Universitatea Tehnică a Moldovei

PAROLE SIGURE UTILIZÂND TEHNICI DE CRIPTOGRAFIE
SECURE PASSWORDS USING CRYPTOGRAPHIC TECHNIQUES

Student: **gr. TI-231 FR,**
Apareci Aurica
Celan David
Vreme Adrian

Coordonator: **Andrievschi-Bagrin Veronica,**
asist.univ.

Chișinău, 2025

CUPRINS

INTRODUCERE

„Securitatea modernă nu mai depinde doar de utilizator,
ci de algoritmi care transformă complexitatea în protecție”

Securitatea aplicațiilor software a devenit o componentă fundamentală în dezvoltarea oricărui sistem informatic modern. Într-o lume digitalizată, în care tot mai multe procese sunt automatizate și stocate online, protejarea datelor sensibile ale utilizatorilor nu mai este opțională, ci esențială. Una dintre vulnerabilitățile frecvent exploatate de atacatori este folosirea parolelor slabe, care sunt adesea prea scurte, formate din cuvinte simple sau reutilizate în mod repetat. Aceste parole reprezintă adesea primul nivel de apărare în fața accesului neautorizat, iar compromiterea lor duce la pierderi de date, fraude, afectarea reputației și chiar atacuri în lanț în cadrul infrastructurilor critice.

În acest context, proiectul propus vizează dezvoltarea unei aplicații web moderne, destinată generării și validării parolelor sigure într-un mod accesibil și intuitiv pentru utilizatori. Soluția implementează o suită de algoritmi specializați din domeniul criptografiei, al selecției stocastice și al evaluării euristice, oferind un cadru robust pentru crearea de parole complexe și dificil de compromis. Prin utilizarea unor mecanisme algoritmice riguroase, aplicația permite generarea de parole unice, memorabile și adaptate cerințelor de securitate moderne.

Scopul principal al proiectului este de a oferi utilizatorilor un instrument eficient pentru crearea de parole rezistente la atacuri de tip dicționar sau forță brută. Aplicația propune o experiență interactivă, în care utilizatorul poate configura lungimea parolei, complexitatea, includerea caracterelor speciale sau memorabilitatea acesteia, toate fiind guvernate de reguli de validare algoritmică. Prin impunerea unor standarde stricte încă din etapa de generare a parolei, aplicația contribuie la creșterea rezilienței sistemelor informatice, acționând ca o primă linie de apărare împotriva accesului neautorizat. Totodată, interfața prietenoasă și opțiunile personalizabile o fac ușor de utilizat chiar și pentru persoane fără expertiză tehnică, consolidând ideea că securitatea poate fi accesibilă și eficientă în același timp.

Utilizarea algoritmilor în această librărie nu este doar o alegere tehnică, ci o strategie esențială pentru a asigura coerență, predictibilitate și eficiență în tratarea problemei parolelor nesigure. Prin aplicarea unei succesiuni algoritmice bine definite, problema generării de parole sigure este abordată într-un mod sistematic și reproductibil, reducând astfel subiectivitatea și riscurile asociate soluțiilor ad-hoc.

Capitolul 1 al lucrării este dedicat *analizei domeniului de studiu*. Aici este discutată importanța temei din perspectiva securității aplicațiilor software și sunt identificate principalele riscuri asociate parolelor slabe. Sunt prezentate de asemenea scopul general al proiectului și obiectivele concrete urmărite, ancorate în standarde de securitate recunoscute internațional.

În *capitolul al* II sunt descriși *algoritmii aleși pentru implementarea soluției*. Este prezentat un model de procesare compus de tip pipeline, care integrează mai multe tehnici succesive: un generator crip-

tografic aleatoriu bazat pe entropie, un constructor mnemonic care creează parole memorabile prin șabloane logice, un transformator Leet contextual pentru complexificarea parolelor, un evaluator de tărie care folosește entropia și reguli euristice, precum și o buclă de reîncercare ce permite ajustarea automată a rezultatului până la satisfacerea criteriilor de securitate. Fiecare subcomponentă este analizată din perspectiva funcționalității, a complexității algoritmice și a modelelor teoretice aferente.

Capitolul al III descrie în detaliu realizarea sistemului. Sunt prezentate *cerințele funcționale și nefuncționale*, arhitectura modulară a aplicației, precum și tehnologiile utilizate în dezvoltare. Tot aici este detaliată implementarea fiecărui modul, de la validarea unei parole existente până la generarea automată, și sunt explicate opțiunile de configurare pe care utilizatorul le poate ajusta. Este descrisă și interfața aplicației, care urmărește să fie intuitivă și prietenoasă pentru utilizator.

Testarea și validarea aplicației, tratate în *capitolul al IV*, sunt esențiale pentru demonstrarea robusteții soluției propuse. Sunt descrise metodele de testare utilizate, rezultatele obținute și analiza performanței atât din punct de vedere funcțional, cât și al securității. Sunt incluse și comparații cu alte soluții similare pentru a evidenția punctele forte și limitele abordării propuse.

În final, lucrarea se încheie cu un set de concluzii care sintetizează realizările tehnice și funcționale ale proiectului, reflectând modul în care obiectivele propuse au fost atinse. Se evidențiază valoarea adăugată a librăriei software dezvoltate, atât din punct de vedere al securității, cât și al flexibilității în integrare. Totodată, sunt formulate propuneri clare pentru direcții viitoare de dezvoltare, cum ar fi extinderea mecanismului de generare a parolelor cu suport pentru autentificare multifactor, integrarea cu manageri de parole existenți sau adaptarea algoritmilor pentru medii cu cerințe specifice .

Accentul este pus pe caracterul evolutiv al amenințărilor cibernetice, care impune o permanentă adaptare a mecanismelor de protecție. În acest sens, soluția propusă este gândită ca o bază solidă, dar deschisă, care poate fi îmbunătățită odată cu apariția de noi recomandări, standarde sau tipologii de atacuri. Prin urmare, proiectul nu oferă doar un instrument funcțional pentru prezent, ci și o fundație pentru viitoare dezvoltări în domeniul securității aplicațiilor software.

I ANALIZA DOMENIULUI DE STUDIU

1.1 Importanța securității parolelor în aplicațiile software

Într-un ecosistem digital tot mai interconectat și dependent de servicii online, parolele continuă să reprezinte unul dintre cele mai utilizate, dar și cele mai vulnerabile mecanisme de autentificare. De la aplicații financiare, conturi bancare și portaluri educaționale, până la platforme de socializare, sisteme guvernamentale și aplicații enterprise, autentificarea prin parolă este omniprezentă și joacă un rol esențial în protejarea informațiilor personale și organizaționale.

Această omniprezență a parolelor vine însă cu o responsabilitate majoră: fiecare punct de acces protejat de o parolă devine o potențială țintă pentru atacuri informatice. În absența unor mecanisme riguroase de verificare a complexității și unicității parolelor, sistemele informatice sunt expuse unor riscuri semnificative. Parolele devin astfel veriga slabă într-un lanț altfel bine protejat, iar atacatorii profită exact de această slăbiciune. Mai mult decât atât, în contextul actual, utilizatorii gestionează în paralel zeci de conturi, iar presiunea de a memora multiple parole îi determină pe mulți să recurgă la practici nesigure: utilizarea aceleiași parole pentru mai multe servicii, alegerea unor combinații simple sau formate pe baza unor date personale ușor de dedus (ex: nume, data nașterii, secvențe numerice). Aceste comportamente, combinate cu lipsa unor politici stricte de validare a parolelor la nivel de aplicație, duc la exploatarea ușoară a conturilor și la compromiterea întregului sistem digital.

