Universitatea POLITEHNICA din București
Facultatea de Electronică, Telecomunicații și Tehnologia Informației

Aplicație web pentru administrarea conturilor bancare

Proiect de diplomă

prezentat ca cerință parțială pentru obținerea titlului de *Inginer*
în domeniul *Electronică, Telecomunicații și Tehnologia Informației*
programul de studii *Microelectronică, Optoelectronică și Nanotehnologii*

Conducători științifici
Prof. Univ. Dr. Ing. Corneliu BURILEANU
As. Univ. Dr. Ing. Ana NEACȘU
Dr. Ing. Dragoș DRĂGHICESCU

Absolvent
Ana MACARI

Anul 2020

Universitatea "Politehnica" din Bucureşti
Facultatea de Electronică, Telecomunicaţii şi Tehnologia Informaţiei
Departamentul **DCAE**

Anexa 1

TEMA PROIECTULUI DE DIPLOMĂ
a studentului **MACARI Ana , 445E-MON.**

1. Titlul temei: Sistem de administrare a conturilor bancare

2. Descrierea contribuţiei originale a studentului (în afara părţii de documentare) şi specificaţii de proiectare:
Dezvoltarea unei aplicații web de tip portofel digital în care vor fi stocate în mod compact și sigur datele tuturor cardurilor deținute de utilizator, chiar dacă conturile sunt afiliate unor bănci diferite.
În aplicație vor fi disponibile informații aferente fiecărui cont, cum ar fi soldul curent, IBAN-ul, tranzacțiile etc. Respectivele date sunt aduse în aplicație (unde urmează să fie prelucrate) în urma autentificării utilizatorului în aplicația de Internet Banking și prin obținerea acordului pentru furnizarea informațiilor către portofelul digital în cadrul unui flux OAuth 2.0. Autentificarea în aplicație se face în doi pași și anume prin verificarea existenței a doi factori: unul de cunoaștere (username și parola introduse de utilizator) și celălalt fiind de posesie (posibilitatea accesării Google Authenticator pe telefonul mobil).
Pentru ca nivelul de securitate să fie unul cât mai înalt, datele folosite la autentificare sunt criptate. În acest sens, lucrarea de față propune studiul mai multor algoritmi de criptare și alegerea unuia dintre ei pentru implementarea finală. De asemenea, aplicația va permite efectuarea transferurilor bancare între cardurile salvate ale utilizatorului.

3. Resurse folosite la dezvoltarea proiectului:
API-urile băncilor expuse în urma directivei UE privind serviciile de plata (PSD2), API-uri REST pentru comunicarea dintre backend și frontend, MySQL, Sequelize, JavaScript, Node.JS, Express, Axios, Vue.JS, Vuex.

4. Proiectul se bazează pe cunoştinţe dobîndite în principal la următoarele 3-4 discipline:
PBD, TTI, POO

5. Proprietatea intelectuală asupra proiectului aparţine: U.P.B.

6. Data înregistrării temei: 2020-01-16 11:59:11

Conducător(i) lucrare,
Prof. dr. ing. Corneliu BURILEANU

Student,
Macari

As. Univ. drd. Ing. Ana Neacșu, Dr. Ing. Dragoș Drăghincescu,

Director departament,
Prof. dr. ing Claudius DAN

Decan,
Prof. dr. ing. Mihnea UDREA

Cod Validare: **d4a9c391fa**

Anexa 5

Declarație de onestitate academică

Prin prezenta declar că lucrarea cu titlul *Aplicație web pentru administrarea conturilor bancare*, prezentată în cadrul Facultății de Electronică, Telecomunicații și Tehnologia Informației a Universității "Politehnica" din București ca cerință parțială pentru obținerea titlului de *Inginer* în domeniul *Inginerie Electronică și Telecomunicații/ Calculatoare și Tehnologia Informației,* programul de studii *Microelectronică, Optoelectronică și Nanotehnologii* este scrisă de mine și nu a mai fost prezentată niciodată la o facultate sau instituție de învățământ superior din țară sau străinătate.

Declar că toate sursele utilizate, inclusiv cele de pe Internet, sunt indicate în lucrare, ca referințe bibliografice. Fragmentele de text din alte surse, reproduse exact, chiar și în traducere proprie din altă limbă, sunt scrise între ghilimele și fac referință la sursă. Reformularea în cuvinte proprii a textelor scrise de către alți autori face referință la sursă. Înțeleg că plagiatul constituie infracțiune și se sancționează conform legilor în vigoare.

Declar că toate rezultatele simulărilor, experimentelor și măsurătorilor pe care le prezint ca fiind făcute de mine, precum și metodele prin care au fost obținute, sunt reale și provin din respectivele simulări, experimente și măsurători. Înțeleg că falsificarea datelor și rezultatelor constituie fraudă și se sancționează conform regulamentelor în vigoare.

București, 23 Iunie 2020

Absolvent *Ana MACARI*

Macari

Cuprins

Lista figurilor

Lista acronimelor

2FA = Two-Factor Authentication
AES = Advanced Encryption Standard
AIS = Account Information Service
AISP = Account Information Service Provider
API = Application Program Interface
ASPSP = Account Servicing Payment Service Provider
ASRO = Asociația de Standardizare din România
CNSS = Committee of National Security Systems
CRUD = Create-Read-Update-Delete
CVV = Card Verification Value
DDL = Data Definition Language
DES = Data Encryption Standard
HTTP = Hyper Text Transfer Protocol
HTTPS = Hyper Text Transfer Protocol Secure
IDE = Integrated Development Environment
IEC = International Electrotechnical Commission
ISO = International Standards Organization
DML = Data Manipulation Language
MD5 = Message Digest 5
NIST = National Institute of Standards and Technology
ORM = Object - Relational Mapping
OTP = One - Time Password
PIS = Payment Initiation Service
PISP = Payment Initiation Service Provider
PSD = Payment Service Directive
RSA = Rivest - Shamir - Adleman
SAC = Strong Authentication Client
SHA = Secure Hash Algorithm
SPA = Single - Page Application
SSL= Secure Sockets Layer
TLS = Transport Layer Security
TPP = Third - Party Provider
UI = User Interface

INTRODUCERE

„Comunicarea duce la comunitate și anume la: înțelegere, intimitate și valori comune”.

Rollo May

La început a fost cuvântul, apoi scrisorile pe aripi de porumbel sau în sticle pe mare, ca într-un final să ajungem la telefon și internet. Așa a luat naștere informația. Or, informația egalează cu puterea, iar dorința de a asigura protecție și de a nu o face accesibilă oricui, a fost unul din scopurile marilor intelectuali.

Din cele mai vechi timpuri, din filele cărților de istorie, aflăm că oamenii au folosit o multitudine de metode pentru a trimite mesaje secrete. Încă din scrierile lui Herodot descoperim cum arta scrierii secrete a salvat Grecia în 480 î. Hr. de armata persană, printr-o tăbliță de lemn scrijelită și apoi acoperită de ceară.

Comunicarea secretă sau ascunderea mesajului a cunoscut diferite mijloace inedite în primele secole: de la scrierea mesajelor cu bastonașe înfășurate cu papirus sau pergament în spirală, la cerneală invizibilă care sub acțiunea doar a unor anumiți catalizatori putea fi citită, până la scrierea mesajului pe o bucată de mătase ce urma a fi împăturită și acoperită cu ceară pentru ca mesagerul să o poată înghiți, obicei întâlnit la vechii chinezi.

Ținând cont de legea atracției universale, secretul vine la pachet cu opusul său – trădarea și dezvăluirea acestuia. Astfel „defectul major” al comunicării secrete, este acela că odată descoperit mesajul, acesta poate fi descifrat, căci toate eforturile se concentrau pe ascunderea mesajului și nu pe codificarea acestuia. Prin urmare, apariția codificării mesajului era imperios necesară nu pentru a ascunde indubitabil mesajele, ci pentru a le face inteligibile pentru cei care ar putea să le descopere.

Așa a apărut – **criptografia** știința care se ocupă cu studiul codurilor și al cifrurilor, iar acest concept a condiționat apariția unei ale științe și anume – **Securitatea informației.**

Cultura securității informației este produsul măsurilor și mecanismelor de securitate, care presupun: garantarea integrității, confidențialității și accesibilității informației. Evoluția și schimbarea tehnologiilor informaționale care sunt exponențiale, obligă la: gestiune, structurare, perceperea tipologiei amenințărilor, diminuarea riscurilor și vulnerabilităților particulare societății informaționale, precum și la fixarea practicilor de control.

Din necesitatea uniformizării și asigurării unui nivel înalt de protecție a sistemelor de control, *Organizația Internațională pentru Standardizare (ISO)* și *Comisia Internațională Electrotehnică (IEC)* au realizat standardizarea securității informaționale. Începând cu anul 2004, Asociația de Standardizare din România (ASRO) a adoptat standardele europene și internaționale - Standardul ISO/IEC 17799 - „Tehnologia Informației - Cod de bună practică pentru managementul securității informației”.

Acum, în Era Tehnologiei, securitatea reprezintă una dintre cele mai importante și complexe științe. Datorită accesului facil la internet, informația nu mai are limite, astfel asigurarea securității informației este absolut necesară, deoarece atât timp cât atributele ei de securitate sunt sigure, informația nu își pierde valoarea. De calitatea și evoluția securității sistemelor informaționale și a criptografiei depinde succesul Epocii Informaționale, garantând lacătele și cheile datelor celor mai de preț.

Aplicabilitate

Societatea informațională se bazează pe crearea, partajarea, utilizarea, înglobarea și manipularea informațiilor cu impact semnificativ în aspectele organizatorice ale tuturor domeniilor: social, politic, economic, industrial, educațional, cultural și al sănătății. Principalii piloni ai societății informaționale - *tehnologiile informaționale* și *comunicarea digitală* - prezintă un traiect ascendent, datorită faptului că volumul de cunoștințe se dublează odată la cinci ani [1].

Dimensiunile societății informaționale și ale cunoașterii (SIC)

- *Socială* – vizează îngrijirea sănătății și protecției sociale, democrației sociale (telemedicina, teleactivități, telealegeri, teleasigurare, etc.);
- *Educațională* – dezvoltarea competențelor digitale și gestionarea inteligentă a proceselor educaționale (învățământ la distanță, biblioteci virtuale, e-Teaching, e-Learning);
- *Ambientală* – urmărește utilizarea resurselor și protecției mediului înconjurător;
- *Culturală* – vizează conservarea și dezvoltarea patrimoniului și al industriei (muzee, galerii de artă online), digitizarea informației (manuale digitizate);
- *Economică* – dezvoltarea economiei digitale, educației consumatorului și culturii antreprenoriale și manageriale (e-Learning, e-Banking, e-Comerț, e-Money, e-Trading, efectuarea plăților online, afacere pe internet).

Dimensiunile societății informaționale (SI)

- *Tehnologică* – servicii, aplicații și securitatea informatică;
- *Socială* – calitatea vieții, protecția datelor cu caracter privat;
- *Culturală* – interacțiunea cultură-tehnologie;
- *Economică* – liberalizarea comerțului cu suprimarea barierelor, investiții private și digitalizare;
- *Politico-administrativă* – guvernare electronică;
- *Juridică* – legislație specific.

Obiectele critice supuse securizării, controlului și gestiunii în condiții de siguranță sunt:

- Sistemele de telecomunicație;
- Sistemele bancare;
- Sistemele de administrare a transportului atât aerian, cât și rutier;
- Stațiile atomice;
- Sistemele de prelucrare și stocare a informațiilor secrete și confidențiale.

Obiective

Scopul prezentei lucrări este dezvoltarea unei aplicații web de tipul unui portofel electronic, care permite consultarea mai multor conturi bancare ale aceluiași utilizator, care provin de la bănci diferite. În ceea ce privește conturile bancare, în aplicație, vor fi disponibile informații precum: soldul curent, IBAN, istoricul tranzacțiilor etc. Respectivele date sunt disponibile în urma obținerii acordului din partea utilizatorului pentru a accesa informațiile financiare în cadrul unui flux OAuth 2.0 inițiat prin autentificarea utilizatorului în aplicația de Internet Banking a băncii respective. Întrucât asigurarea securității este o prioritate, autentificarea în portofelul digital se face în doi pași prin verificarea existenței a doi factori: primul de cunoaștere (username și parola) și cel de al doilea de posesie (OTP expediat prin e-mail). Pentru a prezenta un nivel de securitate cât mai înalt, datele schimbate între server și client în timpul procesului de autentificare sunt criptate prin intermediul unul algoritm ce se bazează pe teorema numerelor prime mari – RSA, astfel încât nimeni nu poată intercepta comunicația.

Avantajele implementării unei astfel de soluții sunt:

- Accesarea sigură, rapidă și concomitentă a informațiilor, aferente mai multor conturi bancare, expusă ca un întreg pe un tablou de bord;
- Prezentarea într-un mod facil a informațiilor necesare transferurilor bancare: suma tranzacției, numărul contului, mesajul, data efectuării;
- Disponibilitatea prezentării tuturor tranzacțiilor efectuate prin intermediul acestor conturi bancare;
- Posibilitatea setării unor limite de cheltuieli pentru diferite categorii pe o anumită perioadă;
- Posibilitatea reprezentării grafice a celor mai recente cheltuieli.

Capitolul 1
SECURITATEA INFORMAȚIEI

1.1. Noțiuni generale

Istoricul securității informațiilor a debutat odată cu securitatea computerului, care avea drept scop protejarea locațiilor fizice, hardware-ul și software-ul împotriva amenințărilor. Necesitatea de a conecta rețelele de calculatoare între ele, a condiționat apariția și dezvoltarea Internetului, care a ajuns la publicul larg în anii ’90. Datorită volumului vast de informații și a necesității de prelucrare în timp real, cu costuri minime și accesibilitate mondială, sistemele informaționale au fost interconectate la rețele.

Inițial, Internetul – instrumentul de partajare a informațiilor nu fost supus unor standarde, iar informațiile stocate au devenit expuse amenințărilor de securitate. În ultimii ani, accentul pus pe securitatea sistemelor informaționale a crescut exponențial datorită conexiunii prin sisteme de operare comerciale și standarde mondiale precum Ethernet, TCP / IP, dar și a tehnologiilor web.

Definiție. ***Securitate*** - „calitatea sau starea de a fi sigur – de a fi ferit de pericole” [2]. Conform CNSS-ului - comitetul național pentru sisteme de securitate - **securitatea informațiilor** este definită drept protecția datelor și a elementelor sale decisive, incluzând hardware-ul, sistemele care stochează, folosesc și transmit informația, indiferent de natura acesteia. Modelul CNSS de securitate a informațiilor a evoluat de la o idee concepută de industria securității informatice, cunoscut sub denumirea de **triunghiul C.I.A**. Odată cu dezvoltarea mainframe-ului modelul C.I.A. a reprezentat standardul industriei pentru securitatea calculatorului, devenind modelul utilizat pe scară largă în estimarea eficacității securității sistemelor informaționale. Acesta se bazează pe cele trei caracteristici care oferă valoare informațiilor și anume: confidențialitate, integritate și accesibilitate.

Confidențialitatea asigură păstrarea datelor secrete, astfel încât, doar persoanele sau dispozitivele autorizate să poată accesa aceste informații. Iar dezvăluirea către entitățile neautorizate constituie o infracțiune.

Integritatea vizează păstrarea datelor în forma lor originală, garantând nealterarea acestora în procesul de transfer dintre sursă și destinație, nici prin căi ilegale, nici accidental. Integritatea este o măsură a corectitudinii datelor și a asigurării încrederii în ele.

Accesibilitatea este factorul care asigură disponibilitatea datelor și resurselor în orice moment, persoanelor și dispozitivelor autorizate.

Pericolele aduse la adresa confidențialității, integrității și disponibilității informațiilor s-au dezvoltat într-o amplă colecție de evenimente nefaste, inclusiv daune intenționate sau neintenționate, distrugeri, furturi informaționale, modificări deliberate cât și accidentale, sau alte utilizări necorespunzătoare, cauzate în urma unor amenințări. Pentru a combate pericolele care sunt în continuă evoluție modelul C.I.A. încă se updatează.

1.2. Concepte cheie despre securitatea informației

Securitatea întrebuințează două strategii de bază:

- Tot ceea ce nu este permis este interzis;
- Tot ceea ce nu este interzis este permis.

Pentru protejarea informațiilor se apelează la doua tactici de implementare:

- Controlul legal al accesului;
- Controlul discreționar al accesului.

Definiție. ***Amenințările de securitate*** sunt circumstanțele sau evenimentele care pot provoca distrugerea, dezvăluirea și modificarea datelor, provenind atât din interiorul, cât și din exteriorul unei instalații/organizații.

Amenințările interne își au originile în:

- *Incidentele accidentale* cauzate de către un angajat necunoscut și neinstruit care execută o acțiune greșită.
- *Incidentele intenționate* cauzate de către angajați sau antreprenori care dețin cunoștințe despre sistemul de control și acces autorizat.

Amenințările externe ale sistemelor informaționale pot fi grupate după cum urmează:

- *Programele malware* – chiar dacă nu sunt direcționate și nu atacă sistemele informaționale, afectează sistemul stopând comunicațiile, degradând datele și provocând opriri forțate.
- *Hackeri* – persoane intenționate să sondeze, pătrundă și să dețină controlul unui sistem de securitate, computer sau rețea pentru a avea acces la informații confidențiale [3].
- *Teroriști* – persoane intenționate să distrugă sistemele de infrastructură IT.

Încercarea de securizare maximă a sistemelor restricționează accesul utilizatorului la informații și impune costuri. Deoarece nu poate fi garantată securitatea integrală a sistemelor, utilizatorii trebuie să își concentreze atenția pe domenii și funcții critice și să aplice măsuri de securitate justificate prin valoarea datelor și a aplicației. În principiu, aplicarea măsurilor de securitate trebuie să fie proporționale din punct de vedere al costurilor cu valoarea datelor, riscului și probabilității asociate unui incident de securitate, cât și consecințelor incidentului [4].

Reducerea cheltuielilor în acest sens poate fi obținută prin aplicarea unor practici și politici bine definite și eficiente de către personalul calificat.

Nu există soluții universale sau mecanisme complet pliabile pe orice problemă, cum nu există nici reguli absolute care să reglementeze securitatea tuturor sistemelor. Protecția sistemelor poate fi asigurată corespunzător prin înțelegerea riscurilor și stabilirea unor limite eficiente ale sistemele informaționale și ale dispozitivelor cu care interacționează. Acest efort necesită să se concentreze nu numai pe tehnologie, ci și pe oameni [5].

În concluzie, elementul cheie în implementarea și menținerea securității unui sistem informatic este stabilirea unor politici și proceduri eficiente de securitate IT. Pentru ca politica de securitate să fie eficientă trebuie să valideze următoarele caracteristici: practică, aplicabilă, prohibitivă și să nu afecteze productivitatea [6]. Securizarea unui sistem informațional este un conglomerat de oameni, relații, procese și organizații ce colaborează eficient.

1.3. Caracteristicile informației

Valoarea informației este dictată de către caracteristicile acesteia. Astfel, modificarea unei caracteristici a informației se răsfrânge și asupra valorii, care de cele mai multe ori scade. Acest lucru este constrâns de circumstanțele în care apar modificări, de exemplu actualitatea informațiilor constituie un factor decisiv, dacă aceasta nu este prezentată la timp, atunci își pierde parțial sau chiar toată valoarea.

Chiar dacă specialiștii și utilizatorii împărtășesc aceleași standarde în ceea ce privește valoarea informației, adesea nevoia utilizatorilor de accesa într-un mod liber, dezinvolt și rapid informația este în dezacord cu limitele profesioniștilor. Un exemplu care oglindește această discordanță este: pentru utilizatori o întârziere de zecimi de secundă în calculul datelor poate fi percepută drept un disconfort, dar pentru profesioniștii securității informației, respectiva întârziere este una minoră, deoarece acest timp este alocat altor sarcini precum criptarea datelor.

Conform modelului C.I.A. cele mai importante caracteristici a informațiilor sunt: confidențialitatea, integritatea, disponibilitatea, autenticitatea, utilitatea, acuratețea, nerepudierea și controlul accesului.

Confidențialitatea protejează datele de atacurile pasive, precum interceptarea datelor de către persoanele neautorizate. Se pot identifica mai multe nivele de protecție a acestui serviciu, fie protejarea tuturor datelor transmise de utilizatorii unui sistem, fie protejează doar un utilizator, un mesaj sau chiar unele porțiuni dintr-un mesaj. Acesta din urmă este mai puțin eficientă decât abordarea pe scară largă care adesea este mai complexă și mai costisitoare ca implementare.

Pentru a spori confidențialitatea informațiilor este necesar:

- Clasificarea informațiilor;
- Stocare securizată a documentelor;
- Aplicarea politicilor generale de securitate;
- Educarea posesorilor informațiilor și a utilizatorilor.

Atunci când ne referim la informații personale, fie despre angajați, clienți sau pacienți, valoarea confidențialității este primordială. Persoanele care tranzacționează cu o organizație se așteaptă ca informațiile lor personale să rămână confidențiale, indiferent de tipul organizației.

Integritatea garantează recepționarea mesajelor în forma lor originală, fără ca acestea să fie copiate, inserate, modificate sau rearanjate. De asemenea, asigură protecția împotriva distrugerii datelor și recuperarea datelor după un atac.

Se consideră că integritatea informației este periclitată atunci când aceasta este coruptă". Afectarea integrității este verificată prin modificările apărute fie la dimensiunea fișierului, fie la hashing-ul fișierului. În acest scop, funcțiile hash sunt cele mai des implementate. Și anume, fișierul este citit de un algoritm specific care folosește valoarea biților din fișier pentru a calcula un număr mare unic, numit **valoare hash**. Dacă un sistem de calculator execută același algoritm de hashing pe un fișier și obține un număr diferit pentru acesta, atunci fișierul a fost compromis, iar integritatea informațiilor este pierdută.

