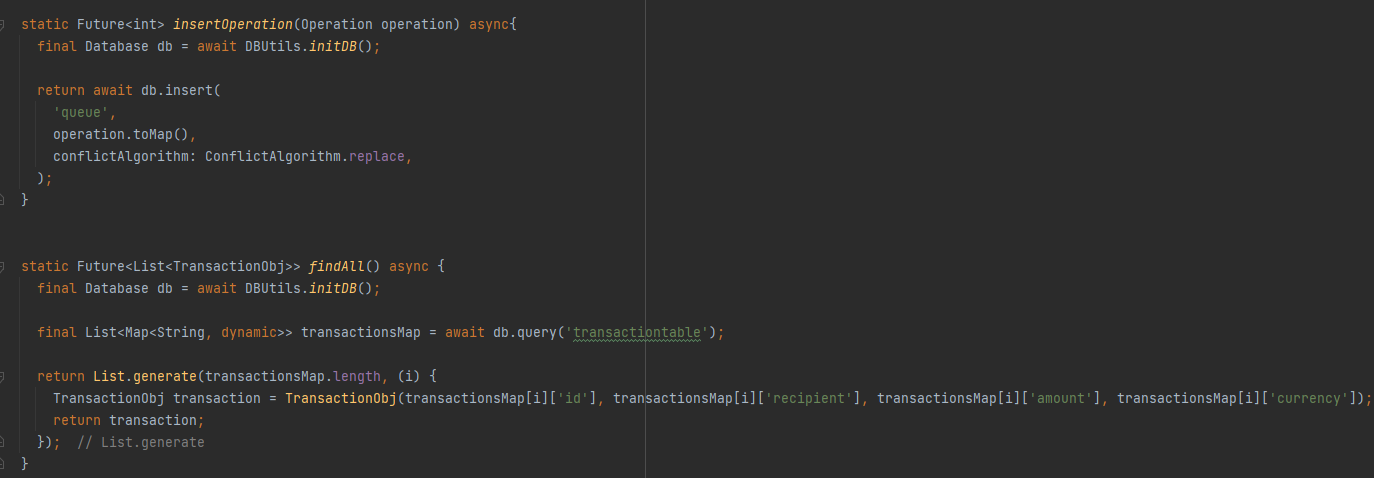
PostgreSQL Server Database Table si date stocate



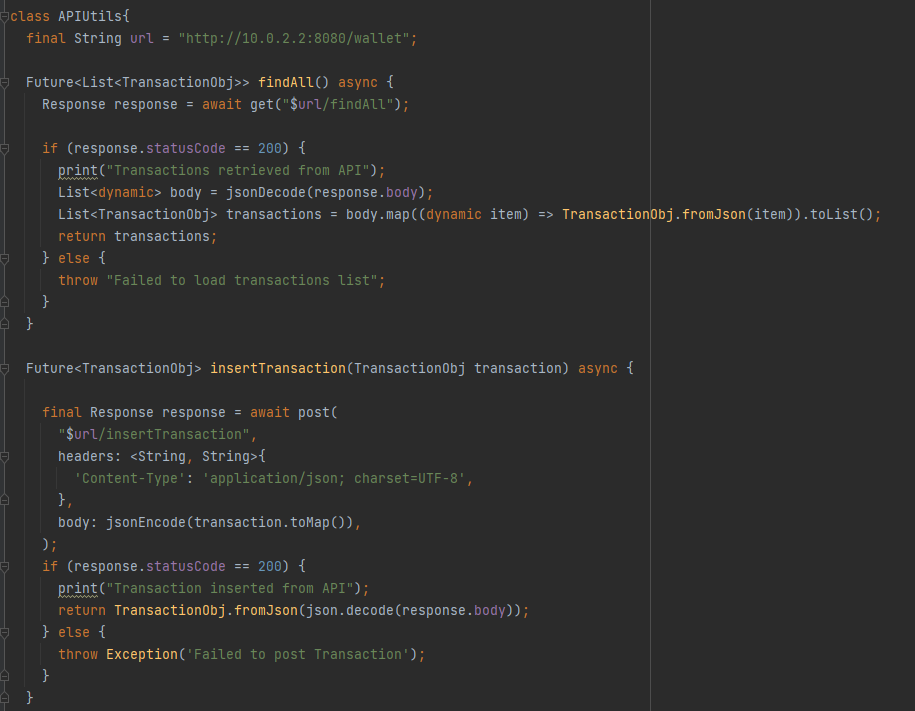
SQFLite Client Database Tables si date stocate

* Client (Flutter)