Parola acționează ca un gardian al identității digitale, fiind adesea singura barieră între un utilizator legitim și un potențial atacator. Din acest motiv, securitatea parolelor nu mai poate fi tratată superficial sau ca o responsabilitate exclusivă a utilizatorului final. Ea devine un subiect de interes strategic atât pentru dezvoltatorii de aplicații software, cât și pentru administratorii de sisteme și arhitecții de infrastructuri informatice. În realitate, parolele slabe sunt printre cele mai ușor exploatate puncte de intrare într-un sistem. Atacurile informatice care vizează parole, precum atacurile de tip brute-force, dictionary attacks, sau metode mai sofisticate precum phishing-ul personalizat, sunt relativ ușor de executat atunci când parolele sunt comune, scurte, sau reutilizate pe mai multe platforme. Mai mult, în contextul actual, în care breșele de securitate sunt dese și informațiile despre milioane de conturi sunt deja expuse în baze de date publice, folosirea unei parole slabe echivalează cu o acceptare tacită a riscului.

Conform studiilor de securitate cibernetică, o proporție semnificativă a atacurilor de succes asupra conturilor online sunt posibile exclusiv din cauza utilizării unor parole nesigure. Incidente notabile din ultimii ani, în care au fost compromise date de milioane de utilizatori, au avut ca punct de pornire exact această verigă slabă. În acest sens, parola nu este doar un element funcțional, ci una dintre cele mai critice componente ale arhitecturii de securitate a unei aplicații.

De aceea, în proiectarea și dezvoltarea sistemelor software moderne, protecția prin parolă trebuie tratată cu aceeași seriozitate ca orice altă măsură de securitate avansată, precum criptarea datelor, auten-

tificarea multifactor sau izolarea la nivel de infrastructură. O parolă puternică, generată cu ajutorul unor algoritmi de înaltă entropie, validată după criterii stricte și stocată în mod sigur, poate constitui un obstacol real în calea atacatorilor. Astfel, gestionarea parolelor nu este doar o chestiune de preferințe ale utilizatorului, ci o responsabilitate tehnică și etică a dezvoltatorului, care trebuie să asigure, prin mecanisme automate și intuitive, crearea unor parole robuste. Doar astfel poate fi garantată o experiență de utilizare sigură și un nivel adecvat de protecție în fața riscurilor digitale din ce în ce mai sofisticate.

Un mod esențial de a înțelege rolul critic al parolelor în securitatea aplicațiilor software, conform figurii 1.1, este prin prisma triadei CIA, *confidențialitate, integritate și disponibilitate*. Acest model fundamental în securitatea informațională subliniază cele trei principii de bază care trebuie protejate în orice sistem digital. Parolele joacă un rol direct în menținerea confidențialității, deoarece ele sunt primul mecanism de apărare împotriva accesului neautorizat. O parolă puternică asigură că doar utilizatorii legitimi pot accesa informațiile sensibile, protejând astfel datele personale, financiare sau organizaționale împotriva expunerii.

În același timp, parolele contribuie indirect la asigurarea integrității și disponibilității sistemului. Dacă un atacator reușește să compromită o parolă slabă, poate modifica sau șterge date (afectând integritatea) sau poate bloca accesul utilizatorilor legitimi (afectând disponibilitatea). Mai mult, atacurile automatizate asupra parolelor, precum cele de tip brute-force, pot duce la blocarea temporară a conturilor sau a sistemului, punând în pericol disponibilitatea serviciului. Prin urmare, utilizarea unor parole sigure și implementarea unor mecanisme de validare eficiente sunt măsuri esențiale nu doar pentru protecția datelor, ci pentru menținerea echilibrului între cele trei componente esențiale ale securității cibernetice.

Figura 1.1 Modelul CIA în securitatea informațională

1.2 Vulnerabilități comune și consecințele parolelor slabe

Deși parolele sunt cel mai răspândit mijloc de autentificare, ele prezintă o serie de vulnerabilități critice atunci când nu sunt alese și gestionate corespunzător. Parolele slabe, reutilizate sau ușor de ghicit pot fi exploatate cu ușurință de către atacatori prin metode automate și eficiente. În absența unor politici

stricte de validare și protecție, aceste vulnerabilități devin o poartă de acces directă către datele și resursele sensibile ale utilizatorilor.

Una dintre cele mai frecvente vulnerabilități întâlnite în aplicațiile software este acceptarea și utilizarea de parole slabe. Acestea includ parole scurte, formate exclusiv din caractere alfanumerice simple, parole comune (ex: „123456”, „qwerty”, „admin”), precum și parole derivate din date personale ușor de ghicit, cum ar fi numele propriu sau anul nașterii. În lipsa unor politici stricte de validare, aplicațiile permit introducerea unor astfel de parole, care pot fi compromise în doar câteva secunde cu instrumente automatizate.

O altă practică periculoasă este reutilizarea aceleiași parole pe mai multe platforme. Aceasta amplifică riscul unui atac de tip „credential stuffing”, în care atacatorii folosesc baze de date cu parole scurse din alte breșe pentru a obține acces în lanț la conturi diferite. În contextul în care milioane de credențiale sunt disponibile public sau pe piața neagră, reutilizarea parolelor slabe devine o invitație deschisă pentru compromiterea datelor personale sau organizaționale.

Parolele slab protejate pot fi de asemenea vulnerabile la atacuri de tip phishing, unde utilizatorul este păcălit să-și introducă parola pe un site fals care mimează interfața unei aplicații reale. Dacă parola compromisă este simplă și reutilizată, un atacator poate obține acces imediat și nestingherit la mai multe conturi. Aceste tipuri de atacuri sunt din ce în ce mai sofisticate și deseori automatizate, exploatând atât lipsa de educație cibernetică, cât și neglijența în proiectarea interfețelor de autentificare.

Consecințele directe ale acestor vulnerabilități variază de la acces neautorizat la conturi personale, până la pierderi masive de date, șantaj digital (ransomware) sau chiar infiltrarea în infrastructura internă a unei organizații. În cazul companiilor, compromiterea conturilor angajaților poate duce la scurgeri de informații comerciale sensibile, la încălcări ale conformității cu regulamente precum GDPR sau ISO 27001, dar și la afectarea reputației și încrederii clienților.

Pe lângă impactul tehnic și juridic, utilizarea parolelor slabe afectează și costurile operaționale, prin necesitatea resetării frecvente a parolelor, intervenții de suport tehnic și implementarea de măsuri reactive. Din acest motiv, prevenirea acestor vulnerabilități prin generarea automată de parole sigure, validarea lor după criterii stricte și educarea utilizatorilor cu privire la bunele practici devine o necesitate critică în orice aplicație software responsabilă.

1.3 Obiectivele și scopul proiectului

Tema proiectului se înscrie în sfera largă a securității cibernetice și vizează una dintre cele mai critice componente ale protecției digitale: parolele. Într-o eră în care tot mai multe aplicații stochează și procesează date sensibile, autentificarea sigură devine o prioritate absolută. Astfel, proiectul abordează tema generării și validării de parole sigure folosind tehnici de criptografie, cu scopul de a reduce riscul accesului neautorizat și de a îmbunătăți securitatea utilizatorilor finali.

Scopul principal al proiectului este dezvoltarea unei aplicații web funcționale care să permită utilizatorilor generarea de parole sigure, complexe și dificil de compromis, utilizând metode criptografice moderne. Această aplicație oferă o interfață intuitivă pentru configurarea parametrilor doriti (lungime, complexitate, tipuri de caractere) și integrează validatori inteligenți pentru a garanta că parolele respectă cerințele actuale de securitate. Prin combinarea algoritmilor de generare pseudo-aleatoare cu evaluări euristice, aplicația ajută utilizatorii să evite parolele comune, slabe sau redundante.

Pentru atingerea acestui scop, proiectul urmărește mai multe obiective concrete. În primul rând, este necesară identificarea și documentarea cerințelor de securitate pentru parole, prin analiza standardelor internaționale (precum NIST, OWASP) și a celor mai bune practici existente. Această etapă include studiul algoritmilor actuali utilizați în generarea de parole și formularea unui set coerent de reguli pentru validarea acestora.