Integritatea informațională este piatra de temelie a sistemelor informaționale, deoarece informațiile nu au nici o valoare sau întrebuințare dacă utilizatorii nu pot verifica integritatea acesteia.

Disponibilitatea permite utilizatorilor autorizați – persoane sau sisteme informatice - să acceseze informațiile dorite fără interferențe sau bariere și să le primească în formatul necesar. Accesul la informații se face din prisma a două profiluri: utilizatorii generali și administratorii de sistem, excepție făcând sistemele de operare care echipează calculatoarele personale.

Autenticitatea păstrează starea datelor în forma în care au fost create, plasate, stocate sau transferate. Aceasta poate fi compromisă prin: spoofing și phishing. Spoofing-ul modifică datele care au fost transmise prin rețea, și anume a pachetelor UDP - User Datagram Protocol, care permite atacatorului să acceseze datele stocate pe sistemele de calcul. Iar când un atacator încearcă să obțină informații personale sau financiare folosind mijloace frauduloase, adesea prezentându-se ca o altă persoană sau organizație, ne confruntăm cu phishing-ul.

Utilitatea oferă valoare informațiilor pentru un anumit scop. Chiar dacă informațiile sunt disponibile, dar nu sunt prezentate într-un format semnificativ pentru utilizatorul final, acestea își pierd din utilitate. Un exemplu elocvent al utilității informațiilor sunt sondajele publice.

Precizia (acuratețea) oferă utilizatorului informații care sigur nu dețin modificări eronate ale conținutului și au valoarea așteaptă. Pentru a fi înțeleasă mai bine această caracteristică este exemplificată prin datele de pe cardul bancar – în momentul autentificării în Internet Banking sau efectuării unei plăți, se utilizează numărul cardului CVV-ul sau OTP-ul trimis prin SMS ca token unic de validare a contului.

Nerepudierea împiedică atât expeditorul cât și destinatarul de a nega transmiterea sau recepționarea unui mesaj. Când un mesaj este expediat, destinatarul poate dovedi că mesajul a fost trimis de pretinsul expeditor. Similar, când un mesaj este recepționat, expeditorul poate demonstra că mesajul a fost recepționat de realul destinatar.

Controlul accesului limitează și verifică accesul la sisteme gazdă și la aplicații prin legături de comunicație. Controlul pentru fiecare entitate care încearcă să obțină acces la un sistem se face prin identificare, autentificare ca întru-un final să se livreze drepturile de acces individuale.

1.4. Echilibrul dintre accesul și securitatea informației

În pofida celor mai bune metode de planificare și implementare, nu poate fi asigurată o securitate absolută a informației. Deoarece aceasta este un proces complex, nu un obiectiv este imperios necesară asigurarea unui echilibru între securitate și accesul la informație.

Răspunsul este Da, la întrebarea dacă este posibilă implementarea unui sistem informațional pus la dispoziția oricui, oricând și oriunde prin orice mijloc. Cu mențiunea că un astfel de sistem cu acces nelimitat reprezintă un risc pentru securitatea informaților. Pe de altă parte, un sistem complet sigur nu ar permite accesul la date oricărui utilizator. Pentru a atinge echilibrul în cadrul unui sistem informațional care răspunde la nevoile utilizatorului, cât și ale administratorului sistemului, nivelul de securitate al sistemului trebuie să fie unul accesibil, dar și bine protejat contra amenințărilor. Un exemplu elocvent în acest sens este Microsoft, care, atunci când a fost instigat să obțină o certificare de securitate la nivel TCSEC C-2 pentru sistemul său de operare Windows, acesta a trebuit să înlăture componentele de rețea și să opereze computerul doar de pe consolă într-o cameră securizată [7].

Din cauza problemelor și a preocupărilor de securitate care există astăzi, sistemul informațional și departamentul de prelucrare a datelor sunt complet absorbite în procesul de gestionarea și protecție a datelor. Atunci când nevoile utilizatorului sunt compromise, poate apărea un dezechilibru, deoarece atenția se îndreaptă doar pe protejarea și administrarea sistemelor informaționale. O astfel de situație poate fi evitată dacă specialiștii asigurară informații accesibile cu minime întârzieri sau obstacole. Acest nivel de disponibilitate poate fi atins doar după ce vor fi abordate toate preocupările cu privire la pierderile, daunele, interceptările și distrugerile sistemelor informaționale.

1.5. Securitatea informației: știință sau artă?

Datorită complexității sistemelor de control aplicate, implementarea securității informaționale este descrisă ca un amestec între știință și artă.

Adevărații profesioniști din domeniul securității, cei care prin gestionarea și operarea sistemelor, asigură funcționarea impecabilă a sistemelor informaționale, sunt supranumiți magicieni sau artizanii de securitate (security artisans) [8].

Securitatea ca știință

Orice defecțiune, problemă sau scurgere de informație ce apare în sistem, este rezultatul interacțiunii hardware și software a sistemului. Timpul este cheia soluționării absolute a acestor defecțiuni, dar din păcate adesea specialiștii acționează contra timp.

Problemele care rămân nesoluționate sunt rezultatul defecțiunilor tehnologice din oricare dintre cele o mie de motive posibile. Dar acestea ar putea fi minimalizate, asigurând că informațiile nu părăsesc sistemul, prin diferite standarde de îngrijire, metodele și tehnici de securitate recunoscute și aprobate la nivel mondial, care oferă o securitate solidă.

Securitatea ca artă

Administratorii sistemelor de control sunt aidoma artiștilor care pictează pe pânză. O mică pată de culoare aici, alta acolo, este suficient pentru a crea imaginea de ansamblu pe care artistul dorește să o expună, dar fără a copleși spectatorul său, folosind termeni de securitate, impunerea unor norme fără a constrânge excesiv accesul utilizatorului. Totul este relativ, astfel nu există soluții universale sau mecanisme complet pliabile pe orice problemă, cum nu există nici reguli sigure și rapide care să reglementeze securitatea tuturor sistemelor. În pofida faptului că există numeroase resurse care ghidează procesul de asigurare a securității, toate sistemele trebuie privite în detaliu și abordate într-un mod unic. Iar ținând cont de premisa „Gândim global, acționăm local”, pentru o funcționare garantată a sistemelor de informație, accentul se pune pe securitate.

Securitatea ca știință socială

Înglobând conceptele științei și ale artei, se conturează cea de a treia perspectivă a securității informației – știință socială. Această știință analizează comportamentul entităților în timp ce aceștia interacționează cu sistemele, indiferent dacă acestea sunt sisteme sociale sau sisteme informaționale. Securitatea informației începe și se încheie cu omul din interiorul sistemului, care comunică în mod conștient sau inconștient cu sistemul. Utilizatorii care au nevoie de informațiile pe care personalul de securitate încearcă să le protejeze, reprezintă cea mai periculoasă legătură din lanțul de securitate. Astfel, pentru a reduce riscul cauzat de utilizatori, administratorii sistemelor creează profiluri de securitate diferite, sustenabile și permisive.

Apariția unor probleme provocatoare, impulsionează spre găsirea celor mai optime soluții, rezultând într-un final un sistem informațional mai performant și mai sigur.

Capitolul 2
CRIPTOGRAFIE

2.1. Noțiuni generale

Definiție. ***Criptografia*** – (cuvânt derivat din forma greacă a cuvintelor kryptós și gráfein reprezentând scriere ascunsă) știința care se ocupă cu studiul codurilor și al cifrurilor pentru a asigura confidențialitatea datelor.

Codarea mesajului în timp a cunoscut două subdiviziuni – transpoziția și substituția. **Transpoziția** – dedicată cu precădere mesajelor scurte, în care literele erau rearanjate cu scopul de a crea anagrame. Spre deosebire de cealaltă subdiviziune, **substituția** – una dintre cele mai vechi modalități de codare, și anume prin gruparea alfabetului în perechi de caractere alese la întâmplare pentru ca apoi fiecare literă să fie înlocuită cu perechea ei. Cel mai cunoscut cifru de substituție este codul de deplasare al lui Cezar, în care fiecare literă din mesajul inițial este substituită cu o literă care se găsește în alfabet la o distanță fixă față de cea înlocuită.

După multe secole, în care substituția mono-alfabetică a fost suficientă pentru transmiterea mesajelor secrete, aducerea unor inovații în ceea ce privește complexitatea mesajului codificat a devenit indispensabilă. Astfel, a fost inventat cifrul care utiliza 26 de deplasări pentru scrierea unui mesaj – numit cifru Vigenère. Ulterior a apărut telegraful, urmat de alfabetul Morse, iar în anul 1918 a fost inventată mașina supranumită Enigma, care a făcut din Germania deținătoarea celui mai sigur sistem de informații din lume în timpul celui de al doilea război mondial. După cel de al doilea război mondial, supremația în domeniul criptării revine englezilor, prin dispozitivul Colossus, declarat părintele computerului modern. Iar după anul 1970, Whitfield Diffie împreună cu profesorul Martin Hellman de la Universitatea Stanford, California, creează criptografia asimetrică sau criptarea cu cheie publică.

Atunci când ne referim la siguranța informației, una dintre puținele garanții demonstrabile este criptografia. Cei doi piloni ai securității criptării sunt: *algoritmul de criptare* și *cheia de criptare*. Pentru a fi garantată o criptare sigură și puternică, este obligatoriu ca algoritmul de criptare să fie sigur, iar cheia de criptare să fie nepublicată și de dimensiuni mari.

Conchizând cele de mai sus, algoritmul folosit poate fi făcut public, spre deosebire de cheia de criptare, care trebuie să fie secretă. Fapt ce a dus la propagarea criptării convenționale pe scară largă.

Cerințele fundamentale pe care trebuie să le îndeplinească un algoritm de criptare sunt:

1. Timpul necesar spargerii codului domină timpul de viață al informației (timpul în care informația are valoare);
2. Costul spargerii codului depășește valoarea informației criptate.

Actualmente există două tipuri de sisteme criptografice:

- **Simetrice**, unde cheia este secretă, folosită atât pentru cifrarea cât și pentru descifrarea mesajelor;

- **Asimetrice**, unde există o cheie publică, iar pentru criptarea și decriptarea mesajelor sunt utilizate chei distincte.

Din punct de vedere algoritmic și al domeniului de aplicabilitate, putem defini patru primitive criptografice:

- Algoritmi de criptare cu cheie simetrică;
- Algoritmi de criptare cu cheie asimetrică;
- Funcții dispersive;
- Semnătura digitală.

2.2. Algoritmi de criptare cu cheie simetrică

Preponderent pentru protejarea confidențialității datelor care sunt transmise prin rețea sau sunt memorate în calculatoare, se utilizează algoritmii criptografici cu cheie secretă. Respectivii algoritmi se bazează pe păstrarea caracterului privat al cheii secrete care este folosită la criptarea și la decriptarea mesajului. Cheia secretă este deținută doar de către expeditorul și destinatarul mesajului, pentru ca utilizatorii neautorizați să nu poată intercepta informația codificată.

Securitatea oferită de acest tip de algoritm este direct proporțională cu lungimea cheii și cu posibilitatea de a o menține privată. În momentul în care se încearcă construirea comunicațiilor secrete între numeroși utilizatori, ar putea exista un inconvenient și anume managementul cheilor. Astfel, pentru *n* utilizatori sunt admise *n(n - 1)/2* legături bidirecționale, respectiv sunt necesare tot atâtea chei. Această situație implică confruntarea cu niște probleme dificile la generarea, distribuirea și memorarea cheilor. Atât timp cât cheia secretă are o dimensiune acceptabilă și este suficient de frecvent modificată, este aproape imposibil de a sparge cifrul, chiar dacă algoritmul de criptare este unul cunoscut.

Asigurarea securității criptării simetrice se poate efectua prin respectarea următoarelor condiții:

- Generarea cheilor are loc doar prin utilizarea unor sisteme pseudo-aleatoare de creare a succesiunilor de biți;
- Modul în care se fac cunoscute cheile utilizatorilor ce au drept de acces la informația criptată este unul distribuit;
- Memorarea cheilor se efectuează doar după ce acestea sunt criptate sub o altă cheie de cifru (cheie master) în scopul evitării interceptării cheilor de către o entitate neautorizată.

Inconvenientul respectivului algoritm de criptare este lipsa unor modalități sigure de distribuție, care să emită periodic chei criptografice, deoarece este necesară schimbarea acestora cât mai frecvent.

Actualmente cei mai utilizați algoritmii cu cheie simetrică sunt:

- **DES** – algoritm de tip bloc (block), standard NIST, dezvoltat la mijlocul anilor ’70, care transformă mesajele de 64 de biți în text criptat de 64 de biți. Cheia DES este de 56 biți și nu 64, deoarece 8 biți sunt de paritate, confuzie comună creată de faptul că la fiecare 7 biți există unul de paritate. Lungimea medie a cheii îl face vulnerabil la atacuri de forță brută. Specific acestui algoritm este faptul

că procesarea datelor are loc la nivel de biți. Cea mai nouă versiune a DES-ului este Triple-DES și se bazează pe aplicarea de trei ori a DES-ului, cu trei chei distincte și independente. Acesta este mai puternic decât DES dar, desigur, este și mai lent [9].

- **AES** – reprezintă un cifru bloc (block), care utilizează blocuri de 128 biți și o cheie de 128 biți ca dimensiune. Specificitatea acestui algoritm constă în procesarea datelor la nivel de octet. Spre deosebire de alte coduri asimetrice, algoritmul AES este mai rapid, chiar și decât 3DES, prezintând același nivel de securitate. Utilizarea AES-ului în arhitecturi de securitate contemporane este sigură.

2.3. Algoritmi de criptare cu cheie asimetrică

Algoritmii criptografici asimetrici (cu chei publice) reprezintă începutul revoluției digitale, fiind utilizați în procesul de criptare și decriptare a unor chei distincte, conectate printr-o relație matematică. Punctul forte al acestui tip de relație este că în pofida cunoașterii uneia dintre chei, determinarea celeilalte, din punct de vedere computațional este foarte dificilă. Fiecare cheie definește o funcție de transformare. Respectiv, cele două transformări definite de către o pereche de chei (publică și privată) sunt inverse una față de cealaltă și pot fi utilizate pentru criptarea, respectiv decriptarea unui mesaj. Deci, este irelevant care din cele două chei este utilizată pentru criptare/decriptare. Dacă procesul de criptare se efectuează cu ajutorul primei chei, atunci procesul de decriptare se face cu cea de a doua cheie, iar pentru transmiterea informațiilor codificate se publică doar una dintre chei. Pentru sistemele de autentificare, uzual sunt publicate cheile ce efectuează criptările. Metaforic vorbind, criptografia asimetrică poate fi vizualizată precum un lacăt care poate fi închis de oricine apasă pe el, însă acesta poate fi deschis numai de deținătorul cheii.

Cele două direcții de utilizare a cripto-sistemelor asimetrice sunt:

- **de autentificare a emițătorului și/sau a datelor**: criptarea datelor se efectuează cu cheia secretă a emițătorului, iar cel ce dorește autentificarea datelor utilizează pentru decriptare cheia pereche care a fost publicată;
- **de confidențialitate:** după publicarea cheii, cel interesat să trimită date confidențiale proprietarului cheii publice, va cripta datele cu respectiva cheie, fiind sigur că, doar proprietarul este singurul care le poate decripta.

Cripto-sistemele asimetrice sunt mai lente decât cele simetrice aproximativ de 1000 de ori și au nevoie de o lungime a cheii de minim 2304 biți pentru a obține aceeași performanță a nivelului de securitate oferit de un sistem de criptare simetric cu o lungime a cheii de 128 biți, însă aceștia au avantajul de a elimina procesul de distribuire a cheilor.

Algoritmii asimetrici sunt utilizați la:

- distribuția cheilor în cadrul algoritmilor simetrici;
- la recunoașterea utilizatorului folosind ca atribut al acestuia semnătura digitală.

Principalii algoritmi de criptare cu cheie asimetrică sunt:

- **Rabin** – sistem de criptare echivalent cu procesul de factorizare a unui număr, problemă care este în sine dificilă. Cheile de peste 1024 de biți garantează o securitate puternică.
- **RSA –** cel mai cunoscut și utilizat algoritm cu cheie asimetrică. În principal este utilizat pentru distribuirea cheilor și efectuarea semnăturilor digitale. Pentru o securitate puternică mărimea cheii publice trebuie să fie cel puțin de 1024 biți. Actualmente algoritmii folosesc o cheie de 2048 biți, adică un număr de peste 640 de cifre zecimale.

RSA este recunoscut ca fiind cel mai sigur algoritm de cifrare și autentificare disponibil comercial. Acesta se bazează pe o idee surprinzător de simplă din teoria numerelor și totuși a rezistat, până în acest moment, la toate atacurile criptanaliștilor.

În pofida faptului că este ușor să înmulțești două numere prime mari (*problemă tip clasă de complexitate P*), este extrem de dificil să factorizezi produsul acestora (problemă tip *clasă de complexitate NP – intermediară*). Deoarece RSA derivă din factorizare, se presupune că aparține clasei de complexitate NP - intermediară (*timp exponențial* **Figura 2.1**), despre care se consideră că ar fi „mai ușoară" decât NP- completă (*timp nedeterminist polinomial*), dar mai complexă decât P (*timp polinomial*). [10] Acest lucru permite să se facă cunoscut produsul numelor și să se utilizeze ca o cheie de criptare, deoarece numerele prime nu pot fi reconstituite din produsul lor. Pe de altă parte, acestea sunt necesare pentru decriptarea mesajului.

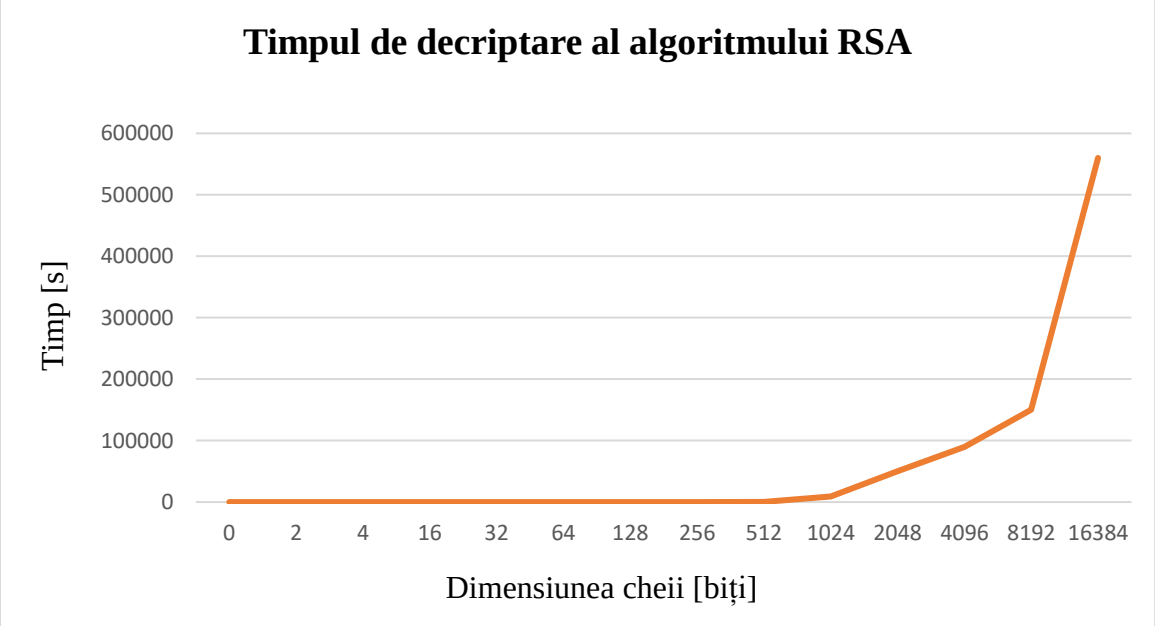


Figura 2.1 Reprezentarea grafică a timpului de decriptare al algoritmului RSA

În continuare sunt prezentate detaliile algoritmului RSA. Fie *p* și *q* două numere prime mari, distincte și aleatoare, care au aproximativ 100 de cifre zecimale fiecare. **N** (1) este produsul acestor două numere, iar *Φ(n)* este **funcția Euler** (2):

$N = p * q$ **(1)**

$\phi(n) = (p-1) * (q-1)$ **(2)**

Exponentul de criptare **e** este numărul care satisface următoarele condiții:

- 1 < e < Φ(n);
- c.m.m.d.c (Φ(n), e) = 1;

Iar exponentul de decriptare **d** este un număr aleator mare care trebuie să satisfacă congruența (3)

$e * d = 1 \left(mod\ \phi(n)\right)$ **(3)**

Mesajul cifrat se obține din mesajul inițial printr-o transformare (codare) bloc. **M** reprezintă blocul mesajului original, M ∈ (0, N – 1).

