**Tema 1 - Sistem de Internet banking**

\*cerintele cu bold sunt cele care afecteaza arhitectura

Cerinte functionale:

* **Utilizatorul va introduce parola de intenet banking, pe baza caruia se va genera un token. Cu acesta se poate loga,** cat timp este valabil, la serviciul de internet banking
* Tokenul generat va fi valabil 30 de secunde
* Introducerea de 3 ori a parolei de internet banking gresita va duce la blocarea accesului la acest serviciu - token-ul nu se va mai genera decat dupa deblocare accesului de catre banca
* Odata inceputa sesiunea, intre serverul bancii si utilizator va avea loc o conexiune securizata (criptata)
* **Utilizatorul va putea vedea toate conturile existente la banca pe numele sau,** putand sa aleaga oricare din ele **pentru a vedea informatii despre acesta sau a solicita diferite tranzactii**
* **Utilizatorul poate vedea toate tranzactile efectuate pe un anumit cont pe zile, luni sau ani, cu detalii despre data efectuarii tranzactiei, suma si beneficiarul/furnizorul**
* **Utilizatorul va putea vedea soldul contului**
* **Utilizatorul va putea transfera bani din contul sau catre alt cont**
* **Utilizatorul va putea scana codul QR de pe facturile firmelor partenere pentru a plati aproape instantaneu factura**
* Utilizatorul va putea crea si salva o lista de coduri IBAN. La efectuarea de transferuri bancare sau la plata une facturi fara cod QR acesta nu va mai trebui sa introduca de fiecare data codul IBAN al beneficiarului, ci il va alege din lista
* **Utilizatorul va putea vedea grafice ale situatiei contului sau si rapoarte privitoare la asta**
* **In cazul tranzactiilor catre alte banci (daca contul beneficiarului nu este la acceasi banca cu cea a utilizatorului) se va efectua transmiterea mai departe a mesajului cu cealalta banca**
* Pentru orice tranzactie efectuata cu succes serverul va afisa un mesaj de confirmare si va updata istoricul tranzactiilor
* Serverul va furniza catre utilizator, la cerere, urmatoarele date:
  + Conturile in care figureaza ca proprietar, din baza de date
  + istoricul tranzactiilor in intervalul cerut
  + soldul contului
  + grafice si rapoarte despre statusul contului intr-un anumit interval de timp
* Serverul va gestiona, in cazul tranzactiilor:
  + updatarea soludului utilizatorului,
  + updatarea soldului beneficiarului daca acesta este la acceasi banca
  + salvarea in istoricul tranzactiilor a acesteia - de ambele parti daca clientii sunt de la aceeasi banca

Cerinte nonfunctionale:

* Baza de date cu utilizatorii si tranzactiile lor va fi stocata pe serverele bancii
* Comunicarea din cadrul sesiunii se va realiza printr-un tunel securizat
* serverul trebuie sa fie capabil sa gestioneze pana la 10 000 de clienti simultan
* aplicatia de internet bankig va putea fi accesata de pe platforme diferite: android, iOS sau site web (desktop)
* Baza de date cu IBAN-ul firmelor partenere pt plata facturilor va fi stocata pe server si trimisa catre aplicatia utilizatorului la logare
* Baza de date cu IBAN-uri salvate de utilizator va fi stocata local

Module, responsabilitati ale acestora, relatii dintre ele:

* UI (User interface): preia comenzile de la utilizator si le trimite la modulele responsabile. Afiseaza utilizatorului raspunsurile serverului. Permite crearea de liste cu IBAN-uri ale utilizatorului
* Login: verifica parola si username-ul si, daca corespund unui client din baza de date, returnează tokenul. Daca modulul de login nu da aprobarea, aplicatia nu poate continua spre oferirea serviciilor acesteia. Daca sunt 3 incercari nereusite va afisa (prin UI), numarul de telefon la care se poate suna pt deblocarea contului. Daca logarea este cu succes face un request la server pentru datele despre cont si datele despre firmele partnere
* History: gestioneaza istoricul tranzactiilor sau il returneaza, la cererea celorlalte module. Informatiile vin din baza de date de la server. Acest modul le poate ordona sau poate permite stocarea istoricului recent la nivel intern, pentru a evita retransmiterea
* Account: preia datele despre utilizator de la server (conturi, sold) si le stocheaza intern, pt a nu fi nevoie sa fie cerute din nou de la server de fiecare data
* Transaction: primeste, prin UI sau Pay, suma de transmis si IBAN-ul beneficiarului si updateaza in baza de date a serverului soldul utilizatorului (in minus) si al beneficiarului (in plus) - daca e cazul, daca nu, este responsabila si sa trimita un request la modulul Communication pt a comunica cu banca la care este contul beneficiarul. Asteapta confirmarea de la modului Communication ca mesajele a fost trimis cu succes catre aceasta banca pentru a considera tranzactia efectuata. Trimite request la modulul de History pt a salva tranzactia (sau incercarea de tranzactie daca a fost nereusit). Trimite inapoi spre UI daca tranzactia a reusit sau nu
* Communicaton: gestioneaza comunicarea cu o alta banca printr-o succesiune de mesaje, cu un protocol dinainte stabilit
* Report: primeste, prin UI tipul de tranzactii pt care se vrea raportul si intervalul de timp cuprins in raport si genereaza un raport pdf. Genereaza,la cerere din UI, grafice cu evolutia soldului sau a platilor in intervalul de timp solicitat. Pentru toate acestea va folosi datele date de modulul de history
* Pay: Interpreteaza codul QR al facturii (fotografiat) pt a deduce firma beneficiara si suma de plata. Trimite un request cu datele obtinute la modului Transaction pt a realiza tranzactia
* Server: contine bazele de date ale istoricului tranzactiilor, datelor despre cont si IBAN-urile firmelor partenere. Modulele o pot interoga prin intermediul lui

Argumentare, observatii:

Modulul Login interogheaza baza de date de pe server pentru a verifica corectitudinea datelor de login. Datele despre cont si IBAN-urile firmelor partenere se obtin la login si se salveaza in structuri proprii, la nivel intern, (prin modulul Account) pentru ca sunt date care se vor modifica rar sau deloc in cursul unei sesiuni deci ar fi ineficienta retransmiterea lor oricand e nevoie. Datele despre istoric sunt preluate de la server de catre modulul History (dupa un request care vine de la acesta), care are si atributiile de a face informatiile despre istoric mult mai usor de utilizat de catre celalate module. Istoricul recent (cel mai frecvent cerut) va fi salvat si el intern, de catre modulul History, pentru a evita retransmisia. Astfel, se evita supraincarcarea serverului, care trebuie sa poata suporta pana la 10000 de utilizatori simultan - se doreste limitarea, pe cat posibil a retransmisiei. Cand modulul Transactions va cere serverului modificari in baza de date, tot el va cere si updatarea in structurile interne ale modulelor a soldului si istoricului. Astfel informatiile vor actuale si in aplicatie, fara a mai fi din nou preluate de pe server.

Diagrama sistemului din perspectiva logica (diagrama de componente): (se gaseste si atasata si in folderul din GitHub, este schema simplificata din cauza limitarilor LucidChart)