Al doilea obiectiv al proiectului constă în dezvoltarea propriu-zisă a aplicației web, iar în cadrul acestuia este inclusă și determinarea algoritmilor necesari pentru implementarea funcționalităților-cheie. Acest pas presupune analiza și selectarea metodelor optime de validare și generare a parolelor, în funcție de cerințele de securitate și uzabilitate. Aplicația integrează o componentă robustă de validare, care verifică lungimea minimă a parolelor, detectează parolele comune sau compromise prin comparație cu baze de date externe, și evaluează diversitatea caracterelor utilizate (litere mari, mici, cifre, simboluri). În ceea ce privește generarea parolelor, a fost implementat un algoritm criptografic de generare pseudo-aleatorie, asigurând astfel un grad ridicat de entropie și imprevizibilitate. În plus, aplicația oferă opțiuni avansate de personalizare, inclusiv generarea de parole memorabile, construite pe baza unor modele fonetice sau semantice, care îmbină ușurința de reținere cu un nivel adecvat de securitate. Această abordare echilibrează nevoile utilizatorilor cu cerințele moderne de protecție a datelor.

Al treilea obiectiv esențial îl constituie testarea și validarea aplicației dezvoltate. Această etapă presupune verificarea funcționalității tuturor modulelor, analiza performanței algoritmilor utilizați, precum și evaluarea gradului de securitate oferit de parolele generate. Testele vor include atât scenarii uzuale de utilizare, cât și simulări de atac (ex: brute-force sau dictionary attack) pentru a măsura reziliența parolelor. Rezultatele obținute vor contribui la ajustarea mecanismelor algoritmice și la redactarea unei documentații clare, care să sprijine utilizarea și extinderea ulterioară a aplicației.

II ALGORITMII ȘI TEHNICILE UTILIZATE

MemoPass este o aplicație web interactivă, interfața, care poate fi vizualizată în figura 2.1, care oferă utilizatorului posibilitatea de a genera parole sigure și ușor de reținut, printr-un proces algoritmic compus din patru pași consecutivi: selecție aleatorie criptografică, construcție mnemonică, transformări Leet și evaluarea tăriei. Arhitectura modulară a aplicației a fost concepută pentru a asigura atât funcționarea autonomă direct în browser, printr-o interfață intuitivă și responsivă, cât și posibilitatea de reutilizare a componentelor sale sub formă de librărie JavaScript, oferind astfel dezvoltatorilor externi o soluție flexibilă și ușor de integrat în alte aplicații web, fără a fi necesare modificări majore ale codului sursă. Interfața vizuală ghidează utilizatorul pas cu pas, oferind feedback instant asupra calității parolei.

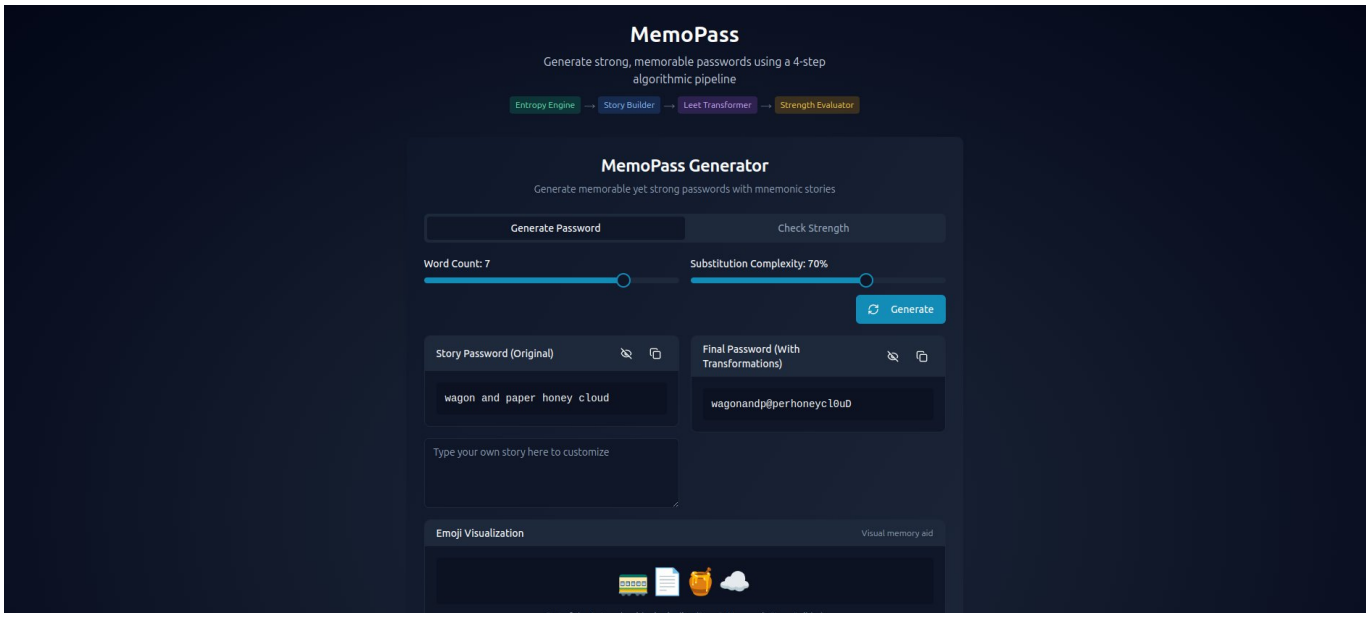

Figura 2.1 Interfața principală a aplicației MemoPass

2.1 Modelul de algoritm compus – pipeline determinist

În centrul arhitecturii aplicației *MemoPass* se află un model de procesare algoritmică secvențială, denumit *pipeline determinist*, care organizează logica funcțională în patru pași bine definiți: selecție criptografică, construcție mnemonică, transformare contextuală și evaluare de tărie. Fiecare dintre acești pași contribuie cu un rol specific la generarea unei parole care să fie atât sigură din punct de vedere tehnic, cât și ușor de reținut pentru utilizator. Modelul reflectă o viziune modernă asupra proiectării aplicațiilor de securitate, în care claritatea logicii, modularitatea și transparența execuției sunt esențiale.

Importanța aplicării acestui model este direct legată de tema și obiectivele proiectului: generarea de parole sigure folosind tehnici avansate de criptografie, dar și asigurarea unei experiențe de utilizare accesibile și eficiente. Pipeline-ul determinist asigură nu doar un control riguros al fiecărui pas, ci și o fundamentare algoritmică solidă, care poate fi explicată, analizată și extinsă în contexte viitoare, spre exemplu, în transformarea aplicației într-o librărie reutilizabilă. În plus, această structură facilitează testarea

individuală a componentelor și creșterea treptată a complexității, pe măsură ce cerințele de securitate devin mai sofisticate.

Aplicația urmează un flux logic secvențial, unde fiecare modul prelucrează datele de ieșire ale celui anterior. Această abordare deterministă garantează că pentru aceleași condiții de input, rezultatul va fi replicabil, favorizând trasabilitatea și auditarea procesului. Mai mult, separarea clară a responsabilităților pe pași permite reutilizarea și extinderea aplicației în alte proiecte , spre exemplu, utilizarea doar a transformatorului Leet sau a evaluatorului de tărie într-o aplicație terță.

Un alt avantaj fundamental este faptul că toate procesele sunt executate local, în browser, fără niciun transfer de date către servere externe. Acest lucru conferă aplicației un grad înalt de confidențialitate, un criteriu esențial în aplicațiile de securitate. Utilizatorul are garanția că nicio parolă generată nu este stocată, transmisă sau expusă în afara mediului de execuție local, ceea ce reduce drastic riscurile legate de interceptarea datelor sau breșe de infrastructură.

Prin această arhitectură, *MemoPass* demonstrează cum un sistem bine gândit din punct de vedere algoritmic poate răspunde simultan cerințelor de securitate cibernetică și nevoilor de uzabilitate, oferind un instrument eficient și scalabil atât pentru utilizatorii individuali, cât și pentru dezvoltatorii care doresc să-l integreze în soluții mai ample.