Mesajul cifrat reprezintă blocul **c** și se obține calculând exponențiala:

$c = M^e\ (mod\ N)$ **(4)**

Iar decriptarea se face prin operația:

$M = c^d\ (mod\ N)$ **(5)**

Pentru funcționarea corectă a algoritmului, cheia publică **(e, N)** și cheia secretă **(d, N)** trebuie să satisfacă următoarea relație:

$M = c^d (mod\ N) = M^{ed} (mod\ N)$ **(6)**

Rezumatul pașilor de efectuare a algoritmului RSA, unde **A** reprezintă destinatarul, **B** – expeditorul (**Figura 2.2**):

1. **A** alege două numere mari prime secrete: **p** și **q**, calculează cheia **N** pe care o și publică;
2. **A** calculează exponentul public **e** pe care îl trimite lui **B**;
3. **B** trimite mesajul **M** către **A**, criptat sub forma unui bloc cifrat **c**;
4. **A** calculează cheia privată **d**, cu ajutorul algoritmului euclidian extins;
5. **A** decodifică mesajul **M.**

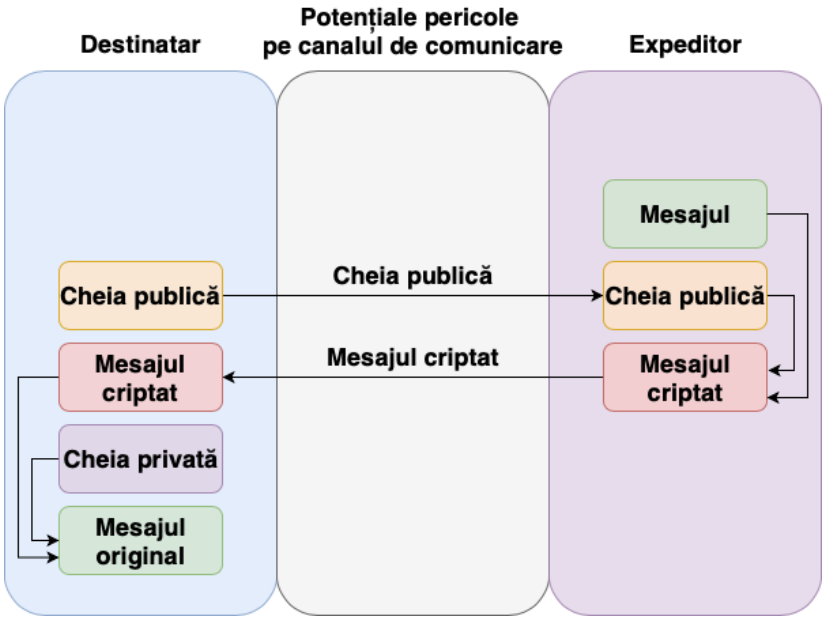


Figura 2.2 Schema de funcționare a algoritmului RSA

În pofida faptului că odată cu apariția articolului lui Boneh și Venkatesan [11], echivalența între securitatea RSA și problema factorizării întregilor a început a fi privită cu scepticism, nimeni nu a reușit să demonstreze ineficiența algoritmului RSA. Conchizând, RSA rămâne a fi unul dintre cei mai siguri algoritmi la ora actuală.

2.4. Funcții dispersive

Funcțiile dispersive, cunoscute și ca funcții hash, au ca intrare un mesaj (șir) de lungime variabilă și generează un șir de control, numit hash sau digest, care are o lungime fixă. Lungimea șirului de control variază în funcție de hash-ul utilizat, dar în general ia valori între 128 și 512 biți. Scopul acestora nu este de a garanta secretul transmisiunii, ci de a altera ieșirea. Astfel, dacă un singur bit de la intrare a fost modificat, atunci acțiunea se răsfrânge asupra ieșirii printr-o avalanșă de schimbări în biți.

Funcțiile hash sunt utilizate cu precădere în următoarele trei domenii:

- asigurarea integrității datelor prin atașarea unui hash generat. Destinatarul poate recalcula hash-ul datelor primite, care este comparat cu hash-ul atașat de către expeditor. Certitudinea că datele trimise nu au fost alterate este validată prin faptul că respectivele coduri coincid.
- sunt parte a procesului de generare a semnăturilor digitale;
- stocarea parolelor. Deoarece, nu se recomandă ca parolele să fie stocate în forma lor originală, o soluție este memorarea hash-ului parolei. Iar atunci când un utilizator introduce o parolă, este calculat hash-ul acesteia, care mai apoi este comparat cu hash-ul stocat. Parola e considerată corectă doar dacă ele coincid.

Cele mai utilizate funcții hash sunt cele din familia MD și SHA - și anume MD5 și SHA-1.

- **MD5 -** funcție de hash, tip bloc, cu o dimensiunea blocului de 512 biți, care a fost dezvoltată de către Rivest în anul 1991. Mesajul inițial este concatenat cu un bit de 1, ulterior este completat cu numărul necesar de zerouri. Ultimii 64 de biți din mesajul preprocesat indică lungimea mesajului inițial.
- **SHA-1** – algoritmul cel mai des aplicat dintre toate funcțiile hash existente din familia SHA. Este utilizat în majoritatea aplicațiilor și protocoalelor moderne de securitate folosite la scară mondială. Acest algoritm produce un hash de 160 biți dintr-un mesaj cu o lungime maximă de $(2^{64} - 1)$ biți.

În pofida faptului că funcțiile hash MD5 [12] și SHA-1 nu mai asigură o securitate înaltă, acestea sunt utilizate în continuare în multe aplicații, datorită eficienței și rapidității lor. Se recomandă utilizarea acestor funcții cu precauție, chiar dacă nu reprezintă un pericol pentru toate categoriile de aplicații. Pentru prima data, atacuri asupra SHA-ului au fost înștiințate în două articole non-tehnice ale lui Schneier cu privire la atacurile asupra funcțiilor hash: *„Cryptanalysis of MD5 and SHA”* [13], *„Cryptanalysis of SHA-1”* [14]. În arhitecturile aplicațiilor contemporane se optează pentru implementarea algoritmului SHA-256.

Capitolul 3
SERVICII BANCARE ONLINE (INTERNET BANKING)

3.1. Noțiuni generale

Dezvoltarea industriei financiare, în ultimele decenii a fost una ascendentă, evoluând de la tranzacțiile bancare convenționale și plăți numerar, la platforme online de tip portofele electronice, unde orice tranzacție se află la doar câteva click-uri distanță.

La începutul anului 2018, Uniunea Europeană și guvernul britanic au anunțat ultimele inițiative în domeniul financiar: a doua directivă a serviciilor de plată a Uniunii Europene (PSD2) și Open Bank. Ambele directive au drept obiectiv perfecționarea standardelor de efectuare a plăților în Spațiul Economic European prin îmbunătățirea vitezei de efectuare a tranzacțiilor, asigurarea unei securități puternice și realizarea schimbului de date între clienți și bănci/firme terțe cât mult mai ușor și eficient. Cu toate acestea, există diferențe sesizabile între ele, fie că se face referință la intervalul de acoperire, perioada de implementare sau la aspectul platformei API.

Se presupune că aceste inițiative realizează mai ușor, mai sigur, mai rapid și chiar mai profitabil conexiuni financiare între clienți și companiile financiare - fie că acestea sunt bănci comerciale majore, agregatori de date financiare sau alți prestatori terți. De exemplu, directiva PSD2 se angajează să includă în industria de plăți entități nebancare, ceea ce face un loc de acțiune mai echitabil pentru diferiți furnizori și clienți. Pe când, Open Banking are ca inițiativă oferirea unor servicii financiare mai convenabile atât pentru clienți, cât și pentru furnizorii de pe piață. Practic, această inițiativă utilizează o interfață de program de aplicație deschisă (API), care permite prestatorilor terți mai mici să își includă și să își prezinte produsele/serviciile pe platformele deja existente. În plus, Open Banking asigură transparența datelor private precum și celor deschise.

Perioada de conformitate avută la dispoziție pentru încorporarea respectivele directive a fost diferită pentru fiecare în parte. Întrucât UE este mai complexă și mai lentă în ceea ce privește adoptarea unor noi norme, agenții financiari li s-au fost impuși ca într-o perioadă de 18 luni să pună în practică aceleași standarde, indiferent de domeniul de activitate. Spre deosebire de UE, firmele financiare din Marea Britanie au încorporat soluțiile Open Banking mult mai rapid, încadrându-se în termenul limită oferit directivei PSD2. Astfel, succesul general a fost unul palpabil în Marea Britanie comparativ cu UE.

Examinând standardele comune și „laissez-faire”-urile celor două directive, conchidem că Open Banking utilizează un singur API. Platforma este convenabilă pentru furnizorii financiari terți care își includ produsele/serviciile pe conturile bancare deja existente, dar evident doar cu acordul clienților. În ceea ce privește PDS2, acesta reprezintă opusul soluției Open Banking. PSD2 se bazează pe API-uri deschise, care sunt definite de piață. Adică nu există o interfață de program de aplicație unificată care să poată fi utilizată în mod universal de către furnizori. Astfel, entitățile financiare individuale sunt nevoite să își creeze propriile API-uri, care de cele mai multe ori, sunt limitate la bazele specifice clienților și în pofida faptului că sunt mai eficiente pentru progresul general, nu au același impact asupra modelului unificat.

Eficiența acestor două directive urmează a fi testată în viitorul apropiat, chiar dacă, mulți analiști financiari, precum și oficiali publici, au declarat deja că inițiativele PSD2 și Open Banking reprezintă următorul pas către o activitate bancară și servicii financiare orientate pe client.

3.2. PSD2

Începând cu anul 2007 directiva originală care vizează serviciile de plată (PSD) a avut drept obiectiv coordonarea și dezvoltarea unei piețe de plată unice în Uniunea Europeană, promovând inovația, concurența și eficiența. Odată cu dezvoltarea comerțului electronic, a portofelelor digitale și a plăților mobile, Comisia Europeană este nevoită să revizuiască PSD-ul inițial pentru a stabili noi limite. În anul 2015, Parlamentul European a aprobat directiva PSD2, care este menită să impulsioneze inovația pe piața europeană și să asigure o securitate mai mare plăților față de versiunea sa anterioară. Noul regulament european pentru serviciile de plată electronică, care a început să intre în vigoare progresiv în perioada 13 ianuarie 2018 - 14 septembrie 2019, implică schimbări fundamentale în industrie, deoarece oferă prestatorilor terților acces la infrastructura bancară și le permite acestora să acceseze conturile bancare pentru a obține date despre clienți, cum ar fi soldurile contului bancar sau istoricul tranzacțiilor, doar cu consimțământul clientului.

Pentru a putea înțelege schimbările aduse de nouă reglementare, este necesară enunțarea participanților incluși la PSD2:

- **TPP** (Third-Party Provider) - Terțe Părți Prestatori;
- **AISP** (Account Information Service Providers) – Prestatorii de Servicii de Informare cu privire la Cont;
- **PISP** (Payment Initiation Service Providers) – Prestatorii de Servicii de Inițiere a Plății;
- **ASPSP** (Account Servicing Payment Service Provider) – Furnizorii de servicii de plată a serviciilor de conturi.

Modificările aduse prin intermediul directivei PDS2 vor avea implicații multiple, unele fiind încă necunoscute, însă cea mai mare provocare este că băncile deschid serviciile de plată către alte companii precum TPP.

Conform directivei, furnizorii terți pot oferi clienților informații agregate despre unul sau mai multe conturi de plată. Soluția respectivă aduce clienților informații imediate și în timp real despre finanțele lor, oferindu-le posibilitatea să gestioneze banii prin intermediul unei aplicații. TPP-urile vor fi nevoiți să respecte și să se adapteze la aceleași reguli ca furnizorii de servicii de plată tradiționale: înregistrarea, autorizarea și supravegherea de către autoritățile competente. Iar pentru a putea opera de oriunde din UE, aceștia trebuie să respecte reglementările din țara de origine. Astfel TPP-urile joacă următoarele două roluri: furnizor de servicii de inițiere a plăților (PISP) și furnizor de servicii de informare a contului (AISP).

PISP - așa numitele instituții de plată care au rolul de a iniția operațiuni de plată, au dus la o schimbare eminentă în industria financiară, întrucât acum sunt folosite doar transferurile bancare (SEPA) și cardurile de plată, ambele fiind oferite de către banca la care este deschis contul bancar. Astfel, nu există prea multe opțiuni prin care să se poată realiza transferul de fonduri într-un cont de plăți

De îndată ce va avea loc autorizarea PISP-ului, comercianții prin acordul semnat de către clienți, vor obține acces la datele contului (**Figura 3.3**). Astfel, cumpărăturile online se vor face direct de la comerciant (precum Amazon), în calitate de PISP, fără a mai fi obligatorie prezența unui intermediar precum: un card fizic sau un alt prestator de servicii de plată (de exemplu PayPal).

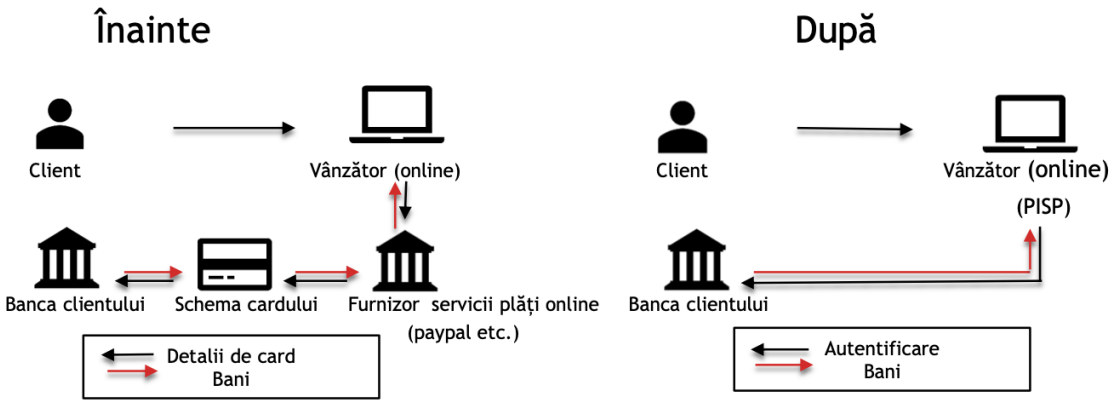


Figura 3.3 Schema de funcționare PISP

Momente importante în cadrul acestui flux:

- Clientul – plătitor trebuie să își dea în mod explicit acordul pentru ca plata să fie executată de către PISP;
- PISP nu deține fondurile plătitorului niciodată;
- PISP se identifică față de bancă și doar în condiții de securitate comunică ordinul de plată;

AISP este reprezentat de către prestatorii care pot accesa conturile bancare și extrage informații cu privire la conturi. Datorită serviciilor furnizate de AISP, clienții care dețin mai multe conturi bancare, vor avea acces la toate informațiile într-o singură aplicație (**Figura 3.4**). Cea mai palpitantă perspectivă a AISP-ului este posibilitatea de a analiza datele și comportamentul clientului în timp real, ducând la eficientizarea tranzacțiilor clientului. Acest lucru este datorat analizei comportamentului clientului și recomandărilor. De exemplu, dacă se ia în considerare un client care economisește, atunci acestuia i se va recomanda contul cu dobânda cea mai avantajoasă.

Momente importante în cadrul acestui flux:

- Pentru ca AISP să poată accesa informațiile care vizează conturile clientului este absolut necesar ca acesta să își dea acordul în mod explicit

- AISP se identifică față de bancă și comunică doar în condiții de maximă securitate cu banca, respectiv cu clientul;
- Utilizarea, accesarea sau stocarea datelor se va face doar pentru prestarea serviciului de informare cu privire la conturi.

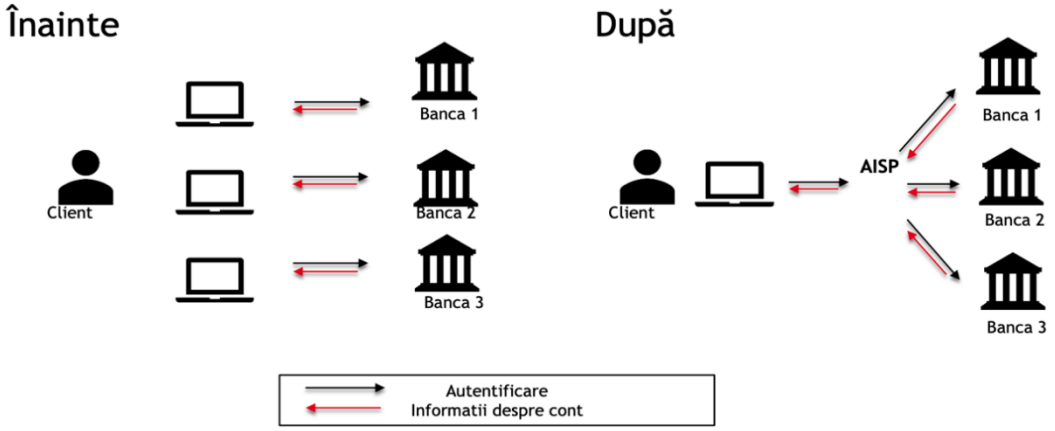


Figura 3.4 Schema de funcționare AISP

PSD2 armonizează și reglementează două tipuri de servicii existente încă de la adoptarea primului PSD în 2007, dar care au devenit mai cunoscute în ultimii ani, și anume, serviciile de inițiere a plăților (PIS) și servicii de informare a contului (AIS).

Serviciile AIS includ colectarea și stocarea informațiilor din toate conturile bancare ale unui client într-o singură aplicație. Acest fapt le permite utilizatorilor să aibă o vedere globală asupra situației financiare și să analizeze cu ușurință cheltuielile și nevoile financiare.

Spre deosebire de AIS, serviciile PIS facilitează utilizarea serviciilor bancare online și anume plățile în mediul online. Respectivele servicii ajută la inițierea unei plăți din contul consumatorului în contul comerciantului prin crearea unei interfețe pentru a pune la punct ambele conturi, completând informațiile necesare transferului bancar (numărul contului, suma tranzacției, mesajul) și informând magazinul de tranzacția ce urmează a fi efectuată. De asemenea, PSD2 facilitează prestarea plăților către furnizorii terțiari care fac parte din aplicația unei bănci, folosind oricare dintre conturile clientului, indiferent dacă acestea aparțin acestei entități sau nu.

Un alt punct forte al directivei PSD2 este implementarea Strong Authentication Client (SAC). Acest mod de autentificare presupune introducerea unor noi cerințe cu privire la politica de securitate, măsurile de control a securității și de atenuare a riscurilor, cu scopul ocrotirii clienților împotriva fraudei și a utilizării într-un mod ilegal a datelor sensibile și cu caracter personal. Pentru efectuarea operațiunilor bancare precum plăți și accesarea conturilor online, după integrarea SCA-ului, etapa de logare se realizează prin intermediul a doi factori de autentificare. În acest sens, băncile au actualizat elementele de autentificare pe care le furnizează clienților, înlocuind coordonatele cardurilor (informațiile scrise pe card precum numărul cardului, data de expirare, CVV), cu codurile trimise prin SMS sau cu jetoane mai avansate, date folosite și în cazul achizițiilor online pentru sporirea securității.

Începând cu ianuarie 2018, intră în vigoare PSD2, chiar dacă au fost semnalate întârzieri în transpunerea directivei în cadrul regulamentului spaniol și în Autoritatea bancară europeană (ABE). ABE a amânat mult timp crearea standardelor tehnice pentru autentificarea puternică și reglementarea accesului TPP. Cu toate acestea, cea mai mare etapă de reglementare a fost autentificarea care a intrat în vigoare la 14 septembrie 2019.

Pe lângă faptul că participă la reducerea complexității din procesul de efectuare a tranzacțiilor, noile serviciile bancare oferă clientului libertatea financiară în luarea deciziilor asumate, bucurându-se de o navigare simplă, intuitivă, dar cel mai important sigură. PSD2 este soluția revoluționară pentru clienții care își doresc să preia controlul asupra finanțelor lor, folosind datele pentru a își analiza cheltuielile.

Înțelegerea datelor privind plățile, le oferă comercianților o nouă perspectivă, ajutându-i să îmbunătățească experiențele clienților, cât și să impulsioneze inițiativele de fidelizare a acestora. De exemplu, prin utilizarea datelor de plată, comercianții ar putea analiza nișa lor de clienți, folosind valori precum frecvența vizitei, traficul locației și cheltuielile medii efectuate. Astfel, ar putea identifica utilizatorii frecvenți, asigurându-le servicii la un nivel înalt, ceea ce ar duce la fidelizarea acestora.

Până de curând, băncile erau unicele instituțiile care livrau clienților informații aferente finanțele lor, începând cu crearea de conturi și emiterea cardurilor de credit, până la împrumuturi bancare și gestionarea economiilor. Între timp au apărut soluții mult mai convenabile, precum companiile fintech, care datorită directivei PSD2, aduc împreună operațiunile financiare efectuate pe diverse conturi bancare, prezentându-le ca un tot întreg. Se consideră că PSD2 va afecta substanțial veniturile bancare. Conform raportului efectuat de către Roland Berger, PSD2 va afecta până la 40% din veniturile industriei bancare europene.

În scurt timp, va fi posibilă utilizarea aplicațiilor terțe care ne vor permite vizualizarea soldului curent, plăților realizate, facturilor și efectuarea cumpărăturilor fără a fi necesară autentificarea în contul bancar. Piața financiară va fi invadată de noi oportunități, care vor duce la conturarea concurenței, dar și la liberalizarea alegerii consumatorului, care va fi influențată de multitudinea inovațiilor și de diversitatea prețurilor.

Acestea fiind spuse, este evident că deschiderea către noile inovații este următorul pas către o activitate bancară și servicii financiare orientate pe client.

3.3. PSD2 în România

Pe data de 13 noiembrie 2019, în Monitorul Oficial al României a fost publicată Legea nr. 209/2019, care transpune directiva PSD2. Iar începând cu data de 13 decembrie 2019, fintech-urile locale s-au bucurat de aprobarea respectivei legi.

Astfel, de la sfârșitul anului 2019, prestatorii de servicii terți români (TPP) au dat start procesului, înregistrându-se la Banca Națională a României. Doritorii dispun 60 de zile pentru a se asigura că contractele aflate în derulare cu dispozițiile titlurilor III și IV (care fac referință la cerințele de transparență, drepturile și obligațiile legate de serviciile de plată) sunt conform noilor reglementări.