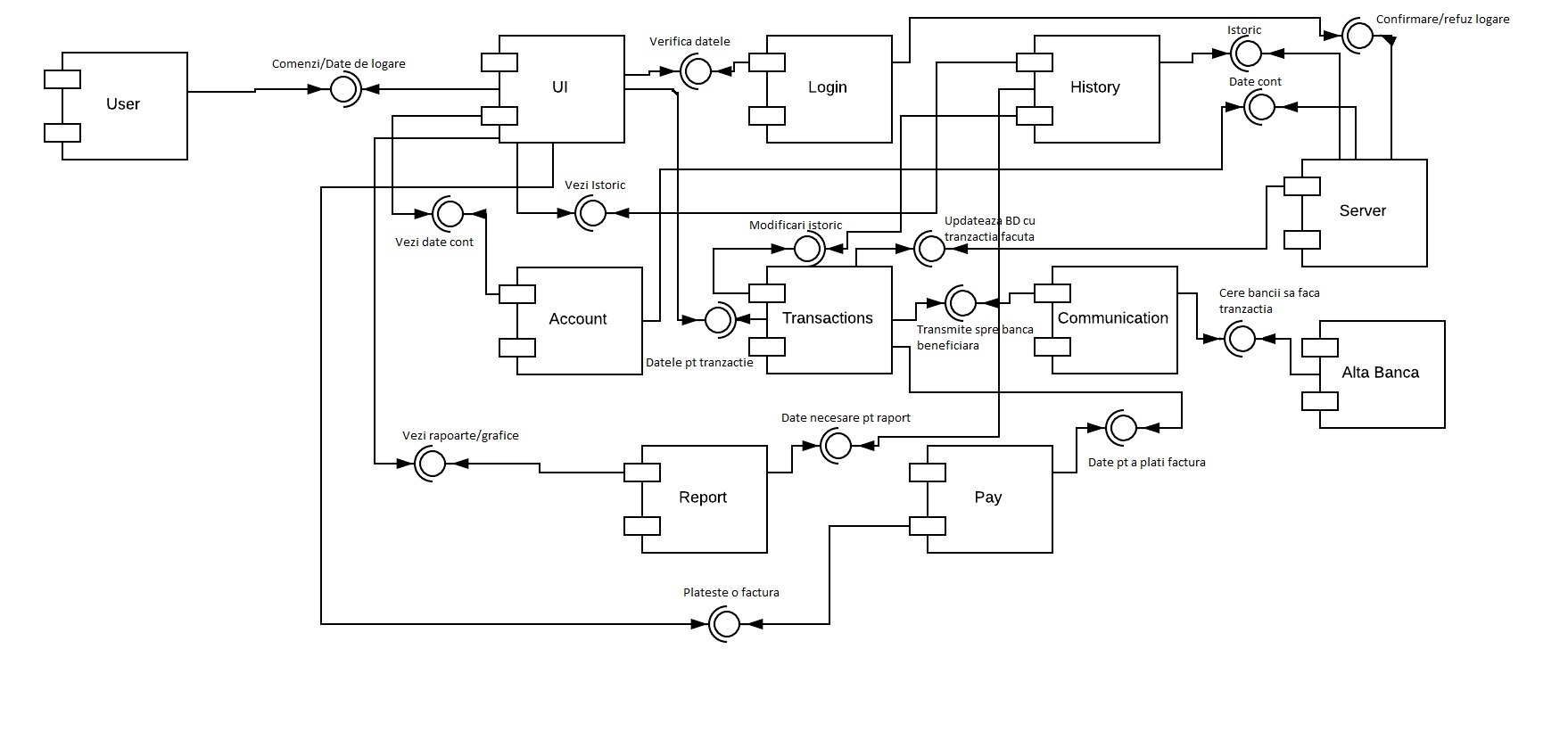
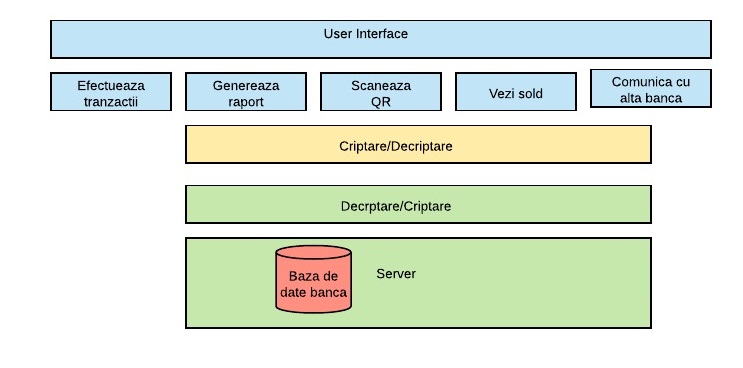


Diagrama arhitecturii din perspectiva logica:

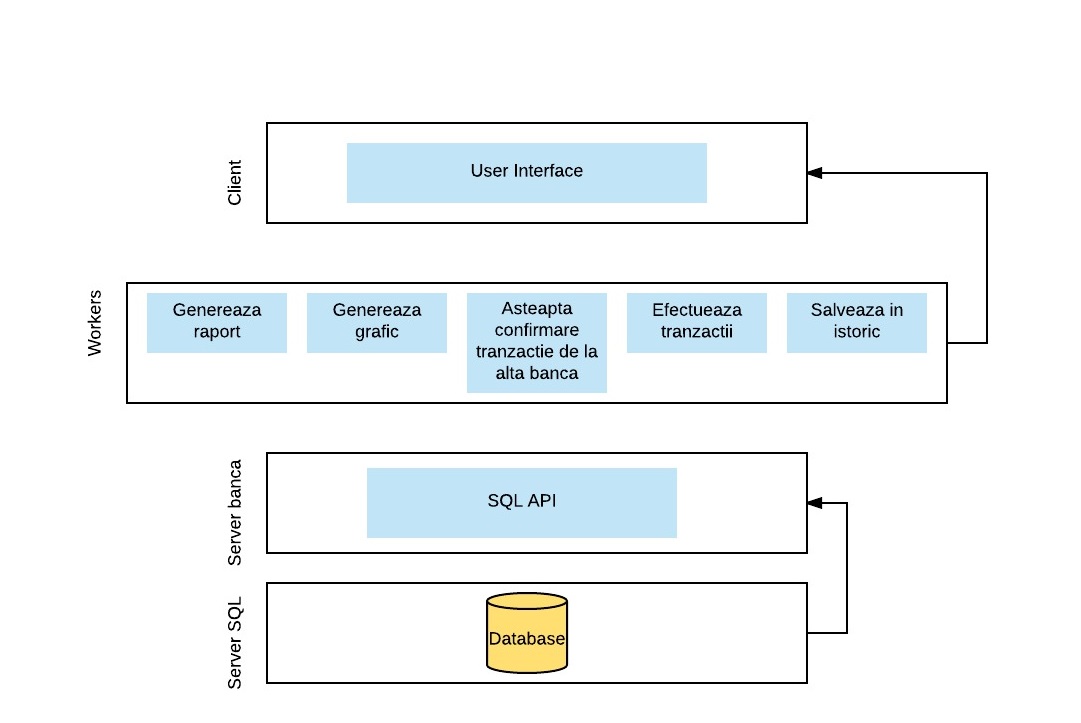


Explicatii: Diagrama descrie interactiunea dintre module cum e prezentata mai sus. Practic, utilzatorul da comenzi interfetei grafice (UI) care le trimite mai departe modulelor responsabile cu acele comenzi. De ex la comanda “log in” modulul de login preia datele din formurale interfetei si le trimite la server pt a le verifica, la comanda “raport”, modulul raport trimite raportul inapoi catre interfata folosind date preluate de la modulul History.

Modulul Transactions cere modificarea Bazei de date de catre server, sau lansarea unei comunicatii cu o alta banca prin modulul Communication, in functie de banca beneficiara. Tot el cere si updatarea informatiilor din modulul History.

Se poate observa cum arhitectura este impartita pe 3 nivele din a 2-a diagrama.

Diagrama arhitecturii din perspectiva proces:



Explicatii: aplicatia ruleaza 2 procese: cea pentru UI si cea care contine thread-uri ce ruleaza in fundal pentru ca sunt mai costisitoare computational. Interfata asteapta rezultatul lor pentru a afisa ce i-a cerut utilizatorul. De partea bancii serverul are un API SQL pe care il va folosi pt a interoga sau updata baza de date cu datele primite, dupa ce le decripteaza.