2.2 Entropia și motorul de selecție

Primul modul al pipeline-ului din aplicația MemoPass îndeplinește un rol fundamental în asigurarea securității procesului de generare a parolelor. Denumit *Entropy & Selection Engine*, acest modul are sarcina de a produce o bază aleatorie sigură, care stă la baza fiecărei parole generate. Fiabilitatea acestei etape este esențială, deoarece orice vulnerabilitate sau lipsă de entropie în datele inițiale poate afecta direct predictibilitatea și, implicit, compromiterea parolei.

Pentru a asigura un nivel ridicat de imprevizibilitate, aplicația utilizează metoda nativă *window.crypto.getRandomValues()* din *Web Crypto API*, o soluție standardizată și aprobată pentru aplicații criptografice. Această metodă generează valori pseudoaleatorii cu sursă de entropie preluată direct din sistemul de operare, oferind rezultate care nu pot fi reproduse cu ușurință. În comparație cu funcții simple precum Math.random(), care sunt deterministe și ușor de anticipat, *getRandomValues()* este adecvat pentru scenarii în care securitatea este o cerință critică.

```
export function generateRandomTokens(wordCount: number = 5): string[] {
  const array = new Uint8Array(16);
  window.crypto.getRandomValues(array);

  const tokens: string[] = [];
  for (let i = 0; i < wordCount; i++) {
    const randomIndex = Math.floor((array[i * 3] * 256 * 256 + array[i * 3 + 1]
* 256 + array[i * 3 + 2]) % dicewareWords.length);
    tokens.push(dicewareWords[randomIndex]);
  }

  return tokens;}
```

Numărul de tokeni generați, care corespund în acest context cuvintele folosite în construcția frazei mnemonic, este configurabil de către utilizator, de obicei între 3 și 7. Această flexibilitate permite adaptarea nivelului de complexitate și lungime a parolei în funcție de nevoile specifice, de la parole scurte pentru autentificări ocazionale până la parole lungi pentru conturi critice. Setul de tokeni rezultat este compus exclusiv din termeni selectați aleator, având o diversitate lingvistică și fonetică ce favorizează formarea ulterioară a unei structuri coerente și memorabile.

Acești tokeni constituie fundația întregului proces algoritmic ce urmează: sunt transformați într-o frază mnemonică coerentă, ulterior modificați vizual prin substituții Leet, și în cele din urmă evaluați din punct de vedere al entropiei și rezistenței la atacuri. Datorită modului în care este construit, *Entropy & Selection Engine* nu doar inițiază procesul de creare a parolei, ci influențează direct toate etapele următoare, făcând din această componentă un element central în arhitectura aplicației MemoPass.

2.3 Constructor mnemonic de parole memorabile

După generarea tokenilor aleatori prin modulul de entropie, aplicația MemoPass trece în etapa de construcție mnemonică, denumită *Mnemonic Story Builder*. Scopul acestui pas este de a transforma un șir de cuvinte izolate într-o frază coerentă, ușor de reținut de către utilizator. Prin selectarea atentă a cuvintelor, se creează o asociere logică sau vizuală care sprijină memorarea naturală, spre deosebire de o parolă formată din caractere complet aleatorii, dificil de reținut și predispusă la a fi uitată sau scrisă undeva nesecurizat.

O privire de ansamblu asupra arhitecturii aplicației și a principalelor componente logice este prezentată în Anexa A, unde sunt ilustrate atât structura modulară a codului, cât și fluxul de date dintre etapele procesării. Funcția *buildMnemonicStory* care are rolul de a construi o propoziție mnemonică (ușor de reținut) și o reprezentare vizuală în format emoji, pornind de la o listă de cuvinte. În prima etapă, se selectează aleatoriu un șablon de propoziție predefinit, care conține locuri rezervate pentru părți de vorbire precum subiect, verb, complement etc. Apoi, cuvintele primite sunt mapate la aceste roluri gramaticale pe baza unor categorii prestabilite. În cazul în care unele locuri nu pot fi completate din prima încercare, se face o a doua trecere în care restul tokenilor disponibili sunt atribuiți categoriilor rămase. După ce toate locurile din șablon sunt completate, șablonul este transformat într-o propoziție coerentă. În final, propoziția este convertită într-o secvență de emoji-uri, folosind o mapare specifică (wordToEmoji), care oferă o reprezentare vizuală compactă și memorabilă. Această metodă permite generarea de conținut mnemonic personalizat și reutilizabil, ideal pentru aplicații care implică memorare sau autentificare prin asociere semantică.

Această strategie se bazează pe asocierea semantică a cuvintelor și pe avantajele memoriei vizuale și contextuale. Fiecare cuvânt selectat este ales astfel încât, alături de celelalte, să formeze o secvență plăcută din punct de vedere fonetic și sugestivă din punct de vedere mental. Această abordare mnemonică este inspirată din metode de învățare accelerate, care au demonstrat eficiență în stocarea și reamintirea de in-

formații complexe. Utilizatorul nu mai este nevoit să rețină o combinație abstractă de caractere, ci poate vizualiza o mini-poveste sau o succesiune de imagini mentale care se consolidează natural în memorie. În acest mod, se reduce semnificativ riscul uitării parolei și, implicit, comportamentele riscante asociate cu recuperarea sau notarea nesecurizată.

În plus, aplicația oferă utilizatorilor mai avansați posibilitatea de a introduce manual o frază mnemonică proprie, folosind un câmp dedicat de tip textarea. Acest lucru le oferă control complet asupra semnăturii semantice a parolei, permițând folosirea unor termeni personali, familiari sau profesional relevanți. Deși fraza introdusă manual urmează aceleași pipeline ca și cele generate automat, posibilitatea de personalizare extinde funcționalitatea aplicației dincolo de generare aleatorie, oferind o experiență adaptabilă stilului fiecărui utilizator.

Prin acest pas, MemoPass reușește să combine securitatea cu uzabilitatea, un obiectiv greu de atins în domeniul securității informaționale. *Mnemonic Story Builder* reprezintă astfel o componentă esențială care facilitează crearea de parole complexe, dar memorabile, reducând dependența utilizatorului de notițe externe sau manageri de parole tradiționali.

Decizia de a include o etapă dedicată construcției mnemonicii în cadrul procesului de generare a parolelor a fost motivată de nevoia de a găsi un echilibru între securitate și memorabilitate. Studiile din domeniul psihologiei cognitive și securității utilizatorului arată că parolele compuse exclusiv din caractere aleatorii tind să fie uitate mai ușor, ceea ce conduce adesea la comportamente nesigure, cum ar fi reutilizarea parolelor, scrierea acestora pe hârtie sau salvarea lor în medii neprotejate. Prin introducerea unei fraze cu sens, creată din cuvinte uzuale și ușor de asociat, aplicația reduce această problemă și transformă experiența generării parolei într-un proces intuitiv. Astfel, nu doar se crește reziliența parolei în fața atacurilor automatizate, ci se facilitează și adoptarea de practici mai bune de securitate de către utilizatorii obișnuiți.

2.4 Transformator Leet contextual

După generarea frazei mnemonice, aplicația trece la etapa de transformare vizuală, în care șirul de caractere este modificat printr-un algoritm de tip *Leet speak*. Acest proces constă în înlocuirea anumitor caractere uzuale (precum a, e, i, o, s, t) cu simboluri echivalente (@, 3, 1, 0, $, 7), crescând astfel diversitatea vizuală a parolei. Spre deosebire de transformările rigide, acest algoritm aplică regulile de substituție în mod condiționat, în funcție de un nivel de complexitate selectat de utilizator.

Parametrul de control este implementat sub forma unui slider numeric, denumit "*Substitution Complexity*", ce permite reglarea intensității transformărilor de la 0% la 100%. Cu cât valoarea este mai mare, cu atât mai multe caractere sunt înlocuite. Acest mecanism introduce o doză de aleatoriu controlat, care asigură că nici două parole generate cu același text de bază nu vor arăta identic, menținând în același timp lizibilitatea. Astfel, parola finală rezultată devine mai greu de spart prin metode automatizate, dar nu devine complet de neînțeles pentru utilizator. De asemenea, funcția aplică o capitalizare aleatoare a unui ca-

racter pentru a introduce o variație suplimentară. Implementarea detaliată a acestui mecanism este prezentată în Anexa B – Algoritm de conversie Leet pentru întărirea parolelor.