Datorită Legii nr. 209/2019, prin directiva PSD2, România face un pas eminent spre servicii bancare deschise. Băncile care au acceptat și au început deja implementarea directivei în România sunt: Alpha Bank, American Express, BCR, BRD, ING, Intesa Sanpaolo, OTP Bank, Raiffeisen și Revolut [15].

Cu scopul de a încredința o protecție mai sigură a operațiunilor de plată, în perfecta concordanță cu directiva PSD2, Legea nr. 209/2019 introduce autentificarea strictă a clienților (SAC) care presupune utilizarea a cel puțin două elemente (2FA – **Figura 3.5**) care fac parte din categoria:

- **Cunoștințelor** (informație cunoscută doar de utilizator) precum PIN-ul, parola, întrebările bazate pe cunoștințe, fraze de acces sau modele de swiping path.
- **Posesiei** (informație deținută doar de utilizator) cum ar fi dispozitivele evidențiate de o parolă de unică folosință (OTP) generată/ primită de pe un dispozitiv (generator token de tip software sau hardware, SMS OTP) sau carduri evidențiate de un cititor de carduri;
- **Inerenței** (informație unică ce reprezintă identitatea utilizatorului) ca de exemplu scanarea amprentei, recunoașterea vocală, examinarea irisului sau a retinei și chiar dinamica tastării sau unghiul în care dispozitivul este amplasat.

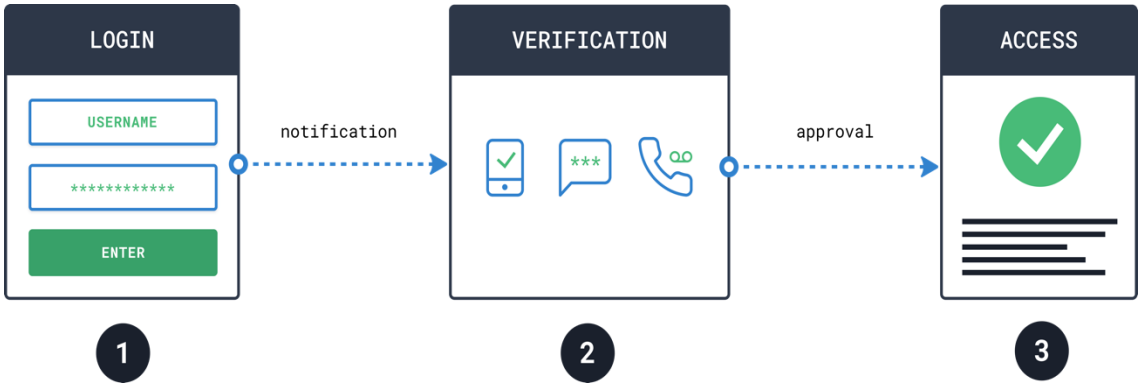


Figura 3.5 Schema de funcționare a autentificării duble (2FA) (imagine preluată de la *[16]*)

Utilizarea noilor instrumente electronice de plată oferă garanție contra fraudei, plățile devenind sigure atât în magazine, cât și în mediul online. Pentru efectuarea plăților sunt obligatorii procedurile de autentificare strictă a clienților (SAC), cu mici abateri de la regulă se realizează plățile de valoare mică sau plățile recurente. Conform regulamentului UE, după fiecare 5 plăti consecutive contactless cu o valoare sub 100 lei/30 EUR, următoarea plată va fi autorizată după introducerea codului PIN.

Pe teritoriul României doar Ministerul Finanțelor Publice și Ministerul Economiei sunt co-inițiatori ai actelor naționale de transpunere, pe când Banca Națională Română este lipsită de dreptul de inițiativă legislativă, neținându-se cont de atribuțiile sale la nivel național.

Capitolul 4
PROIECTAREA ȘI IMPLEMENTAREA SOLUȚIEI

4.1. Structura bazei de date

Proiectarea bazei de date a fost creată conform modelului de date relațional.

Definiție. O bază de date relațională este un tip de baze de date care stochează și oferă acces la punctele de date care sunt legate între ele.

Bazele de date relaționale se construiesc pe prisma modelului relațional, care este o manieră intuitivă și simplă de reprezentare a datelor sub forma unor *tabele,* denumite *relații* [17]. Într-o bază de date relațională, fiecare rând din tabel este o înregistrare ce deține un cod unic numit *cheie*. Cu scopul de a facilita stabilirea relațiilor dintre punctele de date, coloanele din tabel au atribute ale datelor, iar fiecare înregistrare poate deține o valoare pentru fiecare atribut (este posibil ca unele date să fie nule).

Modelul de date relațional a apărut în anii ’70, dar datorită avantajelor sale, acesta continuă să fie modelul cel mai utilizat pentru bazele de date:

- **Simplitatea** – structuri de date omogene și intuitive reprezentate sub forma unor relații tabelare;
- **Normalizarea** – realizată prin descompunerea unei tabele în două sau mai multe tabele;
- **Flexibilitatea** – posibilitatea extinderii modelului fără a afecta datele existente;
- **Integritatea datelor** – asigurată prin constrângerile impuse la nivel de tabelă sau de câmp;
- **Consecvența și precizia datelor** – toate instanțele unei baze de date folosesc aceleași date mereu, datorită multiplelor nivele de integritate ce pot fi introduse în baza de date;
- **Independența logică și fizică a datelor** – indiferent de schimbările efectuate de utilizator modelului logic de date sau de administrator bazei de date, aceste implementări nu afectează programele aplicațiilor care utilizează baza de date;
- **Ușurința manipulării și extragerii datelor** – datorată utilizării limbajului SQL, care furnizează posibilitatea prezentării datelor într-un număr nelimitat de moduri [18].

Pentru reprezentarea logică a structurii bazei de date a fost utilizată diagrama relațiilor între entități prezentată în **Figura 4.1**.

Tabela User conține informații despre utilizatorul aplicației. Acesta în cadrul bazei de date este identificat unic prin intermediul cheii primare *ID* de tipul integer(11). Fiecare utilizator al aplicației are asociat un nume de utilizator unic (username) de tipul varchar(255).

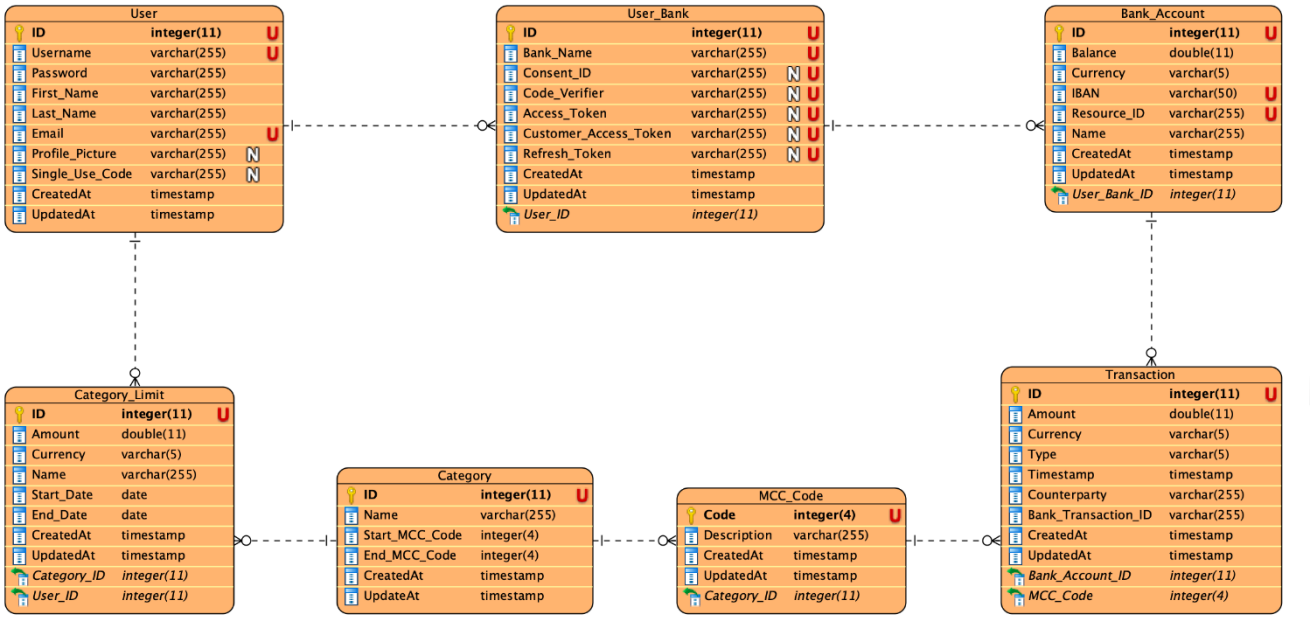


Figura 4.1 Diagrama relațiilor între entități

Pentru conectarea sigură la aplicație utilizatorul definește o parolă care trebuie să satisfacă următoarele caracteristici:

- cel puțin 8 caractere;
- cel puțin o literă mare;
- cel puțin o literă mică;
- cel puțin o cifră;
- cel puțin un caracter special.

Aceasta este salvată în câmpul `password` de tipul `varchar(255)` sub forma unui hash calculat folosind metoda SHA-256. În momentul înregistrării în aplicație, utilizatorul introduce adresa de email pentru recuperarea parolei și pentru a primi codul de validare `Single_Use_Code`, care este folosit în cea de a doua etapă de autentificare. Ulterior înregistrării, utilizatorul are posibilitatea de actualizare a profilului în aplicație cu informații precum numele, prenumele, username-ul, emailul și poza de profil. Acestea sunt reținute în câmpurile: `First_Name` - `varchar(255)`, `Last_Name` - `varchar(255)` și respectiv `Profile_Picture` - `varchar(255)`.

Tabela `User_Bank` include informații caracteristice instituțiilor bancare. Fiecare bancă înregistrată este descrisă printr-un identificator unic sub forma unei chei primare *`ID`* de tipul `integer(11)` și printr-un nume unic salvat în câmpul `Bank_Name` de tipul `varchar(255)`. Ca jetoane de accesare și validare sunt folosiți parametrii memorați în câmpurile: `Consent_ID`, `Code_Verifier`, `Custemer_Acces_Token` și `Accesss_Token` care sunt de tipul `varchar(255)`. Respectivele atribute pot fi și nule, deoarece unele dintre ele sunt specifice doar unei anumite bănci. `Refresh_Token` - `varchar(255)` este utilizat pentru a readuce informațiile de la bancă fără a mai efectua din nou fluxul

de autentificare la bancă. Datele de la bancă, specifice unui user, îi sunt asociate prin intermediul cheii externe `User_ID - integer(11)`.

Tabela `Bank_Account` face referință la datele stocate pe conturi. Fiecare cont este cunoscut prin intermediul cheii primare `ID - integer(11)` și este asociat unei bănci prin câmpul care reprezintă cheia externă - `User_Bank_ID` de tipul `integer(11)`. Restul câmpurilor din tabelă reproduc date specifice unui cont bancar precum: suma – `Balance - double(11)`, valuta – `Currency - enum`, IBAN-ul – `IBAN - varchar(50)` și numele contului – `Name - varchar(255)`. `Resource_ID` este un câmp unic de tipul `varchar(255)` care ajută la identificarea contului în cadrul băncii, utilizat la descrierea tranzacțiilor.

Tabela `Category` include detalii specifice codurilor MCC care oferă informații de valoare referitoare la scopul tranzacției. Câmpul `Name - varchar(255)` indică categoria codului MCC, iar câmpurile `Start_MCC_Code` și `End_MCC_Code` – ambele de tipul `integer(4)` setează limitele fiecărei categorii.

Tabela `MCC_Code` precizează elementele particulare ale codurilor MCC. Fiecare cod este determinat de cheia primară `Code - integer(4)`. Scopul și detaliile unei plății sunt indicate prin intermediul câmpului `Description - varchar(255)` și sunt asociate în mod direct unei categorii prin intermediul câmpului `Category_ID - integer(11)`.

Tabela `Transaction` descrie toate informațiile specifice unei tranzacții. Orice tranzacție în cadrul bazei de date este identificată prin `ID` – cheia primară – `integer(11)`. Tabela include date precum: suma – `Amount - double(11)`, valuta – `Currency - enum`, tipul tranzacției – `Type - enum`, momentul în care este efectuată tranzacția – `Timestamp - timestamp` și destinatarul – `Counterparty - varchar(255)`. Respectiva tabelă este relaționată cu `Bank_Account` și `MCC_Code` prin intermediul cheilor externe `Bank_Account_ID` – de tipul `integer(11)` și respectiv `MCC_Code - integer(4)`.

Tabela `Category_Limit` conține informațiile setate în momentul creării unui buget de către utilizator. Fiecare buget generat este indicat prin identificatorul unic `ID - integer(11)`. Datele setate de către utilizator precum: suma, valuta, numele, data de inițiere și data finală sunt memorate în baza de date pe câmpurile: `Amount - double(11)`, `Currency - enum`, `Name - varchar(255)`, `Start_Date - date` și `End_Date - date`. Un utilizator poate gestiona veniturile sale setând mai multe limite pe categorii diferite, cât și pe toate categoriile împreună. Astfel, tabela `Category_Limit` este relaționată de tabela `User` prin cheia externă `User_ID`, respectiv de tabela `Category` prin câmpul `Category_ID`, ambele fiind de tipul `integer(11)`.

Pentru fiecare înregistrare din baza de date sunt memorate informațiile referitoare la data și ora când aceasta a fost creată (`CreatedAt` de tipul `timestamp`) și când aceasta a fost actualizată (`UpdatedAt` de tipul `timestamp`).

4.2. Tehnologii utilizate

Întreaga aplicație este dezvoltată și înglobată pe platforma **Cloud9 IDE** – un mediu online de dezvoltare integrată care oferă suport pentru mai multe limbaje de programare și permite dezvoltatorilor să schițeze soluțiile mult dorite prin realizarea unui mediu de lucru (workspace) după un template cu un set de instrumente de programare preinstalate. Printre limbajele acceptate de acest IDE se numără: C, C++, PHP, JavaScript împreună cu Node.js, HTML5, CSS3, Java, Python. Cloud9 IDE oferă ustensile pentru debugging, integrare cu sistemele de control (GitHub, Bitbucket etc.), acces la linia de comandă a mașinii virtuale pe care rulează workspace-ul (Linux, Ubuntu) și posibilitatea de colaborare în timp real, fiind astfel o alternativă pentru pair-programming. Principalele avantaje ale acestui instrument sunt configurarea facilă a mediului și accesibilitatea, căci acest IDE poate fi accesat în browser de oriunde și în orice moment prin intermediul unei conexiuni la internet.

JavaScript este unul dintre cele mai uzuale limbaje de multiparadigmă (imperativ, funcțional, orientat pe obiect) din lume [19]. Privilegiul care îl scoate în evidență față de celelalte limbaje de programare este faptul că toate browserele înțeleg și știu să interpreteze JavaScript. La început, JavaScript a fost creat cu scopul de a face paginile web mai interactive și mai pliabile pe alte limbaje de programare (client-side). O mare parte a existenței sale a făcut exact acest lucru, însă, în ultimii ani, JavaScript a evoluat atât de mult încât a ajuns să ruleze nu doar în browsere, ci și pe servere, telefoane sau chiar microcontrolere (server-side) [20].

Node.js este un mediu de execuție JavaScript care a pornit de la interpretorul JavaScript Chrome V8 creat de Google și care este disponibil prin includerea acestuia în cadrul unei aplicații C++ [21]. Această platformă permite rularea codului JavaScript în afara browser-ului, la nivel de server (server-side). Node.js ajută la dezvoltarea serverelor web și a API-urilor cu ajutorul limbajului JavaScript și al unor colecții de module (biblioteci). În general, modulele sunt gestionate de *Node Package Manager* (NPM) și permit accesul atât la fișiere, la operațiile de intrare/ieșire (evented I/O), la lucrul cu bazele de date, cât și la networking. Flexibilitatea oferită de JavaScript, dar și sintaxa sa simplă, concisă și inteligibilă, fac din Node.js un instrument excelent pentru dezvoltarea rapidă a serverelor, bazată pe prototipuri [22].

Express este o bibliotecă Node.js minimalistă și flexibilă care oferă un set robust de funcții pentru realizarea aplicațiilor web și mobile [23]. Acesta, poate fi folosit în dezvoltarea serviciilor **RESTful** sub forma unui middleware care procesează și răspunde cererilor de tip HTTP venite din partea clientului, reprezentând în acest mod legătura dintre backend și frontend.

HTTP (Hyper Text Transfer Protocol) constituie unul dintre cele mai importante și utilizate protocoale web, care aplică următoarele request-uri:

- **GET** - admite accesul la una sau mai multe resurse, fără a le modifica;
- **POST** - permite crearea unei noi resurse;
- **PUT** - aprobă actualizarea unei resurse existente;

- **DELETE** - permite ștergerea unei resurse.

Punctul forte al unui backend RESTful este dezvoltarea rapidă a acestuia și posibilitatea testării independente prin intermediul apelurilor serviciului web care sunt efectuate cu ajutorul instrumentelor software dedicate testării API-urilor (de exemplu Postman).

Sequelize este un ORM – Object-Relational Mapping pentru Node.js care suportă baze de date precum MySQL, PostgreSQL, SQLite [24]. În general, un ORM realizează maparea obiectelor din JavaScript sub forma unor tabele corespunzătoare bazei de date, prin intermediul asocierilor descrise și folosind metadatele din cadrul modelelor. Sequelize pune la dispoziție metode atât pentru realizarea operațiilor de tip **CRUD** (Create-Read-Update-Delete), cât și pentru definirea modelelor, astfel, generând și executând cod SQL specific DML-ului, respectiv DDL-ului. În cadrul acestui ORM, *migrările* sunt folosite pentru a gestiona modificările aduse asupra structurii bazei de date și pentru a evita pierderile de informații. Migrările sunt constituite din două metode reversibile: *up* și *down*. **Up** – implementează modificările datelor, pe când **Down** anulează modificările realizate, aducând modelul la forma sa inițială.

Avantajele utilizării Sequelize sunt:

- Dezvoltatorii se pot concentra pe esența aplicației și nu mai sunt nevoiți să implementeze interfețe între cod și baza de date;
- Reduce costurile și timpul de dezvoltare prin eliminarea codului redundant, respectând astfel principiul DRY (Don’t Repeat Yourself) [25];
- Oferă posibilitatea interogării simultane a mai multor tabele prin realizarea de JOIN-uri între ele.

MySQL este cel mai popular sistem open-source de gestiune a bazelor de date [26]. Modelul relațional de baze de date este un model simplist, cu un suport teoretic foarte stabil, deoarece se construiește pe teoria matematică a relațiilor între mulțimi. Chiar dacă pot stoca structuri de date cu o complexitate ridicată, din diferite domenii, bazele de date relaționale funcționează salvând datele sub forma mai multor tabele, care sunt relaționate între ele prin valorile unor coloane. Fiind și un model simetric, uniformitatea reprezentării datelor, cauzează uniformitatea în mulțimea operatorilor, ceea ce ne permite utilizarea limbajului structurat de interogare **SQL** [27]. Pentru administrarea și vizualizarea bazei de date prin intermediul unei interfețe grafice se poate folosi **phpMyAdmin**.

Vue.js este un open-source model–view framework de JavaScript utilizat pentru construirea atât a SPA-urilor, cât și a UI-urilor. Vue.js este caracterizat de o arhitectură adaptabilă incremental, care se concentrează pe compoziția componentelor și redarea declarativă a acestora. Principala bibliotecă a Vue.js-ului este direcționată pe stratul de vizualizare, pe când funcțiile avansate necesare în cadrul aplicațiilor complexe precum ar fi rutarea sau gestionarea state-ului sunt puse la dispoziție prin intermediul bibliotecilor și pachetelor de asistență, Nuxt.js fiind una dintre cele mai populare soluții [28].

În general, Vue.js permite extinderea HTML-uluii prin intermediul atributelor HTML numite directive. Acestea oferă funcționalități pentru aplicațiile HTML și vin ca directive încorporate sau definite de către utilizator.

4.3. Structura aplicației

Aplicația poate fi descompusă în două mari module: **backend** și **frontend** (**Figura 4.2**). Backend-ul reprezintă partea de server-side, iar frontend-ul pe cea de client-side.

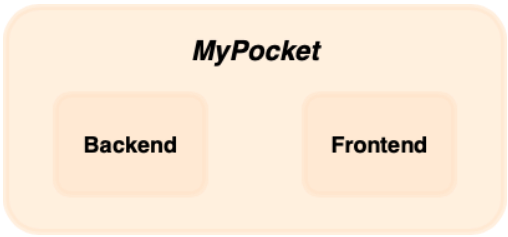


Figura 4.2 Structura generală a aplicației

Pe backend (**Figura 4.3**) avem conturată baza de date care a fost creată și ușor modificată prin intermediul **migrărilor**, populată inițial prin intermediul **seeder**-elor și utilizată prin intermediul **modelelor**.

Incipient, structura bazei de date a fost proiectată mai simplist. Necesitatea efectuării migrărilor care au actualizat structura bazei de date a apărut pe parcurs, mai exact, atunci când structura informațiilor caracteristice băncilor se dovedea a fi diferită față de documentație, din cauza unor înlocuiri sau actualizări ale acestora conform noii directive PSD2. În ceea ce privește seederele, inițial acestea nu au avut drept scop popularea bazei de date cu „garbage data", ci de a completa unele câmpuri cu valori default precum poza de profil a utilizatorului sau pentru ca respectivele date să ajute la definirea unor funcționalități din cadrul aplicației (reprezentarea grafică a tranzacțiilor).