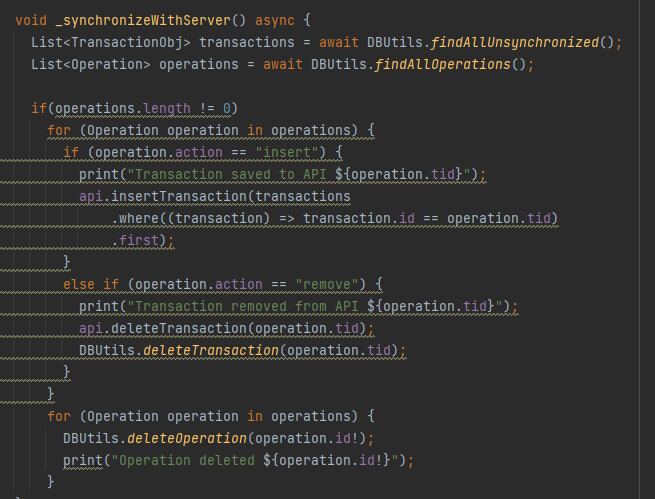
Clasa DBUtils se ocupa cu crearea tabelelor folosite local si a operatiilor CRUD de interactionare cu baza de date. Exista 2 tabele, transactiontable care stocheaza toate informatiile legate de tranzactii si queue care stocheaza toate operatiunile facute local care trebuie sincronizate cu serverul in momentul in care se detecteaza o conexiune la Internet.



Clasa APIUtils se ocupa cu comunicarea cu serverul de Spring Boot. Au fost definite functii care consuma endpoint-urile de REST oferite de server. Acest serviciu API este folosit cand avem o conexiune activa la Internet.

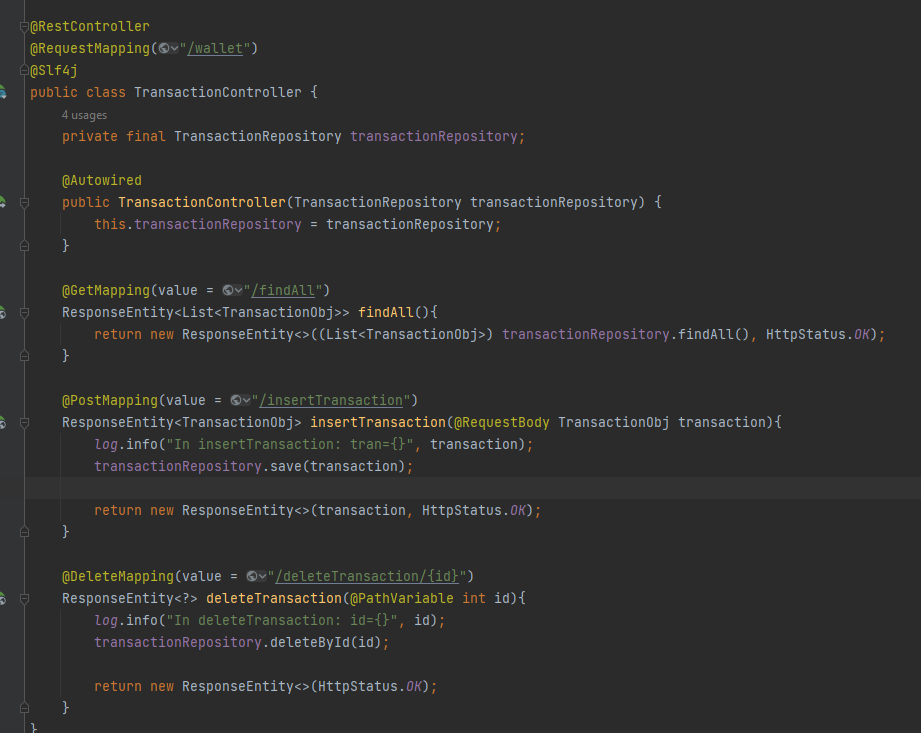


Functiile de checkConnectivity verifica care este starea conexiunii la Internet (Wifi, Date mobile sau fara internet) in momentul in care aplicatia este pornita sau activam sau dezactivam wifi/date mobile. In cazul unei conexiuni la Internet preluam datele de la server, iar in caz contrar din baza de date locala.



Functia synchronizeWithServer verifica ce operatiuni au fost executate cand internetul a fost oprit si incepe sa le trimita catre server pentru actualizare in momentul in care se primeste o conexiune de wifi sau date mobile.

* Spring Boot Server



Serverul de Spring Boot publica un controller REST cu endpoint-urile care pot fi consumate de catre clientul de Flutter. Acestea salveaza datele intr-o baza de date locala PostgreSQL, folosind Hibernate ca si ORM (Object Relational Mapping) pentru a nu fi nevoie de scrierea manuala a query-urilor.