Scopul principal al acestei transformări este dublu: pe de o parte, se urmărește creșterea entropiei vizuale, adică a variabilității caracterelor care alcătuiesc parola, iar pe de altă parte, se dorește reducerea predictibilității în fața algoritmilor de spargere prin forță brută sau analiză de patternuri. Prin folosirea de simboluri neconvenționale, parola capătă o structură mai dificil de anticipat, mai ales dacă este combinată cu un set de tokeni aleși aleator și o lungime suficientă.

Un aspect important este că această transformare nu este impusă în mod absolut, ci adaptabilă la preferințele și nivelul de confort al utilizatorului. De exemplu, pentru o parolă care trebuie reținută frecvent, utilizatorul poate selecta un nivel redus de complexitate pentru a păstra lizibilitatea, iar pentru parole folosite rar sau în medii cu cerințe ridicate de securitate, poate opta pentru un nivel înalt de transformare. Acest control asupra transformării contribuie la personalizarea experienței și la creșterea încrederii utilizatorului în propria parolă.

Prin integrarea acestei etape, MemoPass nu doar generează o parolă cu aspect complex, ci și oferă o soluție scalabilă și transparentă, în care securitatea este construită nu prin impunere, ci prin alegere informată. Transformările Leet contextualizate joacă astfel un rol cheie în crearea unei parole echilibrate: dificil de spart, dar accesibilă mintal, mai ales atunci când este folosită împreună cu fraza mnemonică. Această abordare face parte din filozofia generală a aplicației, aceea de a combina criptografia cu uzabilitatea într-un mod inteligent și adaptabil.

2.5 Evaluator de tărie (entropie și euristici)

Un element central în evaluarea tăriei parolei este entropia, un concept preluat din teoria informației, introdus de matematicianul și inginerul Claude Shannon în lucrarea sa din 1948, "*A Mathematical Theory of Communication*". În acest context, entropia reprezintă măsura incertitudinii asociate unei surse de informație, adică gradul de imprevizibilitate al unui șir de caractere. Cu cât o parolă are mai multă entropie, cu atât este mai greu de ghicit sau generat automat de către un atacator, deoarece numărul de posibilități unice crește exponențial. Entropia parolelor se măsoară în biți, iar fiecare caracter adăugat, dacă este selectat dintr-un set divers, contribuie la creșterea valorii totale. De exemplu, o parolă aleatoare formată din 12 caractere alfanumerice poate avea peste 70–80 de biți de entropie, ceea ce corespunde unui spațiu de căutare de peste 2^{70} posibilități. În aplicația MemoPass, entropia este calculată în timp real, pe baza structurii parolei și a varietății caracterelor utilizate, și este afișată împreună cu o estimare a rezistenței în fața unui atac de tip brute-force.

Evaluarea securității parolelor generate sau introduse de utilizator se realizează printr-un algoritm dedicat, care analizează mai mulți factori relevanți: lungimea parolei, diversitatea caracterelor (litere mari, mici, cifre, simboluri), prezența unor termeni comuni, precum și nivelul de entropie calculat pe baza ca-

racterelor unice. Pe baza acestor informații, se estimează un scor numeric și se clasifică parola într-o categorie de tărie (ex. „Weak”, „Strong”, „Very Strong”), iar timpul estimativ necesar pentru a o sparge prin atacuri brute-force este exprimat într-un format ușor de înțeles. Implementarea completă a acestei logici este prezentată în Anexa C – Algoritm de evaluare a complexității și rezistenței parolelor.

Algoritmul de evaluare ia în considerare o serie de factori cuantificabili, printre care: lungimea totală a parolei, prezența și proporția caracterelor din clase diferite (litere mari, mici, cifre, simboluri), repetitivitatea secvențelor, dar și frecvența unor șabloane comune (ex: „123”, „password”, „qwerty”). Aceste caracteristici sunt corelate cu un scor de entropie calculat în biți, care reflectă gradul de imprevizibilitate al parolei, cu cât entropia este mai mare, cu atât este mai dificil pentru un atacator să o ghicească sau să o genereze prin metode automatizate.

Pe baza acestor factori, aplicația atribuie parolei un scor final pe o scară de la 0 la 4, cu valori intermediare (ex: 3.5/4). Acest scor este acompaniat de o estimare a timpului necesar pentru compromitere prin brute-force, exprimată într-o unitate relevantă, pentru a facilita înțelegerea impactului practic. Astfel, utilizatorul nu doar primește o parolă „validă”, ci înțelege nivelul real de protecție oferit de acea parolă în fața metodelor moderne de atac.

Evaluarea este afișată în interfață printr-o bară de progres colorată, care se modifică în timp real în funcție de scorul obținut, precum și printr-un text informativ care explică punctele tari și slabe ale parolei. Acest sistem vizual oferă feedback imediat și intuitiv, stimulând învățarea și încurajând utilizatorul să ajusteze parametrii până când obține o parolă suficient de robustă. În acest sens, evaluatorul devine un instrument educativ, nu doar unul de verificare.

Prin această componentă, MemoPass se diferențiază de alte aplicații care generează parole fără justificare sau transparență. Algoritmul de scorare implementat nu doar validează rezultatul final, ci contribuie activ la construirea unei relații de încredere între utilizator și sistem, oferindu-i acestuia o explicație clară a motivelor pentru care o parolă este considerată sigură sau nu. Astfel, securitatea devine un proces ghidat, interactiv și adaptat fiecărui context de utilizare.

2.6 Bucla de reîncercare și condițiile de generare

În unele situații, parolele generate nu îndeplinesc standardele minime de securitate, chiar dacă procesul de generare este corect din punct de vedere logic. Acest lucru poate apărea fie din cauza alegerilor utilizatorului, cum ar fi un număr redus de cuvinte sau o complexitate scăzută a transformărilor, fie din cauza naturii aleatorii a generării inițiale, care poate produce ocazional combinații previzibile sau neechilibrate. Pentru a elimina riscul livrării unei parole slabe, aplicația MemoPass introduce un mecanism de control automat al calității, integrat în fluxul de generare.

Procesul complet de generare a unei parole în aplicație este automatizat și structurat în patru etape esențiale: selectarea aleatorie a unui set de cuvinte, construirea unei propoziții mnemonice, aplicarea trans-

formărilor de tip Leet și evaluarea finală a nivelului de securitate. Acest flux este conceput pentru a asigura un echilibru între memorabilitate și rezistență la atacuri brute-force, iar generarea se repetă automat până la obținerea unui scor de complexitate acceptabil. Implementarea detaliată a acestui proces, împreună cu mecanismele intermediare și criteriile de validare, este prezentată în Anexa D – Proces automatizat de generare, transformare și validare a parolelor mnemonic-securizate.

Un aspect important al acestui proces este faptul că totul se desfășoară în fundal, fără a întrerupe interacțiunea utilizatorului cu aplicația. Generarea repetată are loc instantaneu, fără întârzieri perceptibile sau mesaje de eroare, oferind o experiență fluidă și prietenoasă. Utilizatorul vede doar parola finală, care a fost deja validată, și este informat despre scorul acesteia, fără a fi expus la complexitatea procesului din spate. Astfel, aplicația combină eficiența algoritmică cu simplitatea experienței vizuale. Prin introducerea acestei bucle inteligente de reîncercare, MemoPass oferă o garanție implicită de securitate, asigurând că nicio parolă livrată nu este sub standardele minime stabilite. În plus, acest mecanism previne generarea de parole care, deși valide din punct de vedere structural, ar putea fi vulnerabile la atacuri datorită unui context slab de entropie sau unei diversități reduse a caracterelor. Prin urmare, bucla de reîncercare nu este doar o optimizare tehnică, ci o măsură activă de protecție în designul funcțional al aplicației.