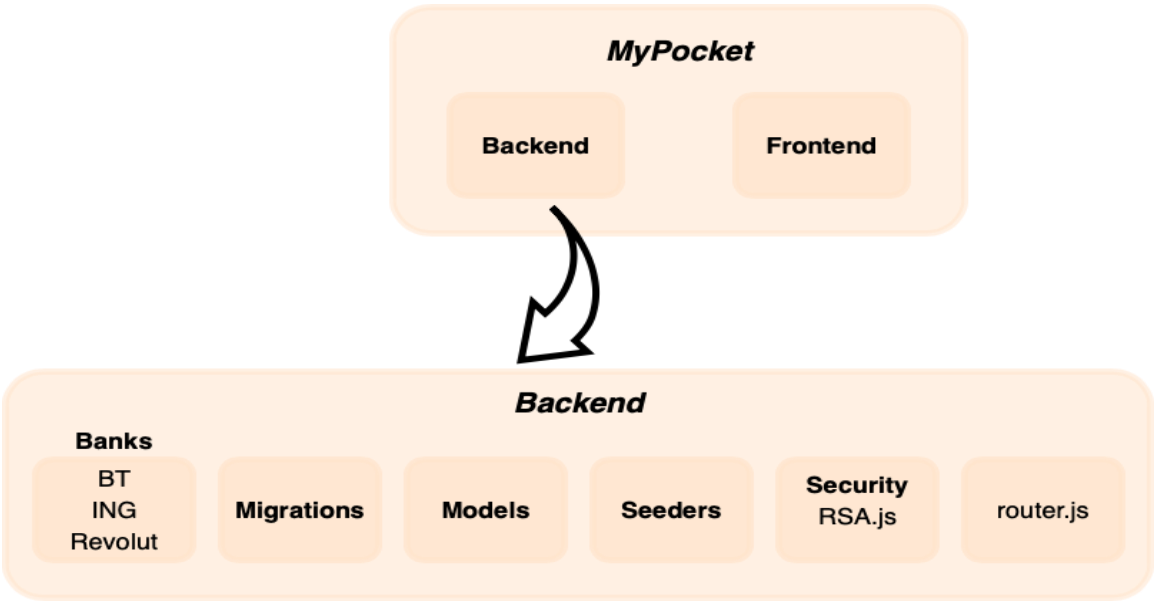


Figura 4.3 Componentele backend-ului

Pe backend sunt definite și rutele băncilor integrate în cadrul aplicației. Pe fiecare rută în parte are loc un întreg flux OAuth care îi oferă aplicației autorizația de a accesa resursele stocate pe serverul băncii. Mai exact, **OAuth** este un open-standard protocol (un cadru de autorizare delegat pentru REST/API) care acordă aplicațiilor posibilitatea „accesului securizat” la serviciile altor sisteme [29].

OAuth se referă doar la autorizare și nu la autentificare. Acestea sunt două concepte diferite: *autorizarea* – cere permisiunea de a acționa, pe când *autentificarea* – demonstrează identitatea persoanei, prin deținerea unor informații specifice (parola/codul de validare). Acest protocol permite aplicațiilor să obțină acces limitat la datele unui utilizator fără a face cunoscută parola acestuia, folosind în schimb jetoane de autorizare pentru a demonstra identitatea între consumatori și furnizorii de servicii.

Ceea ce îl deosebește față de alte protocoale este faptul că OAuth utilizează apelurile API pe scară largă, motiv pentru care aplicațiile mobile, aplicațiile web moderne, consolele de jocuri și dispozitivele Internet of Things (IoT) găsesc OAuth ca fiind o experiență mai bună pentru utilizator.

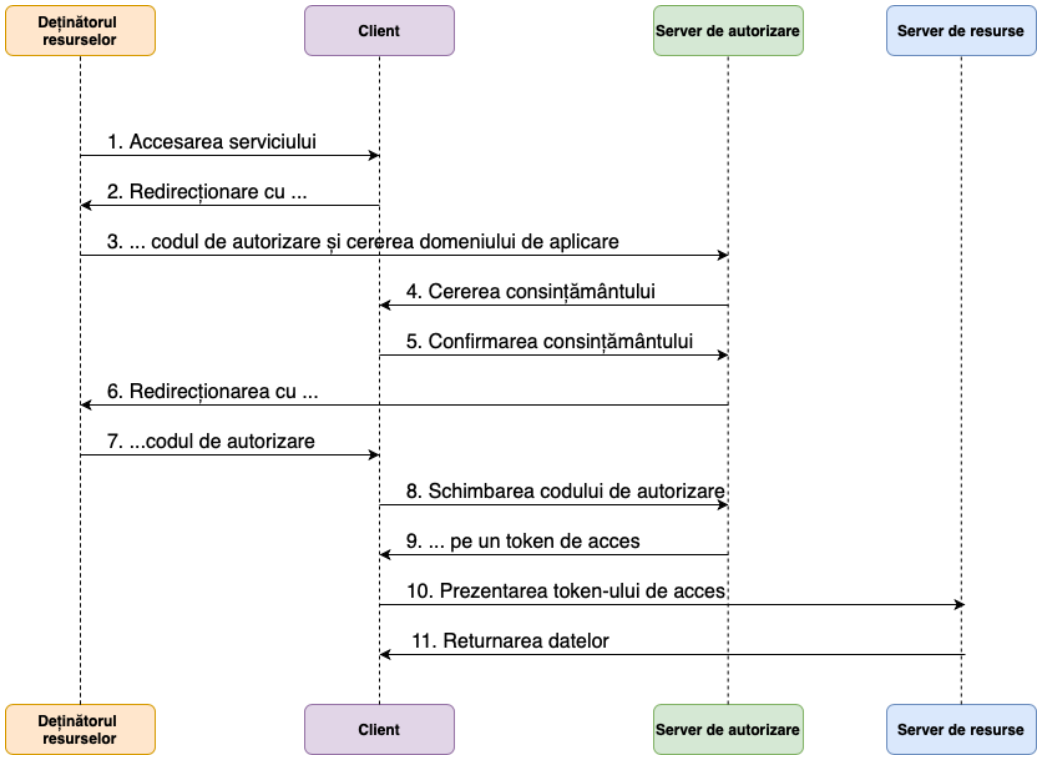


Figura 4.4 Fluxul OAuth 2.0

Fluxul de autorizare într-o implementare tipică OAuth 2.0 (**Figura 4.4**) poate fi descompus în următoarele etape:

- Aplicația solicită autorizarea utilizatorului;
- Utilizatorul autorizează aplicația și livrează dovada;
- Aplicația prezintă dovada autorizării pe server pentru a obține un Token;
- Token-ul este restricționat să acceseze numai ceea ce utilizatorul a autorizat;

Token-ul de acces (**Authorization Code**) este jetonul pe care clientul îl folosește pentru a accesa serverul de resurse (API). Acesta este menit să fie valabil pe o perioadă scurtă de timp: ore sau minute și nu poate fi revocat. **Refresh Token** este jetonul de actualizare, folosit pentru obținerea noilor jetoane de acces. Acesta are o valabilitate mai mare: zile, luni, ani și poate fi revocat. De fiecare dată când token-ul de acces este actualizat, un nou cod de autorizare semnat criptografic este trimis [30].

Au fost menționate două fluxuri diferite: **obținerea autorizației** și **obținerea de jetoane**. Acestea nu trebuie să se întâmple pe același canale, ci pe 2 canale diferite: frontend și backend, așa cum se întâmplă și în cadrul aplicației. Frontend-ul descrie ceea ce se petrece pe browser. Pentru început, browser-ul redirecționează utilizatorul către serverul de autorizare, pentru ca utilizatorul să își dea acordul. Odată ce utilizatorul obține subvenția de autorizare, o înmânează aplicației, iar aplicația finalizează fluxul OAuth pentru a obține token-urile. Jetoanele sunt destinate a fi consumate de către backend. Backend-ul realizează un apel HTTP către serverul de resurse pentru a schimba grantul de autorizare pe jetoane [31].

Toți pașii de mai sus vor fi explicați și detaliați prin intermediul fluxurilor realizate pentru obținerea informațiilor de la băncile din cadrul aplicației: Banca Transilvania, ING și Revolut. Toate cele 3 bănci oferă acces la resursele stocate pe server, însă acestea o fac într-un mod diferit.

Vom începe cu **ING**, care are o structură mai complexă și ușor diferită în comparație cu celelalte bănci.

Pasul 1. Obținerea certificatelor și a cheilor publice

Atât certificatele cât și cheile publice ale acestora le descărcăm de pe site-ul oficial al băncii și le integrăm în cadrul aplicației sub forma unui agent https, exemplificat în **Anexa 1.1.**

Pasul 2. Solicitarea token-ului de acces

Se efectuează un request de tipul ***POST*** la endpoint-ul sandbox-ului băncii: „*https://api.sandbox.ing.com/oauth2/token*”. Pentru apelarea respectivului endpoint, trebuie creată o semnătură cu care se validează request-ul HTTPS. Crearea semnăturii se face prin intermediul a doi parametri calculați: *date* și *digest*. Semnătura este calculată dinamic pentru fiecare request, deoarece valorile care sunt utilizate ca parametri de intrare se modifică de fiecare data atunci când aceasta este generată. **Anexa 1.2.**

Request-ul care solicită jetonul de acces are următorii parametri: *calea endpoint-ului apelat, digestul, data, TPP-Signature-Certificate, keyId* – numărul de serie al certificatului codat în hexa, *semnătura* și *Content-Type* – care forțează parametrii să utilizeze codarea URL de forma x-www-form-urlencoded.

În răspunsul request-ului sunt trimiși parametrii: *access_token* – jetonul de acces al aplicației, *expires_in* – timpul de expirare al token-ului de acces exprimat în secunde, *scope* – lista tuturor scopurilor în care se permite accesul prin respectivul, *token_type* – tipul token-ului și *client_id* – identificatorul unic al clientului.

Pasul 3. Solicitarea codului de autorizare și a token-ului de acces al clientului

Acest pas este necesar, conform directive PSD2, pentru a obține consimțământul clientului (acceptul ca respectivul API îi utilizează datele cu caracter sensibil).

Folosind jetonul de acces al aplicației, obținut în etapa precedentă, este inițiată o cerere de consimțământ a clientului și este generat un cod de autorizare. Apoi, respectivul cod de autorizare este folosit pentru a solicita token-ul de acces al clientului (**Figura 4.5**) care urmează a fi folosit ulterior în cadrul API-urilor PSD2.


```
{
   "data": {
   "access_token":
       "eyJhbGciOiJkaXIiLCJlbmMiOiJBMjU2Q0JDLUhTNTEyIiwia2lkIjoidHN0LTRjOGI1Mzk3LTFhYjgtNDFhOC1hNTViLWE3MTk5MTJiODNkMiIsImN0eSI6IkpXVCJ9..
       8V5hXf_5S68yRVsyIJ1XVA.
       tXFaHiWJ9vV7lBKqny4qne4gi09zMVxlQd4ch6n_Q3ywpxSh8uE4PFvLcfxK_NzN6O_SBp0qetjS3mlo5hgFww_jgrjHEAivzVBZ8np7A-7WwDnH7FwkzMpHX2Wwjck7rT2NJ1u81
       PrGNvH-QlrUcTQNKZZuyCAbTgEF_giz2HBP8RE2tyHLywRG6J4MM8GW8eIBW6IctxnKyoJMYWGiBssPbaerxynoXgHU2PtSovTnfaCvbVb-IqNp7C_Ojhf_2DSoahiz6l9dI9bgLs
       hn3Y15p8Fb3oE9ETF56KeQ0tsQuudSEwnobskaW8FQOkPVhx8QO2EwjKmqtR8ZZKLAVXMsbN6hPzl1KTsm4t5Bd4BqFlzlO82D7Zu3qV4DVFiBg5WP9hjnsPBVFaE3UFz2_Pk0eyj
       u74NZOnG2rsQp2SfBlG8FUzmJPpeqx-LFIw0l4XBeYeoFe-bDT6OBHNmBMEXHew7TEPavfoe7myP_PhB6QIThiiRTsHGLT9wQZBmp1bqk-sbBOXn3BgbWOaO3P0uR450JOVoQhbDf
       oIr2WtqIDTbNR8_PWKvoRF21zsYE22F8MZM4i9HNjhMBeTNvPDfybTvvlCWCSc7O3hCVCEiZvG5mlPRfOU5gHzvxOofNuxW-GA_zPscBhE2UoFNbI7-hyByQp6ODTgt14tcZRZstu
       tX2I2QKAAUYfF_1KJjbXp1YKdl6Vx3TNvoT-4G5fn0wk0KQkJBzug8M-hXkpxCNiSnnzGBSfE-wNYV8RyQfNgZjV5YuZjPn96SK7ohMgNA7_NlpKY6vrHa5edZV8WKB9tSW-RVBXX
       p9PVKGEjpyYV75w3gx__grWGrLQg9nI6wZZFRt-gPmXQuw6x8UzHihWOZnpzvxNHfl1tVjwgiZwieLdylqNH5QkeKALOMIpefYMt4Eligb-4-sIskX8_OXy6Bkh5J9gQVNXvyiQBe
       gFCIvHz5ISxO9pBK2YcIAXBABZFB2vMCTXtggGeJADZUSVH0Nt9SsSwqb7-7y8z0ZZoA2tf615gWRq8R62Y_SiGmKiWT7KXKoq0JY0dm9W3HuN-KoUrFqFGLx9U2zS-8fRU3KnWiO
       e8lXhwYu3fDn6AIxe6RilNJtvfd5H6Q66flpnVkj5YNjKtVCzaXJqA8MCgpQHathHAunEsxQKSkvO0s4HKhHNMYRNwwGWZ7jWn4nCaMLVfl2_I_FGqulQY4eF5q7yT3MsxJJD8bch
       nj7Ceo2-a52PPSrBxYMncD8yf5R7ScjiD-SGAzEroUGNZWQx8PDwDbiU-JFBlS2rrcI1nkT8lC-BZ60qVJXrlCR0xUlNMPmEJf-NXMayxcdYqe4Ax1Xjj_UXtIXbkZNtpjzT2O6YS
       OalPNgduHcrMmv9SJfiMNPFZT7E21C46Ks-PGiMJfxWUwvAGnZVyGmJUFItNk0h7U1vDNhwzyukzBPLyRZiiOFnIdFsMXmCGSHAvD4OD9zol8CAXTRi-kbXhp-s6neO1gTTsoXJip
       JJReIKzdy1ZwIrW5uRkE0RoOhk7kehh9Pf2-fcRQzxgEih1qNNZf_cz9Dp0sDcH9pVAD8qz74enltwChjLBEfvkhS5Cw40TrKefkExOE2NgxRFx3VbmvL7s0ysuw1MTBsmHhdZ5DU
       pissO9sWfOKhQzdPCE3QyfC0fLz2m3gwJ4ZPQBF6irfDS3koiNMxjUb0kWtYYkBZTdXcuF4w8yUxOqZrepz9yZtXdZHBOj2K9K6pRuTCJRCb9xEs8DLfGzn3SW_UXC71GGmDYep5y
       qy-u6tefwYFz_KdY2V6Y0vEhlD-nS-rKKNI2yjWNlO5PzvD-ycfJkbgX0tD3IQ8UB9bs_EQkjUvjinUQ5rMAbCrDFgoWXzVLctdZ7oLKc88fc_2QibnUnPlYkPNch_jM1ZAXBCWbC
       wy8Hm9LoIEirk83sdm7PxOyxl4XxDJy1jo7VjdVnh8u9CV9Jln1j_dp2KgnuXHVII_dJ0hXKXUn_IERgajpQVAQ-LjD19uaX7L2dcsdAFbmgQcQNBkFotuxW1MGaAOOSA_ZOc8gEL
       fFn0n3x5pAY__VFZxQsjvdClZoQ4wszvKUxEh0D2gQ2zOCECJQBH18ZUmBG6YikRzgFv-JIrshtX_UbOB-7OO4NYmBI44Oi2UbcQBoNkLgz94Oy6apXMiqSAZgH_0x1qQf2AkmjWC
       blJKL4PvFMCnSQ84ANIUNGayEiqTKulO0a0tthvfYkuRxGLYjGVqKxvB0ui9AWbQ8_Ad0Ucc1OtwhPukxSUy-P1RMEi3YIZ4MPbVqyGLu3Pc9Ac_.
       mL3EmIpXKNOcPDlyfyomBZcVLfIcNoV8CxLv9VhMOXY",
   "expires_in": 905,
   "scope": "payment-accounts:orders:create granting",
   "token_type": "Bearer",
   "client_id": "5ca1ab1e-c0ca-c01a-cafe-154deadbea75"
 }
}
```


Figura 4.5 Răspunsul request-ului de solicitare a token-ului de acces în cadrul băncii ING

Un nou request este inițiat, doar că de data aceasta unul de tip ***GET***. Parametrii folosiți sunt: *calea endpoint-ului apelat, digest, data*, *authorization* – de forma Bearer <access_token>, k*eyId* – client_id obținut în urma request-ului de obținere a token-ului de acces, *semnătura* și *Content-Type*.

Răspunsul request-ului conține URL-ul aplicației de autentificare în cadrul băncii ING. Întrucât, aplicația simulează un mediu de producție, adresa URL redirecționează către sandbox-ul băncii, unde se va solicita mai întâi țara clientului, apoi selectarea unui client cu un scenariu de test specific.

Pe URL-ul redirecționat este efectuat un nou request, de tip ***POST***, care lasă clientul să autorizeze cererea API-ului și care are ca parametri: *client-id*, *scope* și *redirect-uri* – adresa URL către care este redirecționat clientul după autorizare.

Odată ce etapa anterioară este încheiată cu succes, aplicația de autorizare a ING redirecționează clientul către adresa URL furnizată în parametrul de interogare *redirect_uri* împreună cu codul de autorizare. Parametrul *code* conține codul de autorizare generat dinamic pentru profilul de test selectat. Acest cod de autorizare este utilizat în pasul următor pentru solicitarea unui jeton de acces al clientului.

Acces Token este unul dintre parametrii primiți ca răspuns la un ultim request în cadrul acestui flux. Respectivul request este de tip ***POST*** și pe lângă jetonul de acces, în răspuns mai aduce informații ca: *expires_in* – timpul de expirare al token-ului de acces exprimat în secunde, *refresh_token* – jetonul utilizat în cazul în care expiră token-ul de acces, *refresh_token_ expires_in* – timpul de expirare al token-ului de actualizare.

Pasul 4. Apelul API-ului

Ultimul pas din cadrul acestui flux se efectuează fie cu token-ul de acces al aplicație, fie cu token-ul de acces al clientului.

La **BT** etapele din cadrul fluxului OAuth sunt ușor simplificate.

Pasul 1. Înregistrarea unui furnizor terț de servicii (TPP) ca client OAuth2

Pentru accesarea informațiilor despre contul unui utilizator, un furnizor terț trebuie mai întâi obținut consimțământul utilizatorului. Aceasta se face printr-un request de tip ***POST*** la endpoint-ul: „*https://api.apistorebt.ro/bt/sb/bt-psd2-aisp/v1/consents*". Respectivul request are parametrii: *access, recurringIndicator* – dictează dacă acordul generat va fi folosit de mai multe ori, *validUntil* –până pe ce data va fi valid acordul, *combinedServiceIndicator* și *requencyPerDay* – frecvența maximă de solicitare pe zi pentru un acces fără implicarea PSU-ului. Răspunsul request-ului aduce furnizorului TPP înregistrat informații precum: *client*_id – ID-ul clientului, *client*_secret – secretul clientului, *redirect_uri* – adresa URL către care este redirecționat clientul după autorizare, toate acestea conform standardului OAuth2.

Cu datele primate ca răspuns este construit un URL pe care TPP îl utilizează pentru redirecționarea utilizatorul către pagina de autentificare BT. Utilizatorul folosește acreditările BT pe internet pentru a se autentifica și a oferi consimțământul asupra listei de conturi și acțiuni ce pot fi efectuate pe respectivele conturi (lista de tranzacții, soldurile , informații despre cont – care vor fi oferite de server).

La finalul fluxului de autentificare și de oferire a consimțământului este generat un jeton pentru TPP, care conține parametrii: *access*_token, *token*_type, *refreash*_token, *expires_in*. Cât despre utilizatorul, acesta este redirecționat către redirect_uri.

Pasul 2. Apelul API-ului

În urma fluxului OAuth 2, jetonul de acces va fi schimbat pentru un cod de purtător, care va fi utilizat în toate apelurile ulterioare.

Revolut este banca cu arhitectura cea mai simplă, în ceea ce privește accesarea informațiilor. Pentru a fi asigurată securitatea clientului când acesta accesează contul, la inițierea fluxului standard de autentificare OAuth2 este utilizat JSON Web Token (JWT).

În general, mecanismul JWT necesită utilizarea unei chei private / publice pentru a semna cererea token-urilor. Crearea unei perechi de chei publice / private o putem face utilizând OpenSSL.

Cheia publică odată încărcată în cadrul API-ului, duce la generarea unui certificat „publickey.cer" care semnat cu token-ul JWT permite solicitări către API și totodată furnizează parametrul *client_id*, care la rândul lui permite construirea URL de redirecționare OAuth, utilizat după etapa de autorizare.

Autorizarea aplicației se face în interfața web, apăsând butonul de: „Authorise API access", care are ca acțiune definită redirecționarea către o pagină în care trebuie introdus codul de autorizare. Odată introdus, acest cod de autorizare este dat contra schimb pe jetonul de acces și pe jetonul de reactualizare. Acesta din urmă, devine activ doar atunci când jetonul de acces a expirat, adică după 40 de min.