Indicatori de calitate urmariti:

* Scalabilitatea (nr maxim de conexiuni simultane) : se realizeaza prin reducerea pe cat posibil a retransmisiei datelor: datele sunt aduse de pe server la nivel local la inceput. Daca se fac tranzactii sau plati se updateaza baza de date, dar nu se mai cer inca o data toate datele de la server, ci se updateaza si datele stocate local, astfel informatia este actuala pentru utilizator fara a fi nevoie de comunicare suplimentara cu serverul.
* Securitatea: se realizeaza, in cadrul aplicatiei prin autentificare si criptare a datelor care se vehiculeaza intre server si aplicatie sau intre aplicatie si o alta banca.
* Portabilitate: aplicatia va fi construita pt mai multe sisteme de operare (iOS si Android) si web

Model arhitectural folosit:

Three-Tier - avem un modul de interfata cu utilizatorul (UI), mai multe module care sunt responsabile de functionalitatea efectiva (Raport, Transaction, History, Pay, Account) si un modul in care se stocheaza datele (Server). UI trimite cereri catre modulele de functionalitate si informatiile de care acestea au nevoie. Modulele de functionalitate opereaza cu date luate in prealabil de la server. S-a ales aces stil pentru ca este potrvit unei aplicatii web. In plus, datele despre cont erau stocate pe un server la distanta, era nevoie de o interfata cu utilizatorul dar si de module care, intre ceea ce cere utilizatorul si server sa efectueze operatiile necesare pentru a cere apoi serverului direct modificarile de care are nevoie.

Stiluri arhitecturale::

* Bazat pe componente: pentru o mai usoara intretinere si dezvoltare ulterioara
* Bazat pe inlantuire: UI->Module de functionalitate->Server. Pentru ca serverul trebuie sa dea ceea ce ii cere utilizatorul in UI, dar intrucat interfata trebui sa fie simpla si prietenoasa, modulele de functionalitate preiau cererea acestui si o traduc in date directe pentru server. Serverul nu mai face el calculele, dar se bazeaza de calulele facute de modulele de functionalitate, care la randul lor se bazeaza pe cerea utilizatorului
* Middleware bazat pe mesaje: pentru ca avem un emitator (utilizatorul) care prin intermediul aplicatiei transmite mesaje catre receptor (serverul cu baza de date)
* Client Server: pentru ca, in esenta relatia dintre aplicatie si server este una de tip client server

Tehnologii middleware:

* Middleware bazat pe mesaje (intre aplicatie si baza de date) :ales din cauza ca permite portabilitatea intre platforme diferite. In loc sa se apeleze direct baza de date, se trimite un mesaj criptat serverului care interogheaza sau update-aza baza de date in functie de aceste
* Midleware tip Remote Procedure call (intre module): modulele comunica intre ele prin apelarea unor functii care realizeaza interfata dintre ele. Ales din cauza ca este cel mai firesc de folosit in aceasta situatie. Nu voi avea nevoie ca un modul sa faca ceva decat daca ii se cere (se apeleaza o functie a sa)

Scenarii de validare (testare):

-Serverul nu cade atunci cand, prin intermediul unui test automat se simuleaza concomitent 10 000 de request-uri din aplicatie

-Se verifica ca tranzactiile se efectueaza in mod corect, in sensul in care suma extrasa este cea ceruta, iar suma primita de beneficiar este aceeasi. Se simuleaza mai multe tipuri de erori (caderea conexiunii, blocarea aplicatiei etc) care pot aparea si se verifica ca, in acest caz, se face roll-out si tranzactia nu se efectueaza pentru ca exista riscul efectuarii ei incorecte

- se utilizeaza diverse coduri QR de pe facturi pentru a se verifica ca aplicatia le interpreaza corect si este capabila sa duca la bun sfarsit o astfel de tranzactie

-se introduce o parola gresita de 3 ori pentru a verifica ca in acest caz accesul utilizatorului la internet banking este blocat ( pentru a preveni atacurile de tip dictionar)

-se incearca interceptarea datelor dintre server si apicatie, pentru a verifica ca datele sunt criptate (informatia obtinuta este irelevanta)

- se verifica ca, la efectuarea mai multor tranzactii aplicatia a inregistrat corect istoricul

- se verifica ca datele despre cont si istoric afisate de aplicatie corespund realitatii din baza de date

- se verifica ca rapoartele sunt generate corect

- se verifica ca introducerea unui nou modul de functionalitate nu destabilizeaza sistemul