Condițiile de declanșare a buclei sunt definite clar și se bazează pe scorul euristic (ex. sub 2.5 din 4) și entropia minimă acceptată (ex. sub 70 biți). Atunci când aceste praguri nu sunt atinse, aplicația generează automat o nouă combinație de tokeni și reia procesul de transformare și evaluare. Din punct de vedere tehnic, această logică este implementată printr-o structură iterativă de tip do-while, care asigură repetarea pașilor până la satisfacerea tuturor cerințelor de tărie.

```
 do {
    stages.stage1 = `Generated ${wordCount} random tokens`;
    originalWords = generateRandomTokens(wordCount);
    stages.stage2 = `Created story: "${originalWords.join(', ')}" → senten-
ce`;
    story = buildMnemonicStory(originalWords);
    stages.stage3 = `Applied transformations (complexity: ${Math.round(com-
plexity * 100)}%) to "${story.sentence}"`;
    transformedPassword = applyLeetTransformations(story.sentence, complexi-
ty);
    strengthEvaluation = evaluatePasswordStrength(transformedPassword);
    stages.stage4 = `Evaluated strength: ${strengthEvaluation.score}/4
(${strengthEvaluation.strengthText})`;
    score = strengthEvaluation.score;
    if (score < 3) {
      stages.stage1 = "Restarting due to low strength score...";
    }
  } while (score < 3);
```

Această abordare oferă un avantaj esențial: garanția că utilizatorul nu primește niciodată o parolă slabă, indiferent de alegerile făcute sau de setările implicite. Chiar dacă utilizatorul setează un nivel redus de transformare Leet sau un număr mic de cuvinte în fraza mnemonică, aplicația are capacitatea de a ajusta automat și dinamic acești parametri, în limitele stabilite, pentru a menține standardele de siguranță.

De asemenea, bucla permite generarea în fundal a mai multor iterații, fără a necesita intervenție manuală din partea utilizatorului. Acest comportament este ideal în contextul aplicațiilor moderne, unde simplitatea experienței este la fel de importantă ca nivelul de protecție oferit. Utilizatorul observă doar rezultatul final – o parolă acceptabilă și validată – fără să fie conștient de numărul de tentative anterioare respinse automat.

Prin această logică robustă și reactivă, MemoPass adaugă un strat suplimentar de protecție, consolidând misiunea aplicației de a livra exclusiv parole sigure, fără compromisuri. Bucla de reîncercare devine astfel un mecanism de corecție inteligentă, capabil să compenseze aleatorietatea excesivă sau alegerile suboptime, păstrând în același timp controlul și transparența în procesul de generare.

2.7 Analiza comparativă a tehnicilor utilizate

În cadrul aplicației MemoPass, procesul de generare a parolelor este alcătuit dintr-o succesiune logică de pași, fiecare bazat pe tehnici algoritmice distincte, alese în mod deliberat pentru a echilibra securitatea, uzabilitatea și personalizarea. Pentru a evidenția eficiența soluției propuse, este importantă o analiză comparativă între metodele aplicate în acest pipeline și alte tehnici consacrate utilizate în aplicații de generare a parolelor. Spre deosebire de generatorii clasici, care produc șiruri de caractere complet aleatorii, MemoPass utilizează o abordare compusă și semantică, începând cu selecția entropică a tokenilor (bazată pe *window.crypto.getRandomValues()*), urmată de construirea unei fraze mnemonice, transformări contextuale și evaluare euristică. Această arhitectură oferă un avantaj semnificativ în ceea ce privește memorabilitatea parolelor, aspect adesea ignorat în generatoarele tradiționale, care se concentrează exclusiv pe complexitate tehnică.

```
export function generateRandomTokens(wordCount: number = 5): string[] {
  const array = new Uint8Array(16);
  window.crypto.getRandomValues(array);

  const tokens: string[] = [];
  for (let i = 0; i < wordCount; i++) {
    const randomIndex = Math.floor((array[i * 3] * 256 * 256 + array[i * 3 + 1]
* 256 + array[i * 3 + 2]) % dicewareWords.length);
    tokens.push(dicewareWords[randomIndex]);
  }

  return tokens;}
```

Metoda *Diceware* este o tehnică consacrată pentru generarea de parole sigure și ușor de reținut, bazată pe alegerea aleatorie a cuvintelor dintr-o listă special concepută, care conține 7776 de intrări distincte. Fiecare cuvânt din această listă este asociat unui cod format din cinci cifre între 1 și 6, corespunzător rezultatelor obținute prin aruncarea a cinci zaruri. Astfel, pentru a genera o parolă Diceware de 5 cuvinte, se fac 5×5 = 25 aruncări de zar, ceea ce oferă o combinație cu un nivel ridicat de entropie (peste 60 de biți), suficient pentru a rezista atacurilor brute-force moderne. Avantajul major al Diceware este că rezultatul, o succesiune de cuvinte precum *"laser planet coffee yellow bridge"*, este mult mai ușor de memorat decât o parolă aleatoare formată din caractere, păstrând totodată un grad ridicat de securitate.

Totuși, Diceware are o limitare esențială: deși entropia este matematic ridicată, parolele generate pot fi recunoscute ca secvențe de cuvinte reale, ceea ce le face vulnerabile în fața unor atacatori care folosesc dicionare extinse și modele de limbaj. În acest context, MemoPass aduce o îmbunătățire importantă: pe lângă alegerea aleatorie a cuvintelor, aplicația aplică transformări adaptabile asupra propoziției generate, în special transformări Leet contextualizate, care înlocuiesc caracterele în funcție de context și de un nivel de complexitate controlabil. Astfel, parolele rezultate devin mai rezistente la recunoașterea automată, păstrând însă o anumită lizibilitate pentru utilizator. Prin combinarea structurii mnemonice (care ușurează reținerea) cu metode de obfuscare inteligentă, MemoPass depășește modelul Diceware în termeni de securitate practică în medii moderne.

```
const dicewareList: Record<string, string> = {
  "11111": "apple",
  "11112": "bridge",
  "11113": "coffee",
  "11114": "drive",
  "11115": "earth",
  "11116": "forest",
  "11121": "galaxy",
  "11122": "hammer",
  "11123": "island",
  "11124": "jungle",
  "11125": "laser",
  "11126": "mountain",
  ...
};
```

Comparativ cu soluții de tip zxcvbn (utilizate pentru validarea parolelor, nu pentru generarea lor), MemoPass încorporează funcționalitatea de scorare euristică internă, oferind un feedback direct, fără dependență de biblioteci externe. Funcția zxcvbn, utilizată pentru evaluarea tăriei parolelor, aplică un model euristic avansat care combină analiza lingvistică, recunoașterea de modele comune și estimări probabilistice pentru a determina cât de ușor ar putea fi spartă o parolă. În loc să se bazeze doar pe entropia teoretică, zxcvbn segmentează parola în componente recognoscibile și estimează cea mai probabilă strategie de spargere, simulând un atac real. Pe baza acestor analize, funcția returnează un scor de complexitate de la 0 la 4, entropia totală în biți, timpul estimat necesar pentru spargere în mai multe scenarii și oferă utilizatorului recomandări pentru îmbunătățirea parolei. Această abordare face ca zxcvbn să fie o soluție eficientă pentru evaluarea practică a securității parolelor în aplicații moderne. În plus, MemoPass rulează 100% local, fără stocare sau transmitere de date către servere externe, ceea ce elimină riscurile legate de confidențialitate, o preocupare frecventă în rândul utilizatorilor conștienți de securitate.

Mai mult, prin modul său iterativ de generare, care reia automat procesul până la obținerea unei parole suficient de sigure, MemoPass oferă o garanție implicită de calitate, eliminând complet riscul ca utilizatorul să primească o parolă slabă. Acest comportament nu doar că întărește securitatea, dar reduce și presiunea asupra utilizatorului, care nu trebuie să cunoască detalii tehnice precum entropia, scorurile euristice sau tipologia atacurilor cibernetice. Prin abstractizarea acestor elemente într-un proces automatizat,

aplicația contribuie la democratizarea accesului la practici de securitate avansate, fără a compromite nivelul de protecție oferit.