Astfel, în cadrul întregului flux, efectuăm doar un singur request de tip ***POST*** care are ca parametri de intrare: *grant_type, authorization_code, client_id, client_assertion_type, client_assertion* și care ca răspuns aduce: *access_token, token_type, expires_in și refresh_token.*

Funcționalitatea de aducere a conturilor în cadrul aplicației la fel a fost creată pentru fiecare bancă în parte. Însă, spre deosebire de fluxul de autorizare, prezentarea conturilor, în principiu funcționează la fel pentru toate băncile: un request de tip ***GET*** pe *endpoint-ul băncii*, plus calea: */accounts*. Singurele detalii diferite sunt unele headere particulare fiecărei bănci în parte. Datele din răspunsul request-ului sunt salvate în baza de date pentru siguranța aplicației.

Idem, funcționalitatea de aducere a listei de tranzacții specifice fiecărui cont bancar, a fost creată pentru fiecare bancă, dar modul de funcționare nu prezintă deosebiri însemnate de la o bancă la alta. Se efectuează un request de tip ***GET*** pe endpoint-ul creat prin alăturarea: *http hostului, calea către conturi, id-ul* contului și calea finală – */transactions*. Diferențele minore de la o bancă la alta sunt parametrii din header-ul și corpul request-ului. Forma răspunsului este una diferită la fiecare bancă, astfel acesta trebuie prelucrat mai în detaliu pentru a obține în linie generală aceleași tipuri de date. De exemplu, pentru a dobândi counterparty la banca ING, un câmp al răspunsului trebuie prelucrat prin intermediul unui regex particular în funcție de tipul tranzacției (IN/ OUT).

O altă parte importantă a aplicației este partea de frontend-ul (**Figura 4.6**). Aici stocăm toate datele și metodele care conturează forma aplicației. Frontend-ul aplicației este construit din mai multe foldere care sunt definite ca componente de bază ale aplicației, precum componenta de *Login* – permite utilizatorului să se logheze în aplicație, *Register* – permite utilizatorului să își creeze un cont pentru aplicație, *Profile* – aici sunt stocate toate datele despre utilizator, iar unele dintre ele pot fi modificate, *Accounts* și *Transactions* – aduc informații despre conturile bancare și tranzacțiile utilizatorului.

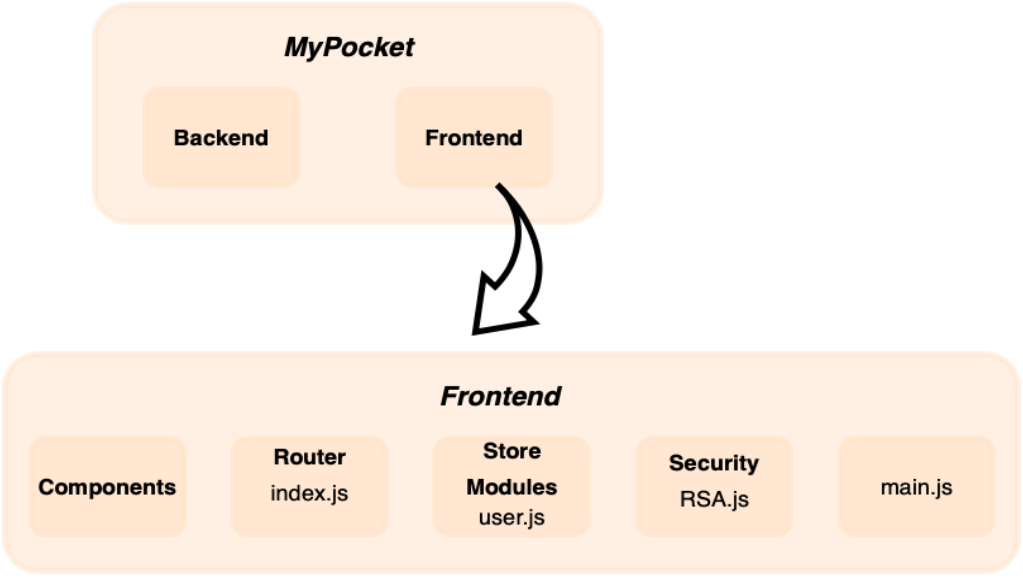


Figura 4.6 Componentele frontend-ului

Pe acest canal mai avem stocate și fișiere dedicate băncilor. În respectivele fișiere sunt definite request-uri care creează URL-urile de redirecționare după etapa de autorizare din cadrul fluxului OAuth.

Două componente importante ale părții de frontend care necesitau a fi implementate odată cu alegerea framework-ului Vue.js sunt ***store*** și ***router***. Folosirea Vue.js-ului implică deja compunerea unei aplicații cu componente definite și predefinite, iar pentru rutarea acestora este necesară folosirea router-Vue-ului.

Pentru o funcționare perfecta a aplicației, toate componentele trebuie să fie mapate corect și atent, ca apoi router-Vue-ul să fie anunțat unde să redirecționeze componentele.

Router-Vue a eliminat dezavantajul aplicațiilor SPA: incapacitatea de a partaja link-uri către pagina care este definită mai jos, în cadrul unei anumite pagini web. SPA-urile servesc utilizatorilor un singur răspuns bazat pe URL de la server, iar marcarea anumitor ecrane, cât și partajarea linkurilor către anumite secțiuni specifice este dificilă, adesea imposibilă. Însă prin pachetul „vue-router", calea curentă poate fi modificată prin intermediul unor evenimente declarate (clickuri pe anumite butoane sau link-uri).

Pe baza aplicației dezvoltate, o mulțime de exemple ar putea fi oferite: butoanele în cadrul panoului principal de prezentare (*Add a New Account, View All Accounts, View All Transactions, Create a New Budget*) sau link-urile înglobate în hedereul aplicației (*myPocket, My Profile*). Odată creat un eveniment (la click sau doar la trecerea cu mouse-ul peste element), user-ul este redirecționat către o altă pagină, de exemplu, la apăsarea butonului de *View All Accounts*, utilizatorul este direcționat pe pagina unde sunt prezentate toate conturile sale cu detaliile aferente fiecărui cont.

Codul aferent ecranelor exemplificate în **Figura 4.7** se regăsește în **Anexa 2**.

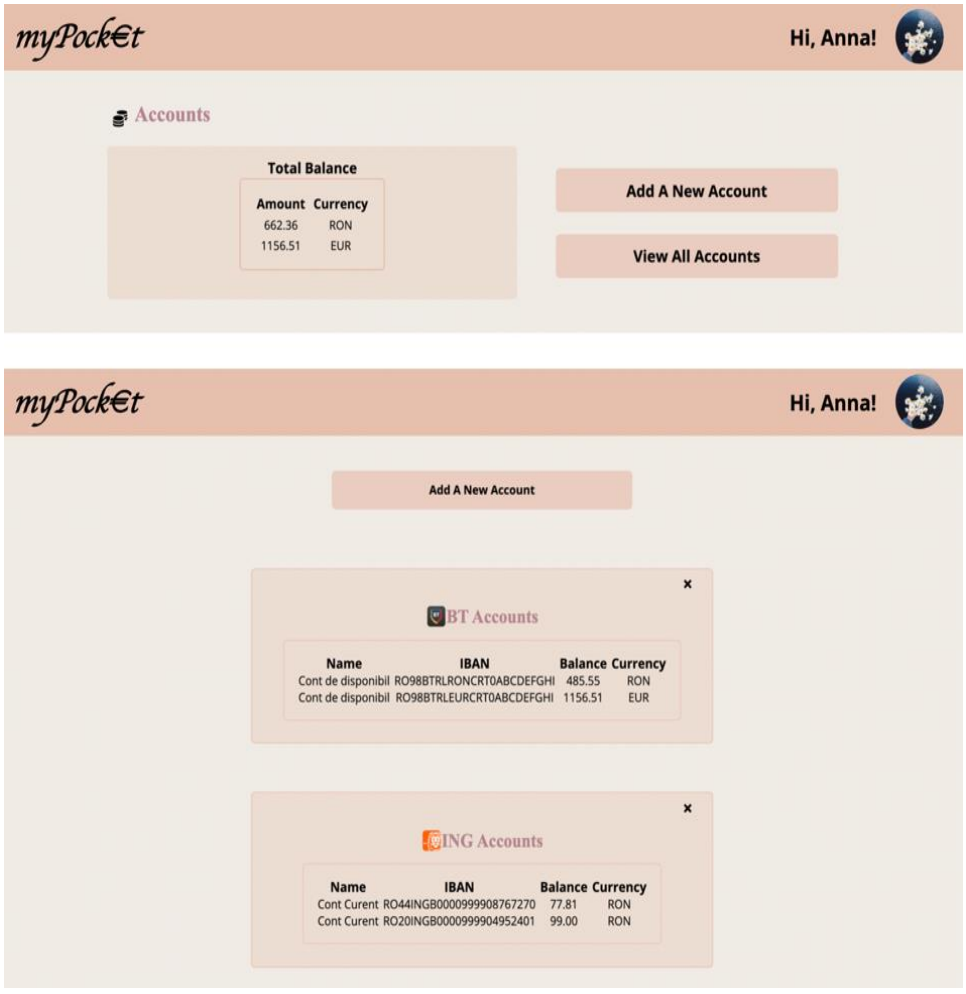


Figura 4.7 Captură de ecran care demonstrează rutarea componentelor în aplicație

Soluțiile web implementate cu Vue.js implică utilizarea **Vuex**-ului – o librărie și un model de gestionare a state-ului (state management pattern). Acesta este întrebuințat precum un magazin

centralizat pentru toate componentele din cadrul aplicației și definește anumite reguli pentru asigurarea faptului că starea (state) componentei nu poate fi mutată decât într-un mod previzibil [32].

Orice componentă este conturată prin trei concepte de bază (**Figura 4.8**):

- ***state*** (starea),
- ***view*** (vederea grafică a stării) și
- ***actions*** (modalitățile prin care este posibilă modificarea stării).

Prin definirea și separarea conceptelor și aplicarea regulilor de menținere a state-ului și a view-ului ca două concepte independente, oferim codului mai multă structură și mentabilitate.

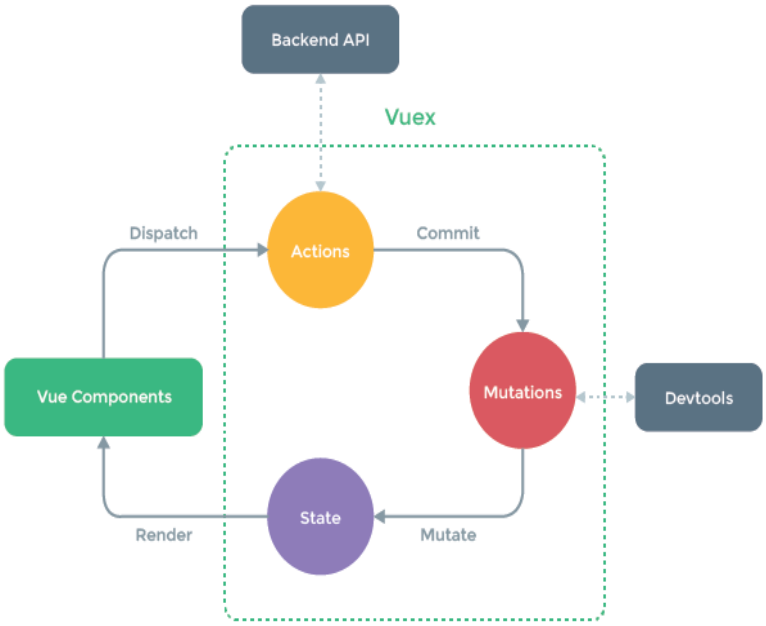


Figura 4.8 Reprezentarea fluxului unic de date (figură preluată de la *[32])*

Vuex folosește un singur arbore de stare - adică un singur obiect conține toate stările la nivel de aplicație. Acest lucru implică existența unui singur magazin – **store** pentru fiecare aplicație. Un arbore unic localizează mai rapid o anumită bucată de stare și permite în mod facil modificarea stării actuale în scopuri de depanare. Pentru evitarea conflictelor, state-ul și mutations-urile sunt împărțite în module.

În cadrul secțiunii **store** a soluției web prezentate sunt stocate toate modelele utilizate în aplicație. Prin intermediul modelului *user.js* este creat obiectul de **user** care este folosit activ în toată aplicația.

Inițial în state au fost declarate toate atributele unui *user* și setate cu valori default. Doar utilizând **mutations**, **state-**ul este modificat în store. Pentru a identifica cu adevărat când starea utilizatorului trebuie schimbată se folosesc **getters**, care sunt reevaluați doar în momentul în care unele dintre dependințe sunt modificate.

În aplicație au fost declarate mai multe mutații: *loginUser*, *setUser*, *logoutUser*, *addError* și *clearErrors*, toate fiind sincrone. Iar pentru a evita conflictele de genul: care mutație trebuie să schimbe starea, atunci când sunt apelate asincron mai multe metode, au fost declarate și niște **actions-uri**: *register, login, codeVerification, resendCode, getUserData, clearErrors* și *logout.* Astfel, la apelul unei anumite acțiuni, aceasta comite una sau mai multe mutații.

În **Anexa 3** se poate observa explicit declararea acțiunii **login.** În structura sa, aceasta conține mai multe mutații: *clearErrors*, *setUser* și *addError*. Dar fiecare mutație va fi comisă doar când este apelată respectiva acțiune și dacă se evaluează cu adevărat expresiile condițiilor din sintaxa *if*.

4.4. Algoritmi și metode de protecție

Fiind o aplicație în care are loc stocarea și prezentarea informațiilor cu caracter sensibil este important să se asigure o protecție maximă împotriva atacatorilor. Astfel, la etapa de autentificare în aplicație, mai exact la validarea codului de posesie este utilizat algoritmul de criptare **RSA**.

Pe email-ul consemnat de utilizator la etapa de înregistrare este expediat un cod unic și aleator de 8 caractere exprimat în hexa obținut prin metoda **randomBytes()** din librăria **crypto**.

singleUseCode = crypto.randomBytes(4).toString('hex')

Odată generat, codul de unică folosință este expediat către utilizator și totodată memorat în baza de date. Ca mai apoi, codul introdus de către utilizator, în câmpul destinat din cadrul formularului de validare, să fie criptat prin intermediul algoritmului **RSA**. Codul trimis către baza de date este criptat cu scopul de a evita ca comunicarea dintre client și server să fie interceptată de entități neautorizate.

Îndată ce codul criptat ajunge la baza de date acesta este decriptat și comparat cu valoare deja stocată pe server, etapă efectuată pentru confirmarea identității utilizatorului.

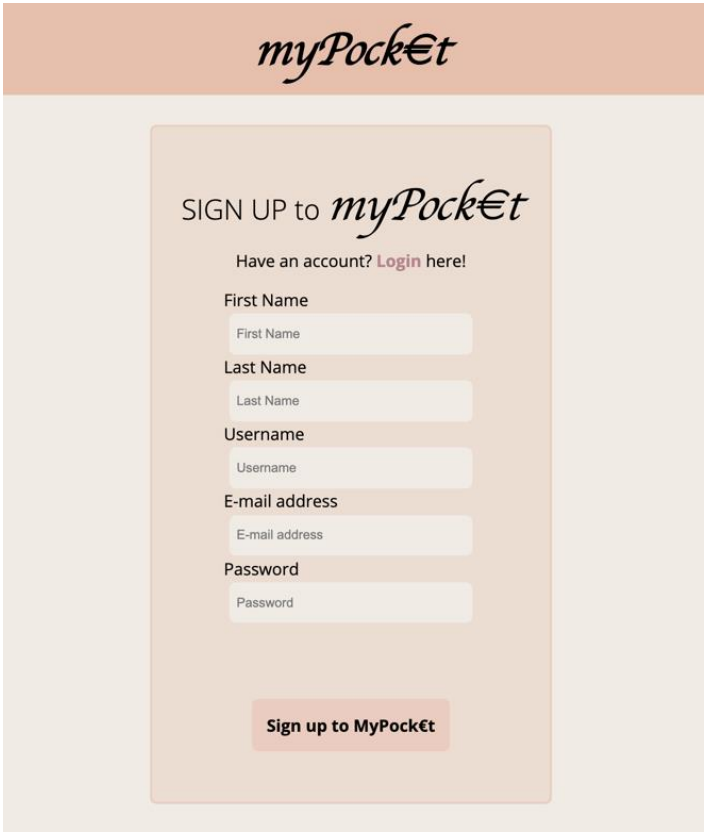
La implementarea algoritmului s-a utilizat librăria **big-integer** [33], deoarece primitiva *Number* din JavaScript nu poate reprezenta numere mai mari de $2^{53} - 1$ [34]. Ca algoritmul RSA să fie robust este necesar ca cheia acestuia să fie de cel puțin 1024 de biți, un număr care depășește cu mult valoare maximă oferită de *Number*. În plus, librăria dispune de foarte multe metode care facilitează calcularea parametrilor din algoritm. Metodele respective sunt mult mai rapide decât funcțiile customer. De exemplu, dacă facem referință la metoda *isProbablePrime,* aceasta oferă un răspuns mult mai rapid decât orice funcție implementată de dezvoltator, căci pentru a verifica dacă un număr reprezentat de 256 de biți este prim, indiferent că folosim sau nu Cifrul lui Eratostene, ar dura mai bine de câteva minute.

Codul aferent implementării algoritmului RSA este adăugat în **Anexa 4**.

Capitolul 5
PREZENTAREA SOLUȚIEI INFORMATICE

5.1. Utilizarea aplicației

Primul pas pentru a accesa aplicația *MyPocket* este înregistrarea (**Figura 5.1**). Utilizatorul își creează contul pe *MyPocket* completând câmpurile formularului de înregistrare cu date personale și valide. În



cazul în care datele oferite nu sunt valide, atunci submiterea formularului nu este posibilă.

Figura 5.1 Pagina de înregistrare în aplicația MyPocket

Etapa care precede înregistrarea este etapa de autentificare a utilizatorului (**Figura 5.2**), care se realizează în două etape – **2FA**. În prima etapă utilizatorul completează informațiile cunoscute doar de el: username-ul și parola, ambele setate în faza anterioară. Îndată ce sunt verificate și validate valorile introduse de utilizator, acesta este redirecționat în pagina de verificare a informațiilor de posesie. Dacă username-ul sau parola introduse nu coincid cu valorile memorate în baza de date, atunci utilizatorul primește un mesaj de eroare.

În cea de a doua etapă de autentificare utilizatorul primește un cod de verificare (OTP) pe email-ul indicat la înregistrare. Dacă din anumite motive codul nu a fost expediat, atunci se optează pentru retrimiterea codului de verificare. Odată ce a fost validat codul, utilizatorul este direcționat către pagina principală a aplicației.

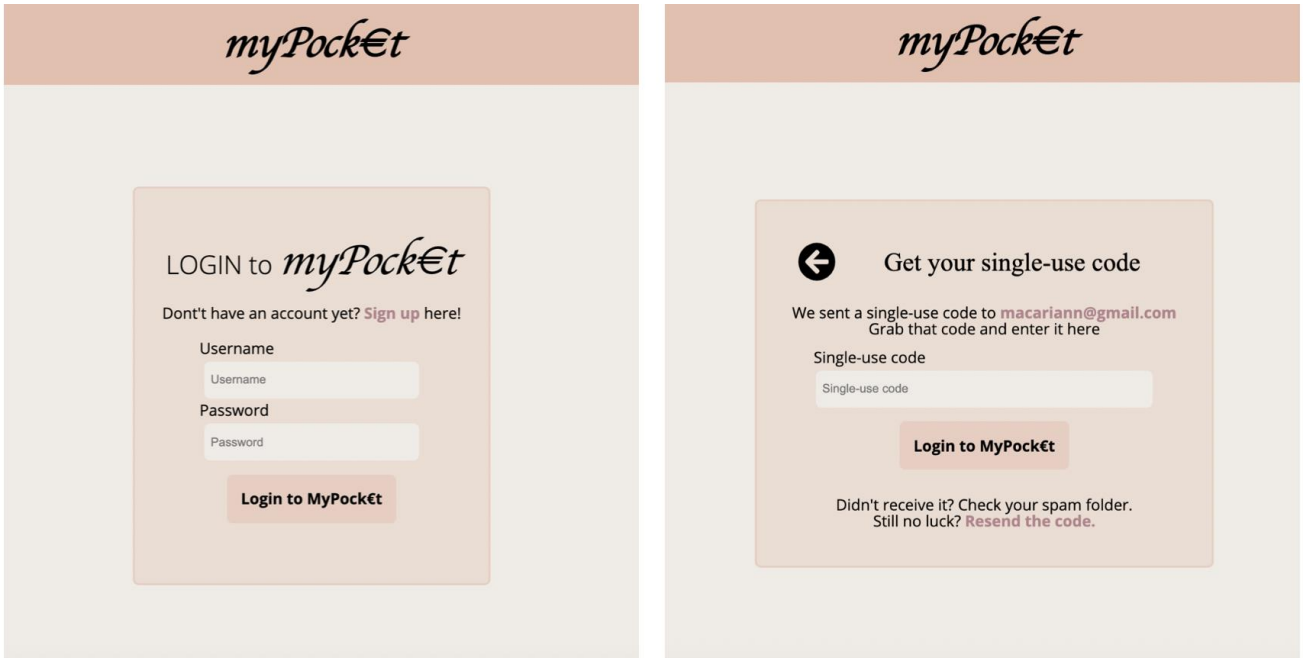


Figura 5.2 Cele două etape de autentificare în aplicația MyPocket

Pagina principală a aplicației *MyPocket* este un tablou de bord (**Figura 5.3**) care prezintă mai multe funcționalități încorporate. Pe acest tablou de bord sunt incluse trei componente: *Accounts*, *Transactios* și *Expenses.*

Accounts oferă informații despre conturile bancare ale utilizatorului. În secțiunea de *Total Balance* este prezentă suma pe care utilizatorul o deține în conturi. Butonul de *Add a New Account* deschide o nouă pagină de interogare unde sunt prezentate băncile compatibile cu aplicația. Dacă una dintre bănci este aleasă atunci utilizatorul este direcționat pe o nouă pagină unde trebuie să urmeze fluxul de OAuth2 specific băncii selectate. Odată încheiat acest flux, utilizatorul ajunge pe pagina de *View All Accounts*, unde sunt prezentate toate conturile utilizatorului împreună cu cele mai relevante informații (**Figura 5.4**).