Această automatizare inteligentă, combinată cu opțiunile extinse de personalizare, face ca MemoPass să răspundă eficient nevoilor unei game largi de utilizatori, de la începători fără cunoștințe tehnice până la utilizatori avansați sau profesioniști în domeniul IT. În plus, faptul că toate operațiile se desfășoară local, în browser, fără transmiterea de date către servere externe, consolidează încrederea utilizatorului în aplicație și o face potrivită pentru medii cu cerințe stricte de confidențialitate.

Din perspectiva rezilienței algoritmice, MemoPass integrează mai multe straturi de siguranță – de la generare criptografică, la transformare contextuală și evaluare dinamică –, toate coordonate într-un sistem coerent și reactiv. Această structură stratificată face ca aplicația să se adapteze automat la situații diverse, compensând aleatorietatea cu validare și corectare automată.

Tabelul 2.1 - Analiza comparativa a tehnicilor utilizate

Criteriu	**MemoPass**	**Diceware**	**xcvbn (Dropbox)**	**LastPass Generator**
Tip funcționalitate	Generare + evaluare + personalizare	Doar generare	Doar evaluare	Generare simplă
Metodă de generare	Entropie criptografică + mnemonică	Aleatorie cu liste predefinite	N/A	Aleatorie (configurabilă)
Memorabilitate	Ridicată (fraze + emoji + Leet)	Ridicată (fraze)	Scăzută	Scăzută
Transformări avansate	Da (Leet contextual, configurabil)	Nu	Nu	Limitat (doar simboluri)
Evaluare tărie	Internă (entropie + euristici)	Nu	Da	Da (scor simplificat)
Interfață vizuală educativă	Da (feedback, scor, emoji)	Nu (de obicei CLI sau statică)	Parțial	Minimală
Execuție locală completă	Da	Da	Parțial	Nu (în cloud, necesită cont)
Personalizare utilizator	Ridicată (slidere, inputuri)	Medie (număr de cuvinte)	N/A	Medie (lungime, tip caractere)
Scop educațional	Da	Nu	Parțial	Nu

Interpretarea comparativă a soluțiilor analizate evidențiază faptul că MemoPass oferă cel mai bun echilibru între siguranță, memorabilitate și personalizare, datorită arhitecturii sale modulare, a transformărilor configurabile și a feedback-ului vizual educativ. Soluții precum *Diceware* se remarcă prin simpli-

tate și eficiență în generarea de parole ușor de reținut, însă nu includ transformări avansate sau mecanisme de evaluare automată a tăriei. În contrast, *zxcvbn* este foarte eficient pentru validarea parolelor existente, oferind o analiză euristică detaliată, dar nu are capacitate de generare. Pe de altă parte, LastPass Generator oferă funcționalitate de bază pentru crearea de parole puternice, dar se dovedește a fi mai rigid, dependent de infrastructura cloud și mai puțin transparent în procesul de scorare și entropie. Aceste diferențe subliniază avantajele unei soluții integrate, locale și educative, cum este MemoPass.

Dincolo de analiza funcționalităților, complexitatea reală a unui algoritm de generare a parolelor poate fi măsurată și în termeni matematici, prin evaluarea entropiei și estimarea numărului de combinații posibile care pot fi generate. Complexitatea unei metode de generare a parolelor poate fi înțeleasă prin numărul total de combinații posibile pe care le poate produce și prin cât de greu este pentru un atacator să le anticipeze sau să le spargă. În cazul metodei Diceware, parolele sunt formate dintr-o succesiune aleatoare de cuvinte extrase dintr-o listă standard de aproximativ 7776 termeni. Chiar și cu un număr mic de cuvinte (de exemplu, cinci), se obține un spațiu foarte mare de combinații posibile, ceea ce face ca parolele să fie teoretic sigure împotriva atacurilor brute-force. Această metodă este însă vulnerabilă atunci când atacatorii folosesc dicționare extinse sau modele care identifică șiruri de cuvinte reale.

Spre deosebire de Diceware, MemoPass îmbunătățește semnificativ complexitatea parolei printr-un proces în mai multe etape. După generarea cuvintelor de bază, aplicația construiește o propoziție mnemonică, aplică transformări de tip Leet într-un mod inteligent și contextualizat, introduce simboluri și majuscule aleatorii și în final evaluează puterea parolei. Această abordare face ca numărul de variante distincte ale aceleiași parole să crească exponențial. Astfel, chiar dacă un atacator ar identifica structura de bază a propoziției, aplicarea acestor transformări suplimentare îl obligă să ia în calcul un număr mult mai mare de versiuni posibile, ceea ce crește semnificativ timpul necesar pentru a sparge parola.

Un alt avantaj major al MemoPass este că aceste transformări nu sunt aplicate haotic, ci în funcție de un parametru de complexitate care poate fi ajustat. Astfel, utilizatorii pot alege un echilibru între lizibilitate și securitate. În plus, analiza finală a parolei, care ține cont de varietatea caracterelor, de lungime și de entropia estimată, oferă un scor de tărie și un timp estimativ de spargere, ceea ce ajută utilizatorul să înțeleagă riscurile reale și să își ajusteze parola în mod informat.

Astfel față de metodele tradiționale care oferă doar o bază sigură, dar previzibilă, MemoPass introduce o arhitectură de generare stratificată și adaptabilă, care duce la o creștere semnificativă a complexității parolelor și la o protecție sporită împotriva atacurilor automatizate.

III REALIZAREA SISTEMULUI

3.1 Cerințe funcționale și nefuncționale

Pentru ca aplicația *MemoPass* să răspundă în mod eficient atât nevoilor reale ale utilizatorilor, cât și exigențelor impuse de standardele actuale de securitate cibernetică, a fost necesară definirea clară și structurată a unui set de cerințe funcționale și nefuncționale. Aceste cerințe au stat la baza procesului de proiectare și implementare, asigurând că aplicația nu doar îndeplinește scopul său tehnic, generarea de parole sigure, ci oferă și o experiență de utilizare intuitivă, rapidă și sigură. Stabilirea acestor cerințe a fost esențială pentru a alinia dezvoltarea aplicației cu obiectivele proiectului, respectiv crearea unei soluții care generează parole robuste, ușor de memorat, adaptabile preferințelor utilizatorului, și imposibil de compromis prin metode automate de atac.

Cerințele funcționale au vizat comportamentul concret al aplicației, interacțiunile directe cu utilizatorul și rezultatele livrate, precum generarea parolelor, aplicarea transformărilor Leet, sau evaluarea tăriei. Pe de altă parte, cerințele nefuncționale au vizat aspecte precum performanța aplicației, siguranța procesării locale, compatibilitatea cu diferite platforme și experiența generală de utilizare. Prin definirea acestor cerințe, *MemoPass* a fost conceput nu doar ca o simplă unealtă tehnică, ci ca un produs software complet, coerent și scalabil, pregătit pentru extindere viitoare sau integrare în aplicații externe.

Cerințele funcționale includ:

- generarea de parole pe baza unei fraze mnemonice (originală sau auto-generată);
- aplicarea de transformări stil Leet în funcție de un nivel de complexitate selectabil;
- evaluarea în timp real a tăriei parolei (cu scor, entropie și estimare brute-force);
- vizualizarea emoji ca sprijin pentru memorare;
- posibilitatea de personalizare a parolei prin ajustarea numărului de cuvinte și a complexității;
- opțiunea de copiere rapidă a parolei în clipboard.

Cerințele nefuncționale includ:

- executarea locală completă, fără transmiterea datelor către server;
- răspuns în timp real, fără întârzieri percepute de utilizator (timp de răspuns sub 100 ms);
- compatibilitate cu browsere moderne;
- interfață intuitivă, prietenoasă, responsive și accesibilă.