Transactions furnizează informațiile aferente tranzacțiilor efectuate de către utilizator de pe conturile bancare importate. În secțiunea de *Last n transactions* sunt prezentate ultimele *n* tranzacții, unde n poate fi modificat și poate lua valori de: 3, 5 sau 7. Dacă butonul de *View All Transactions* este acționat, atunci clientul este redirecționat către o nouă pagină unde sunt prezentate toate tranzacțiile utilizatorului împreună cu cele mai relevante informații (**Figura 5.5**).

Expenses expune grafic tranzacțiile efectuate în ultima lună, conform categoriilor definite de codurile MCC. Prin intermediul secțiunii *Budget*, utilizatorul poate să își stabilească bugetele viitoare. Există posibilitatea de a seta un buget total aplicat asupra tuturor categoriilor, cât și posibilitatea de a fixa un buget pentru fiecare categorie în parte (transport, utilități etc.). Acționarea butonului *Create a New Budget* deschide o nouă pagină, prin care pot fi filtrate bugetele (**Figura 5.6**). În mod implicit, bugetele sunt setate cu valabilitate de o lună începând de la data acționării butonului. Spre deosebire de alte aplicații, utilizatorul poate să prelungească / seteze un alt termen de valabilitate a bugetului, în funcție de necesități. Bugetele sunt stabilite pe categorii de interes. Dacă utilizatorul optează ca noul buget să fie stabilit pe o categorie setată anterior, atunci apare o pagină de interogare, în care utilizatorul confirmă sau infirmă modificarea efectuată. Eliminarea bugetelor are loc din aceeași secțiune.

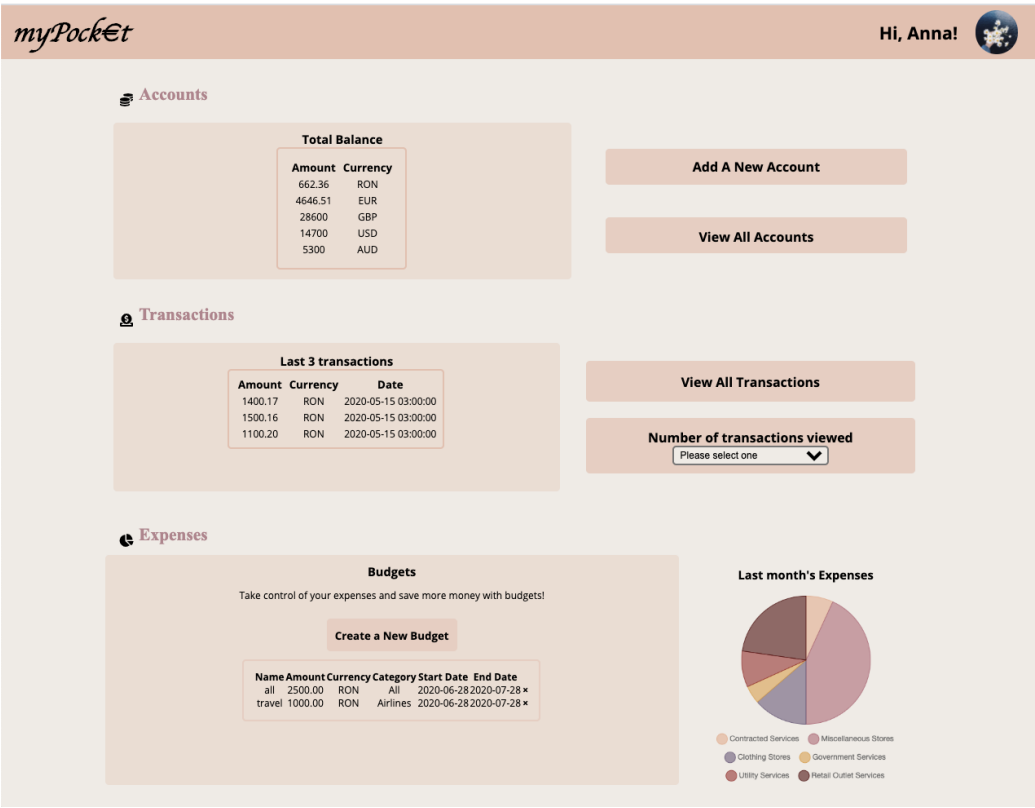


Figura 5.3 Panoul de bord al aplicației MyPocket

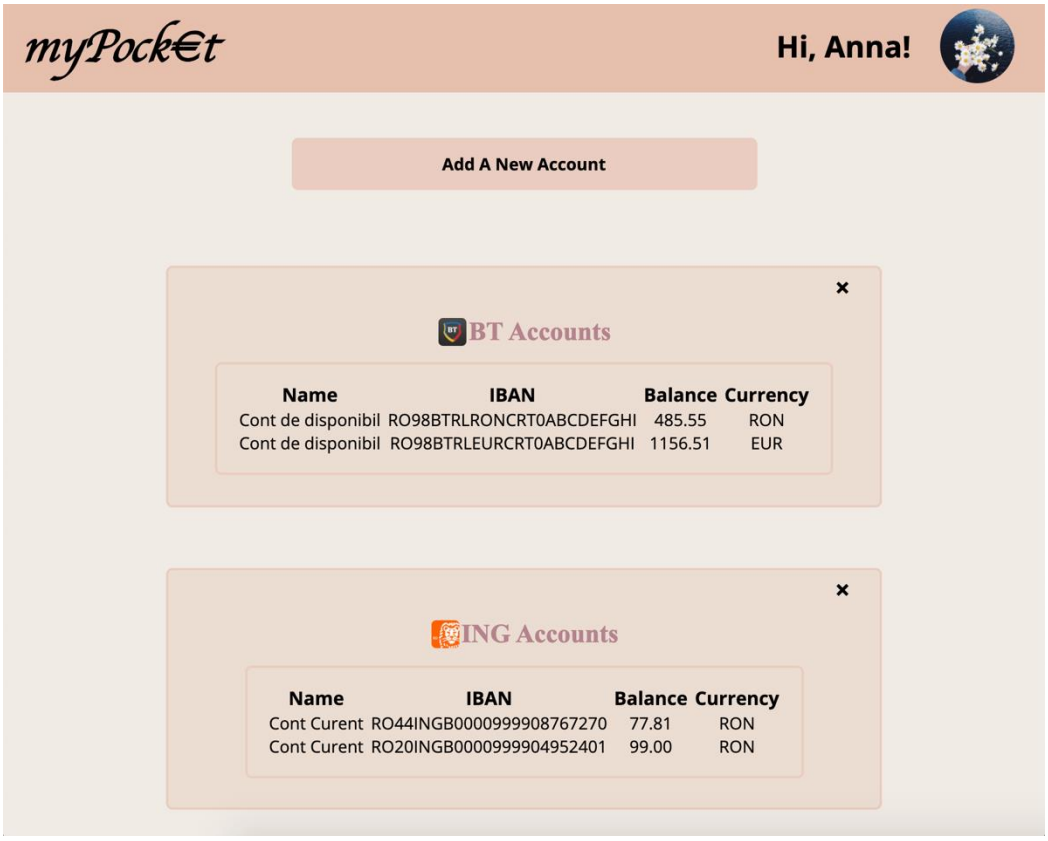
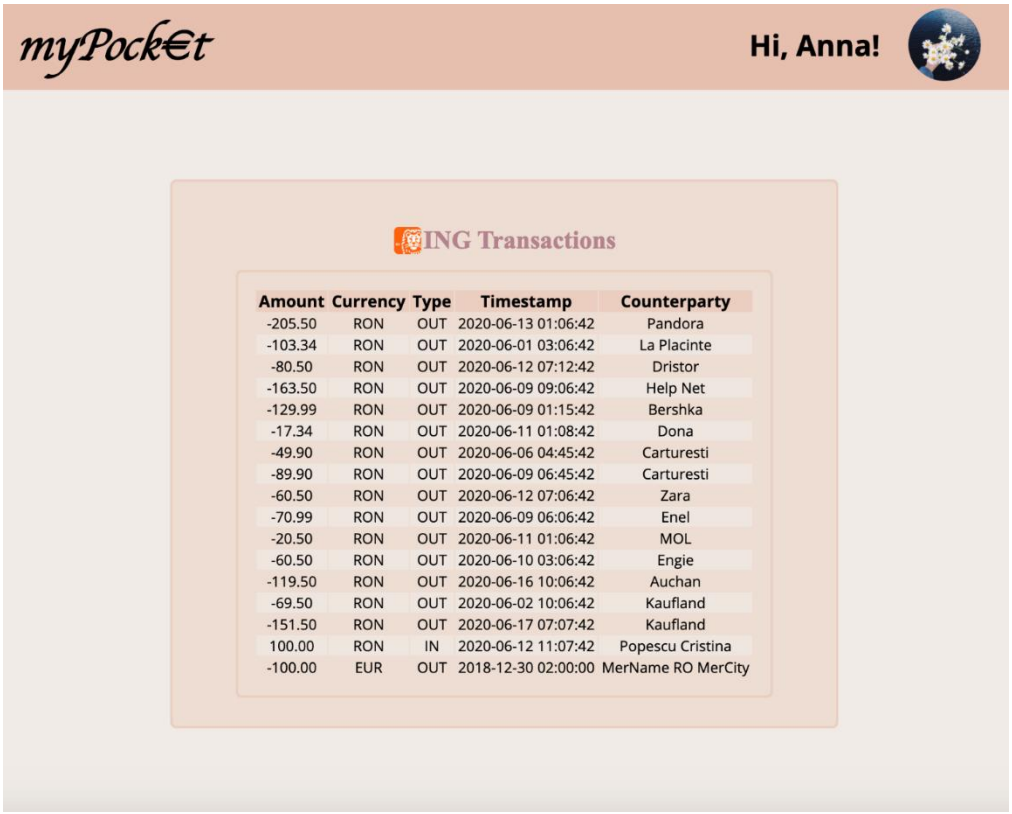


Figura 5.4 Pagina de prezentare a conturilor



Amount	Currency	Type	Timestamp	Counterparty
-205.50	RON	OUT	2020-06-13 01:06:42	Pandora
-103.34	RON	OUT	2020-06-01 03:06:42	La Placinte
-80.50	RON	OUT	2020-06-12 07:12:42	Dristor
-163.50	RON	OUT	2020-06-09 09:06:42	Help Net
-129.99	RON	OUT	2020-06-09 01:15:42	Bershka
-17.34	RON	OUT	2020-06-11 01:08:42	Dona
-49.90	RON	OUT	2020-06-06 04:45:42	Carturesti
-89.90	RON	OUT	2020-06-09 06:45:42	Carturesti
-60.50	RON	OUT	2020-06-12 07:06:42	Zara
-70.99	RON	OUT	2020-06-09 06:06:42	Enel
-20.50	RON	OUT	2020-06-11 01:06:42	MOL
-60.50	RON	OUT	2020-06-10 03:06:42	Engie
-119.50	RON	OUT	2020-06-16 10:06:42	Auchan
-69.50	RON	OUT	2020-06-02 10:06:42	Kaufland
-151.50	RON	OUT	2020-06-17 07:07:42	Kaufland
100.00	RON	IN	2020-06-12 11:07:42	Popescu Cristina
-100.00	EUR	OUT	2018-12-30 02:00:00	MerName RO MerCity

Figura 5.5 Pagina de prezentare a tranzacțiilor

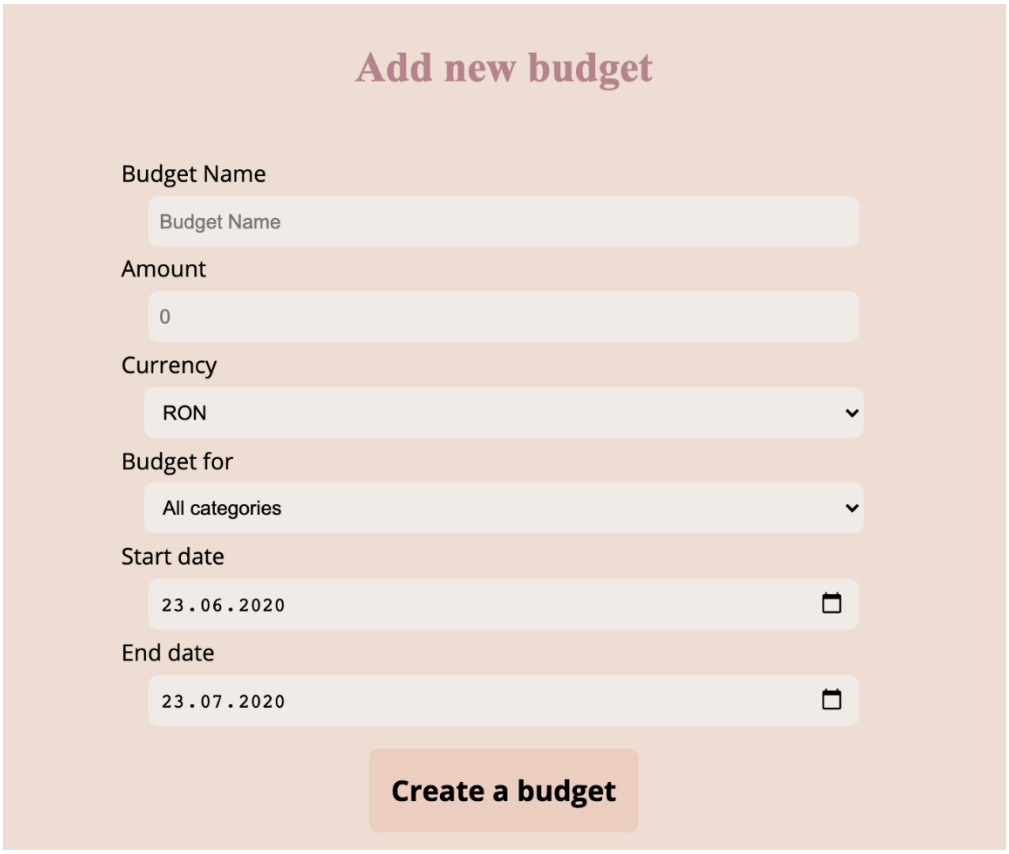


Figura 5.6 Pagina de creare a unui nou budget

Pe fiecare ecran al aplicației este prezent un header. Acesta permite navigarea între ecranele principale ale aplicației și prezintă câteva informații esențiale despre utilizator. Sub poza de profil a utilizatorului, expusă în stânga ecranului este ascuns un meniu care devine vizibil doar la glisare mouse-ului peste poză. Mini-meniul oferă două opțiuni: trecerea pe pagina de profil a utilizatorului sau delogarea din aplicație.

Pe pagina de profil sunt indicate datele utilizatorului care au fost introduse la înregistrare sau la ultimul update al profilului: *Fisrt Name, Last Name, email-ul, username-ul* și *poza de profil*, care dacă nu este modificată de către utilizator este setată în mod implicit (**Figura 5.7**). Pentru a trece în modul de editare a datelor trebuie acționat butonul de *Edit*. În noua pagină există posibilitatea de a modifica email-ul, username-ul și parola (**Figura 5.8**).

La activarea butonului de *Logout*, sesiunea de logare a utilizatorului este încheiată, iar acesta este direcționat către pagina de autentificare.

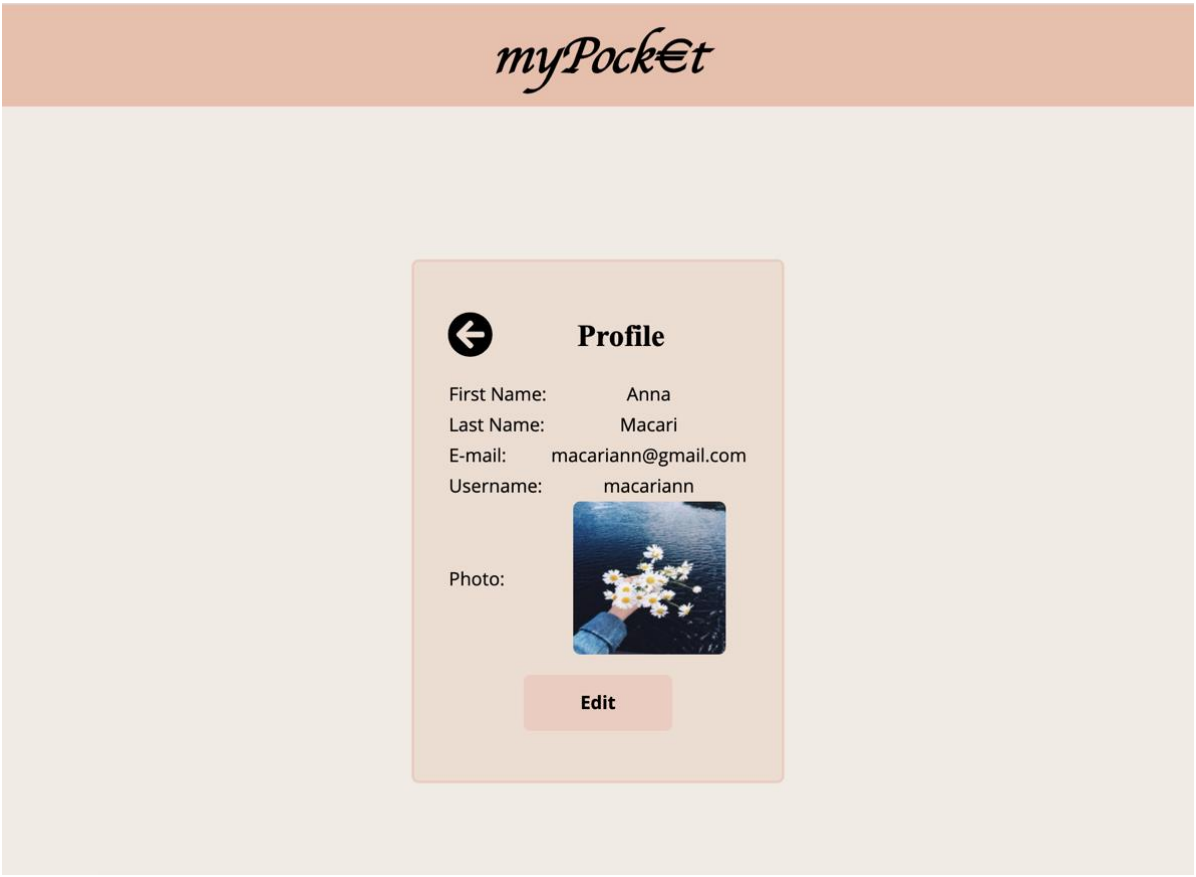


Figura 5.7 Pagina de profil a utilizatorului

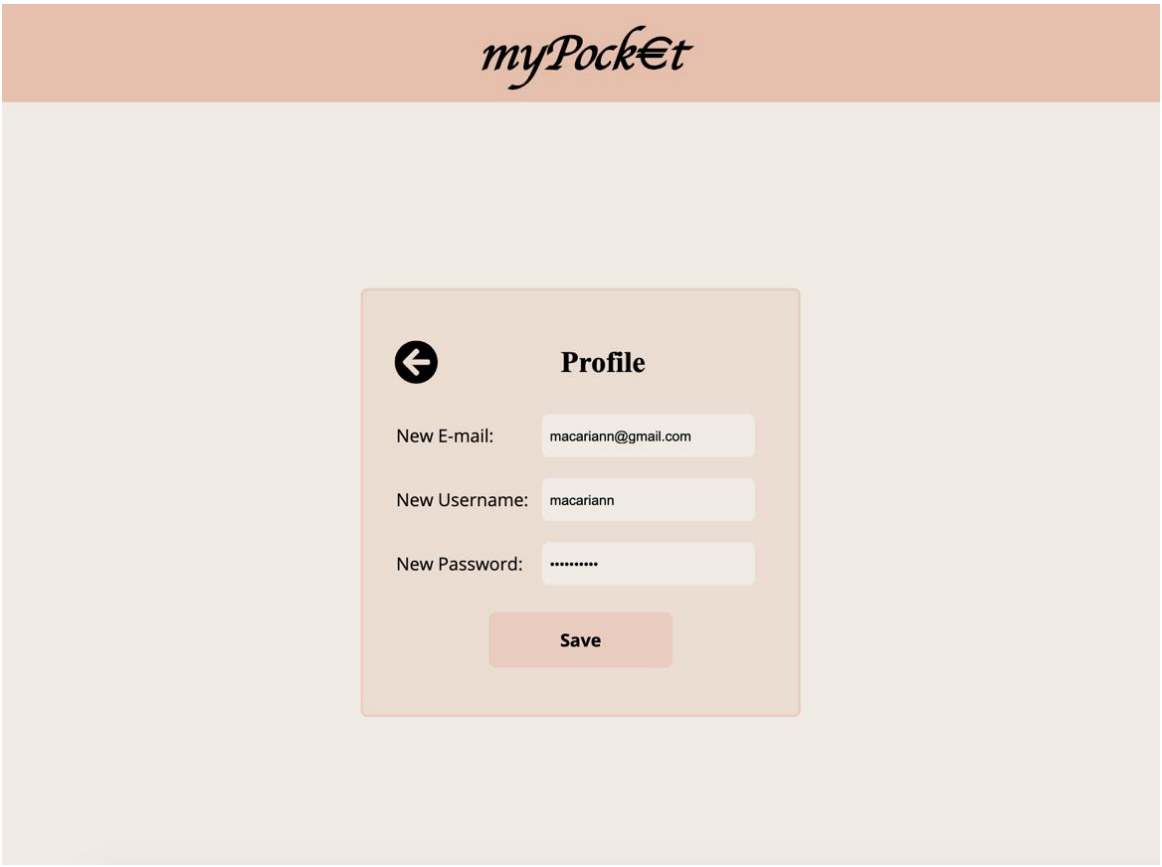


Figura 5.8 Modul de editare a datelor pe profilul utilizatorului

5.2. Compararea cu alte aplicații asemănătoare

Se pot găsi câteva aplicații asemănătoare care deja au fost puse în producție, precum: Spendee sau Number. În linie generală, atât aplicația MyPocket, cât și cele enumerate anterior oferă aceleași funcționalități: prezentarea conturilor bancare, prezentarea tranzacțiilor efectuate, setarea bugetelor și vizualizarea cheltuielilor. Adică toate aceste soluții informatice au ca scop menținerea finanțelor într-o formă bine definită și oferirea unei imagini de ansamblu a acestora. Ca dezavantaj, nici o aplicație nu pune la dispoziție efectuarea plăților direct din aplicație, ceea ce poate reprezenta o îmbunătățire remarcabilă.

În aplicația Spendee, fiind o aplicație internațională, sunt integrate băncile mai multor țări, aproximativ 2500, un număr impresionant. Însă pentru România ofertele nu sunt eminente, doar câteva bănci specifice acestui teritoriu.

Accesibilitatea pe telefoanele mobile este un mare avantaj, iar Spendee oferă aceasta, însă contracost. Pentru MyPocket transformarea soluției informatice web într-o aplicație mobilă este un plan de viitor, la fel ca și posibilitatea de a face schimbul dint-o valută în altă valută.

Capitolul 6
ANALIZA NIVELULUI DE SECURITATE AL APLICAȚIEI

Nivelul de securitate și gradul de protecție al datelor sensibile și cu caracter personal este definit ca unul dintre cele mai important aspect în analiza unei soluții informatice. În contextul societății moderne informația înseamnă putere, iar protecția acesteia este imperios necesară. Drept dovadă, la nivel european a fost dezvoltat Regulamentul General de Protecție a Datelor (GDPR). De asemenea, odată cu adoptarea directivei PSD2, majoritatea aplicațiilor au început să utilizeze 2FA pentru a asigura SAC-ul.

Securitatea aplicație la un nivel cât mai înalt poate fi asigurată prin analiza potențialelor vulnerabilități, găsirea unor soluții mai performante și implementarea acestora.

Pentru a activa conform standardelor și pentru a asigura SAC-ul, aplicația folosește autentificarea dublă, unde, primul pas constă în validarea datelor din categoria **cunoștințelor** – username-ul și parola, iar pasul doi presupune confirmarea datelor din categoria **posesiei** – OTP-ul primit prin email.

În vederea protejării datelor sensibile în cazul unei eventuale breșe de securitate, parolele sunt salvate criptat în baza de date utilizând funcția de dispersie SHA-256. Acesta presupune executarea a 64 de runde de operații pentru criptare, iar șirul rezultat este unul de 256 de biți. Prin natura operațiilor folosite pentru calcularea rezultatului, funcțiile hash sunt ireversibile, adică aflarea șirului inițial de caractere, pornind de la rezultat fiind este imposibilă.

Adevărat, existența unor baze de date care conțin rezultatul funcțiilor hash pentru o listă amplă de combinații uzuale face ca parolele comune precum „12345” să fie vulnerabile și algoritmul nesigur. Însă, aici, aceasta nu presupune o problemă, deoarece pentru ca o parolă să fie declarată validă în cadrul aplicației, aceasta trebuie să îndeplinească condițiile unei parole complexe – conține cel puțin: o literă mare, o literă mica, o cifră și un caracter special.

Al doilea pas este cel care garantează o autentificare sigură. În general, informația de posesie este dificil de obținut. Iar de cele mai multe ori aceasta este reprezentată printr-un cod unic – OTP expediat fie prin SMS, fie prin email, așa cum e și în cadrul aplicației.

Codul de verificare reprezentă un cod unic, aleator, de o lungime de 8 caractere, exprimat în hexa. Obținerea respectivului cod se face prin metoda **randomBytes()** din librăria **crypto** [35]. Numele librării deja implică faptul că metoda de obținere a numărului aleator este una sigură, aceasta generând date pseudo-aleatorii puternic criptate. Codul de verificare având o lungime de 8 caractere și fiind exprimat în hexazecimal, este relativ greu de memorat pentru a fi utilizat de o entitate neautorizată.

Criptarea codului de validare prin intermediul algoritmului RSA oferă un plus de siguranță acestei etape. Algoritmul se bazează pe teoria numerelor prime mari, ceea ce îl face o problemă cu complexitate NP, adică dificil de decriptat prin metodele clasice. Utilizarea numerelor prime mari înseamnă că cheia publică a algoritmului are cel puțin 512 biți. În aplicație, cheia publică are o lungime de 1024 de biți, astfel, garantat, codul criptat nu poate fi decriptat de cineva care nu deține cheia.

Singura îmbunătățire posibilă la această etapă ar fi expedierea OTP-ului nu prin email, ci prin SMS, deoarece poșta electronica nu este la fel de sigură.

Legătura dintre aplicație și serverul de backend al soluției informatice este efectuată printr-o serie de API-uri RESTful. Un API RESTful este constituit din:

- Endpoint – un localizator uniform de resurse (URL) de forma: protocolul prin care se realizează conexiunea (HTTP), adresa de bază a host-ului și calea către resursa sau colecția de resurse dorite. Exemplu: *http://api.example.com/resources/*;
- Metoda HTTP: **GET**, **POST**, **PUT**, **DELETE**;
- Tipul conținutului (ContentType) primit în cadrul cererii și/sau trimis ca răspuns al acesteia. Exemplu: *application/json, plain/text*.

Schimbul de date cu serverul se realizează prin intermediul protocolului HTTP. Acesta este considerat un protocol riscant. Astfel, pentru un nivel mai înalt de protecție, în cadrul soluției, trebuie utilizat protocolul **HTTPS**. *HyperText Transfer Protocol Secure* reprezintă varianta criptată a HTTP-ului, care folosește Transport Layer Security (TLS) sau Secure Sockets Layer (SSL). Spre deosebire de protocolul HTTP, prin care datele sunt transmise în mod direct, protocolul HTTPS încapsulează datele cu ajutorul unui flux TLS/SSL, astfel acestea sunt criptate înainte de a fi transmise clientului [36]. Utilizarea acestuia împiedică atacurile de tipul „man-in-the-middle", cum simpla interceptare a comunicării dintre server și client nu mai este suficientă pentru accesarea informațiilor transmise. De asemenea, pe lângă asigurarea protecției și integrității datelor, protocolul HTTPS pune la dispoziție o metodă de autentificare a serverului web prin intermediul certificatelor digitale. În acest mod, clientul poate fi sigur de identitatea serverului cu care comunică.

Implicit, serverul web *Express* folosit pe backend-ul aplicației aplică protocolul HTTP. Astfel, pentru remedierea problemei de securitate expuse mai sus trebuie să urmăm pașii:

- Obținerea unei chei private și a unui certificate digital emis de o autoritate cunoscută;
- Crearea unui server HTTPS prin intermediul modulului dedicat din Node.js;
- Utilizarea serverului Express precum un middleware pentru prelucrarea request-urilor primite prin intermediul serverului HTTPS creat anterior;
- Redirecționarea traficului HTTP către noul protocolul sigur HTTPS.

O altă expunere de securitate a aplicațiilor care folosesc limbajul SQL pentru manipularea informațiilor stocate în bază de date sunt atacurile de tipul *„SQL injection"*. Acesta reprezintă unul dintre cele mai simple și mai uzuale moduri de atac asupra securității unei aplicații informatice care constă în dobândirea sau alterarea informațiilor din baza de date prin introducerea unei secvențe SQL în câmpurile de introducere a datelor în cadrul aplicație.

Pentru combaterea acestei vulnerabilități, nu s-a utilizat interogarea bazei de date prin compunerea explicită a comenzilor SQL (raw queries). Dar, au fost aplicate metodele destinate manipulării datelor puse la dispoziție de biblioteca Sequelize, precum: *find(), findAll(), findOrCreate(), create()* și *destroy()*. Acestea modifică datele introduse de utilizator prin marcarea caracterelor speciale, ca de exemplu: *Hello "World"* devine: *Hello\"World\"*, astfel oferind protecție împotriva atacurilor de tipul SQL injection.

CONCLUZII

Ținând cont de situația actuală și de schimbările survenite în urma directivei PSD2 în industria bancară, soluția informatică prezentată mai sus ar cunoaște un adevărat succes, deoarece actualmente în România nu există nici o aplicație care se ofere accesul la mai multe bănci concomitent.

Astfel, ceea ce a început ca un simplu contor de cheltuieli s-a dezvoltat într-o aplicație care are drept scop menținerea finanțelor în formă și oferirea unei imagini de ansamblu a acestora, fără nici un fel de efort, accesibil de oriunde, oricând și doar la câteva clicuri distanță.

Pentru alegerea soluției a fost realizată o analiză a situației actuale a domeniului bancar. Iar din această analiză a reieșit necesitatea implementării unei astfel de aplicații web care să permită gestionarea resurselor. Gestionarea finanțelor, fără stresul monitorizării fiecărui bănuț cheltuit, ar ajuta la schimbarea obiceiurilor financiare și la garantarea unor economii.

Următorul pas în realizarea soluției propuse a fost proiectarea acesteia conform metodologiilor de proiectare a sistemelor informatice și a cerințelor impuse de PSD2. Identificarea utilizatorilor aplicației și a funcționalităților disponibile acestora a reprezentat punctul de plecare în definirea tabelelor bazei de date și a relațiilor dintre acestea.

Implementarea soluției a fost posibilă datorită utilizării tehnologiilor moderne destinate dezvoltării aplicațiilor scalabile și robuste, cât și datorită infrastructurii cloud, care a sporit considerabil flexibilitatea și eficiența aplicației. Coeziunea acestora a constituit cel mai rapid drum de la o idee, la un produs finit – o aplicație funcțională care gestionează finanțele.

Întrucât securitatea este prioritatea numărul unu, singura cantitate minimă necesară de date sensibile (datele de autentificare) sunt criptate, iar autentificare este realizată în două etape. Algoritmii folosiți (SHA-256, RSA) sunt siguri și imposibili de decriptat. Toate informațiile contului financiar sunt folosite numai în modul de citire, cu scopul verificării soldului și a descărcării listelor de tranzacții. Manipularea contului nu poate exista sub nici o formă, deoarece aplicația nu inițializează plăți.

Comunicarea cu băncile pentru obținerea datelor financiare este una sigură. Odată autentificat în aplicația de Internet Banking a băncii, aplicația primește un jeton de autorizare cu timp de viață limitat, prin intermediul căruia accesează informațiile bancare, dar nici de cum nu primește parola de autentificare a utilizatorului cu banca. Interceptarea datelor în procesul de transfer dintre server și contul de aplicație este dificilă, astfel doar utilizatorii autorizați au acces la informații.

Asigurarea securității nu este un produs final, dar este un proces continu, astfel optimizări și îmbunătățiri în cadrul aplicației mereu ar fi posibile, precum cele menționate în capitolul anterior: *Analiza nivelului de securitate al aplicației* sau ca de exemplu: toate datele stocate în baza de date să fie dublu criptate cu algoritmi diferiți.

Consider că datorită necesității de evoluție și a dorinței de a menține controlul indiferent de domeniu, această aplicația s-ar bucura de un real succes, pentru că gestionarea resurselor financiare este o problemă cu care se confruntă fiecare dintre noi.

Bibliografie

[1] T. Spring, „Introduction to Information Security”, Elsevier, 2014.

[2] „Dex Online” Available: https://dexonline.ro/definitie/securitate. [Accesat 23 05 2020].

[3] United States Government General Accountin Office, „Critical Infrastructure Protection Challenges and Efforts to Secure Control Systems” GAO-04-354, US.

[4] J. Katzel, „Many Facets, One Point” *Control Engineering*, July, 2005..

[5] M. Naedele și D. Dzung, „Industrial Information System Security Part 1” *ABB Review,* nr. 2, p. 66–70, 2005.

[6] Laboratory Idaho National, „Control Systems Cyber Security: Defense in Depth Strategies” *U.S. Department of Homeland Security,* nr. INL/EXT-06011478, May, 2006.

[7] Microsoft și C2 Evaluation and Certification for Windows, „Microsoft Online” 1 November 2006. Available: http://support.microsoft.com/default.aspx?scid=kb;en-us;. [Accesat 02 06 2020].

[8] D. B. Parker, Fighting Computer Crime, New York: Wiley Publishing, 1998.

[9] National Institute of Standards and Technology, „Data Encryption Standard (DES)” U.S. Department of Commerce, FIPS 46-3, 1999.

[10] M. Zelich, „Quora” 05 06 2019. Available: https://www.quora.com/What-is-the-time-complexity-for-an-RSA-algorithm-What-is-a-solution-with-the-derivation-of-the-complexity?fbclid=IwAR17BsNoi0AhY2gGOJswcAcNOl2Zof8UprH8Z1rDPE1kGTs8Odi42JDBwtE. [Accesat 22 06 2020].

[11] D. Boneh și R. Venkatesan, „Breaking rsa may not be equivalent to factoring Proceedings of Eurocrypt” *LectureNotes in Computer Science,* vol. 98, nr. 1233, p. 59–71, 1998.

[12] R. Rivest, The MD5 Message-Digest Algorithm, MIT Laboratory for Computer Science and RSA Data Security: RFC, 1992.

[13] B. Schneier, „Cryptanalysis of MD5 and SHA: Time for a New Standard” 2004. Available: http://schneier.com/essay-074.html.. [Accesat 12 06 2020].

[14] B. Schneier, „Cryptanalysis of SHA-1” 2005. Available: http://www.schneier.com /blog /archives/2005/02/cryptanalysis_o.html . [Accesat 12 06 2020].

[15] „openbankingtracker” Available: https://www.openbankingtracker.com/?page=1&query=romania. [Accesat 12 06 2020].

[16] „autonomia.digital” 29 11 2019. Available: https://autonomia.digital/2019/11/29/two-factor-authentication.html. [Accesat 14 06 2020].

[17] I. Lungu, A. Bâra, C. Bodea, I. Botha și V. Diaconiț, Baze de date. Organizare, proiectare și implementare, București: Editura ASE, 2011.

[18] V. Pupezescu, Curs și laborator - Proiectarea bazelor de date, București, 2019-2020.

[19] „Stack Overflow” Stack Overflow Developer Survey, 2019. Available: https://insights.stackoverflow.com/survey/2019#technology- programming-languages. [Accesat 15 06 2020].

[20] A. Toma, „andrei.ase.ro/tehnologiiweb.html” 2019. Available: http://andrei.ase.ro/tehnologiiweb.html?fbclid=IwAR1304QVFZJQdZLTCUn3sLHJfrT5zWI6TkzT4pMdnlkLXKX80JW91G-hQp0. [Accesat 14 06 2020].

[21] „nodejs.or,” Available: https://nodejs.org/en/. [Accesat 14 06 2020].

[22] „toptal.com” 2019. Available: https://www.toptal.com/nodejs/why-the-hell-would-i-use-node-js. [Accesat 14 06 2020].

[23] „expressjs.com” Available: https://expressjs.com. [Accesat 14 06 2020].

[24] „sequelize.org” Available: https://sequelize.org. [Accesat 14 06 2020].

[25] M. Zurini și A. Zamfiroiu, Calitate și testare software, București: Editura ASE, 2017.

[26] GmbH și Solid IT, „db-engines.com” 06 2019. Available: https://db-engines.com/en/ranking.. [Accesat 12 06 2020].

[27] „umfcv.ro” Available: http://www.umfcv.ro/files/0/3/03_Baze%20de%20date.pdf. [Accesat 14 06 2020].

[28] „www.vuemastery.com” 2019. Available: https://www.vuemastery.com/courses/intro-to-vue-js/vue-instance/?fbclid=IwAR2Kf9iCONg43mBEKzm2XgCKUVR-WbTFE_2mla-Hee4ZjnbS2z-egVdjPvw. [Accesat 14 06 2020].

[29] „varonis.com” Available: https://www.varonis.com/blog/what-is-oauth/. [Accesat 14 06 2020].

[30] „digitalocean.com” 2018. Available: https://www.digitalocean.com/community/tutorials/an-introduction-to-oauth-2. [Accesat 14 06 2020].

[31] „developer.okta.com” 2017. Available: https://developer.okta.com/blog/2017/06/21/what-the-heck-is-oauth. [Accesat 14 06 2020].

[32] „vuex.vuejs.org” 2019. Available: http://www.vuex.vuejs.org. [Accesat 15 06 2020].

[33] „github.com” Available: https://github.com/peterolson/BigInteger.js. [Accesat 14 06 2020].

[34] „developer.mozilla.org” Available: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/BigInt. [Accesat 14 06 2020].

[35] „developer.mozilla.org” Available: https://developer.mozilla.org/en-US/docs/Web/API/Crypto?fbclid=IwAR27l4im6p17ggsHdJ03nZceaulSHItvOFUIxsmAzV-6j-0GrFvTA7pjn5U. [Accesat 15 06 2020].

[36] „www.eff.org” Available: https://www.eff.org/encrypt-the-web. [Accesat 16 06 2020].

Anexa 1

1. **Obținerea unui certificat, a cheii publice asociate acestui certificat și integrarea lor sub forma unui agent HTTPS pentru inițierea fluxului OAuth 2.0. în cadrul băncii ING.**

```
var httpsAxios = axios.create({
    httpsAgent: new https.Agent({
        cert: fs.readFileSync(path.join
            (__dirname, env.CERT_PATH + 'example_eidas_client_tls.cer')),
        key: fs.readFileSync(path.join
            (__dirname, env.CERT_PATH + 'example_eidas_client_tls.key'))
    })
});
```

2. **Calcularea parametrilor: date și digest, care construiesc semnătura digitală.**

```
let payload = 'grant_type=client_credentials';
let payloadDigest = 'SHA-256=' +
    crypto.createHash('sha256').update(payload).digest('base64');
let dateString = new Date().toGMTString();
let signingString = '(request-target): post ' + env.TOKEN_PATH + '\n' +
    'date: ' + dateString + '\n' +
    'digest: ' + payloadDigest;
let sign = crypto.createSign('RSA-SHA256');
sign.write(signingString);
sign.end();
```

Anexa 2

Exemplul de rutare a componentelor.

```
import VueRouter from 'vue-router';
import HomePage from '../components/LoginOK/HomePage';
import ViewAccounts from '../components/ViewAccounts/ViewAccounts';

const router = new VueRouter({
  mode: 'history',
  routes: [
    {
      path: '/',
      name: 'Home',
      component: HomePage,
      meta: {
        requiresAuth: true
      }
    },
    {
      path: '/accounts',
      name: 'ViewAccounts',
      component: ViewAccounts,
      meta: {
        requiresAuth: true
      }
    }
  ]
});

export default router;
```

Anexa 3

Exemplul de stocare a datelor în *state* și de modificare a acestora prin *actions*. Acțiunea de login.

```
import axios from 'axios';
import router from '../../router';
import consts from '../../consts';

export default {
  namespaced: true,
  state: {…},
  mutations: {…},
  getters: {…},
  actions: {
    login({ commit }, { username, password }) {
      commit('clearErrors');
      axios.post(consts.backendServer + '/login', {
        username, password
      }).then((result) => {
          if (result.data.message == "ok") {
            console.log(result.data.userData);
            commit('setUser', result.data.userData);
            router.push('/validation', () => { });
          } else if (result.data.message == "WRONG_PASSWORD") {
              commit('addError', "Incorrect Username or Password.");
          } else if (result.data.message == "NOT_FOUND") {
              commit('addError', "This user doesn't exist.");
          }
        }).catch(console.error);
    }
  }
};
```

Anexa 4

Algoritmul RSA - funcția de generare a cheii publice și funcția de decriptare.

```
const bigInt = require('big-integer');

function generateRSA(keysize) {

  function randomPrime(bits) {
    const min = bigInt(6074001000).shiftLeft(bits).minus(33);
    const max = bigInt.one.shiftLeft(bits).minus(1);
    while (1) {
      const p = bigInt.randBetween(min, max);
      if (p.isProbablePrime(256)) {
        return p;
      }
    }
  }

  const e = bigInt(65537);
  let p; let q; let lambda;

  do {
    p = randomPrime(keysize / 2);
    q = randomPrime(keysize / 2);
    lambda = bigInt.lcm(p.minus(1), q.minus(1));
  } while (bigInt.gcd(e, lambda).notEquals(1) ||
p.minus(q).abs().shiftRight(keysize / 2 - 100).isZero());

  return {
    N: p.multiply(q),
    e: e,
    d: e.modInv(lambda)
  };
}

var generateRSAparam = generateRSA(1024);
var e = generateRSAparam.e;
var d = generateRSAparam.d;
var N = generateRSAparam.N;

function decryptRSA(c) {
  return bigInt(c).modPow(d, N);
}

module.exports = { decryptRSA, e, N };
```

Algoritmul RSA- funcția de criptare.

```
import axios from 'axios';
import consts from '../consts';
import bigInt from 'big-integer';

async function encryptRSA(M) {
      var publicKey = await axios.get(consts.backendServer +
            '/publickey');
      return bigInt(M).modPow(publicKey.data.e, publicKey.data.N);
}

export {encryptRSA};
```