3.2 Arhitectura soluției software

Aplicația *MemoPass* este construită pe baza unei arhitecturi modulare de tip client-side, în care toate operațiile sunt executate local, în browserul utilizatorului. Această alegere arhitecturală oferă multiple avantaje: eliminarea dependențelor de server, asigurarea confidențialității datelor, reducerea latenței și portabilitatea crescută în medii diverse (web, desktop progresiv etc.). Prin eliminarea completă a procesării externe, aplicația oferă un nivel de protecție semnificativ împotriva interceptărilor, atacurilor de tip „man-

in-the-middle” și scurgerilor de date. Sistemul este organizat logic în patru module principale, corespondente celor patru pași ai pipeline-ului algoritmic:

- generator de entropie – responsabil de selectarea aleatoare a cuvintelor pe baza unei surse criptografice (window.crypto.getRandomValues()).
- constructor mnemonic – generează o frază coerentă și memorabilă, care acționează ca bază semantică a parolei.
- transformator contextual (Leet) – aplică transformări selective asupra parolei pentru a-i crește complexitatea fără a compromite lizibilitatea.
- evaluator de tărie – evaluează parola finală în funcție de entropie, diversitate și scoruri euristice, oferind feedback vizual și tehnic.

Fiecare modul din aplicație este izolat logic, ceea ce înseamnă că este proiectat să îndeplinească o singură responsabilitate bine definită, fără a depinde de implementarea celorlalte componente. Această separare clară asigură un design modular, care facilitează întreținerea codului, testarea independentă a fiecărui modul și extinderea flexibilă a aplicației prin adăugarea de noi funcționalități fără a afecta sistem.

Arhitectura modulară a aplicației aduce beneficii semnificative în ceea ce privește testarea individuală a componentelor. Separarea clară a responsabilităților permite ca fiecare modul să fie verificat izolat, prin teste unitare dedicate, fără a depinde de comportamentul altor părți ale sistemului. De exemplu, generatorul de entropie poate fi testat independent de modulul care evaluează tăria parolei, ceea ce duce la o detecție mai precisă a eventualelor erori și la o depanare mai eficientă. Această abordare contribuie la creșterea fiabilității aplicației, deoarece asigură că fiecare unitate funcționează corect în mod autonom.

Pe lângă avantajele legate de testare, modularitatea facilitează reutilizarea componentelor în alte aplicații sau contexte. Fiecare modul poate fi extras și utilizat separat, fie ca funcție standalone, fie ca parte a unui pachet importabil. Spre exemplu, transformatorul Leet, responsabil de aplicarea unor reguli complexe asupra parolei, poate fi integrat cu ușurință într-un sistem educațional pentru predarea securității digitale sau într-un formular de creare cont care necesită validări avansate. Astfel, logica dezvoltată nu rămâne captivă într-o singură aplicație, ci poate fi valorificată în mai multe scenarii.

Modularitatea oferă și un grad ridicat de extensibilitate, făcând posibilă adăugarea de noi etape în fluxul de generare și evaluare a parolelor, fără a afecta funcționalitățile existente. Arhitectura permite introducerea unor funcționalități suplimentare, precum integrarea cu un manager de parole, generarea de coduri pentru autentificare cu doi factori (2FA), sau verificarea parolelor împotriva unor baze de date compromise. Toate acestea pot fi realizate fără rescrierea codului existent, datorită unei organizări bine gândite care încurajează dezvoltarea incrementală și adaptabilă la noi cerințe.

Interfața utilizatorului interacționează cu aceste module printr-un strat de orchestrare (Controller), care gestionează inputul utilizatorului (ex: ajustarea complexității sau introducerea manuală a unei povești), declanșează secvențial execuția pașilor pipeline-ului și actualizează interfața cu rezultatele finale.

3.3 Tehnologii utilizate (JavaScript, criptografie, UI/UX)

Pentru implementarea aplicației MemoPass, au fost selectate tehnologii moderne și ușor portabile, care permit rularea completă a aplicației pe partea de client (client-side), fără a necesita un backend. Alegerea acestor tehnologii a avut în vedere trei criterii esențiale: siguranță, performanță și compatibilitate multiplatformă.

Limbajul principal utilizat pentru logica aplicației este JavaScript (ES6+), care a permis implementarea funcționalităților de generare a parolelor, aplicare a transformărilor stil Leet, validare a tăriei parolei și gestionare a interacțiunilor cu utilizatorul. Datorită suportului nativ în toate browserele moderne, JavaScript asigură rularea rapidă și fără dependențe externe.

Pentru generarea parolelor într-un mod sigur și imprevizibil, aplicația utilizează Web Crypto API, mai exact metoda *window.crypto.getRandomValues()*. Acest API furnizează o sursă de entropie criptografică de înaltă calitate, validată pentru utilizare în aplicații sensibile la securitate. Prin această abordare, se evită utilizarea generatoarelor pseudoaleatoare nesigure precum *Math.random()*.

Structura vizuală a aplicației este realizată folosind HTML5 și CSS3, asigurând o separare clară între logică și prezentare. Pentru construirea unei interfețe moderne, responsive și ușor de întreținut, aplicația este dezvoltată folosind *framework-ul React*, care permite componentizarea codului, actualizarea eficientă a interfeței prin virtual DOM și o organizare clară a funcționalităților. React oferă un ecosistem matur, extensibil și potrivit pentru aplicații interactive precum MemoPass, unde starea utilizatorului trebuie actualizată rapid pe măsură ce acesta interacționează cu generatorul de parole.

Pentru stilizarea aplicației și dezvoltarea rapidă a UI-ului, a fost integrat framework-ul Tailwind CSS, un sistem de utilitare CSS care permite definirea stilurilor direct în clasele HTML/JSX. Tailwind oferă o abordare declarativă și predictibilă pentru layout, culoare, tipografie și responsivitate, reducând necesitatea scrierii de fișiere CSS dedicate. În combinație cu Flexbox și CSS Variables, Tailwind permite un control fin asupra alinierii elementelor, spațierii, animațiilor și temării aplicației. Astfel, MemoPass reușește să livreze o interfață coerentă, modernă și ușor de adaptat pe orice dispozitiv.

O caracteristică distinctivă a aplicației este utilizarea emoji-urilor și a codurilor Unicode pentru a reprezenta parolele mnemonice într-o formă vizuală atractivă. Această abordare ajută utilizatorul să asocieze mai ușor componentele parolei cu imagini sugestive, îmbunătățind astfel memorabilitatea fără a compromite complexitatea. Pentru elementele grafice interactive, cum ar fi pictogramele (ex: butonul de copiere, ascunderea parolei), au fost utilizate biblioteci moderne de interfață precum FontAwesome și Lucide. Acestea oferă seturi coerente de iconuri scalabile și integrate nativ cu HTML/CSS.

Toate aceste tehnologii au fost alese cu scopul de a permite funcționarea completă în regim offline, eliminând necesitatea unei conexiuni permanente la internet și reducând semnificativ vectorii de atac. În același timp, acestea asigură o portabilitate ridicată, permițând rularea aplicației în medii diverse, de la browsere desktop până la aplicații mobile progresive (PWA).

3.4 Interfața de utilizator și opțiunile de configurare

Interfața aplicației *MemoPass* este concepută cu accent pe claritate, interactivitate și personalizare, pentru a facilita generarea de parole sigure chiar și pentru utilizatorii fără cunoștințe tehnice. Designul este minimalist, intuitiv și bazat pe componente vizuale bine delimitate, oferind feedback în timp real în funcție de opțiunile selectate. Interfața este complet responsive și poate fi utilizată fără dificultăți pe desktop, tabletă sau telefon mobil. Utilizatorul are la dispoziție o serie de controale personalizabile care influențează modul în care este generată parola: poate selecta numărul de cuvinte utilizate în fraza mnemonică, folosind un slider interactiv; are posibilitatea de a ajusta nivelul de complexitate a transformărilor Leet, exprimat procentual (0–100%), ceea ce afectează gradul de substituție a caracterelor; poate introduce manual propria frază mnemonică într-un câmp dedicat, pentru un control complet asupra semnăturii de bază a parolei;

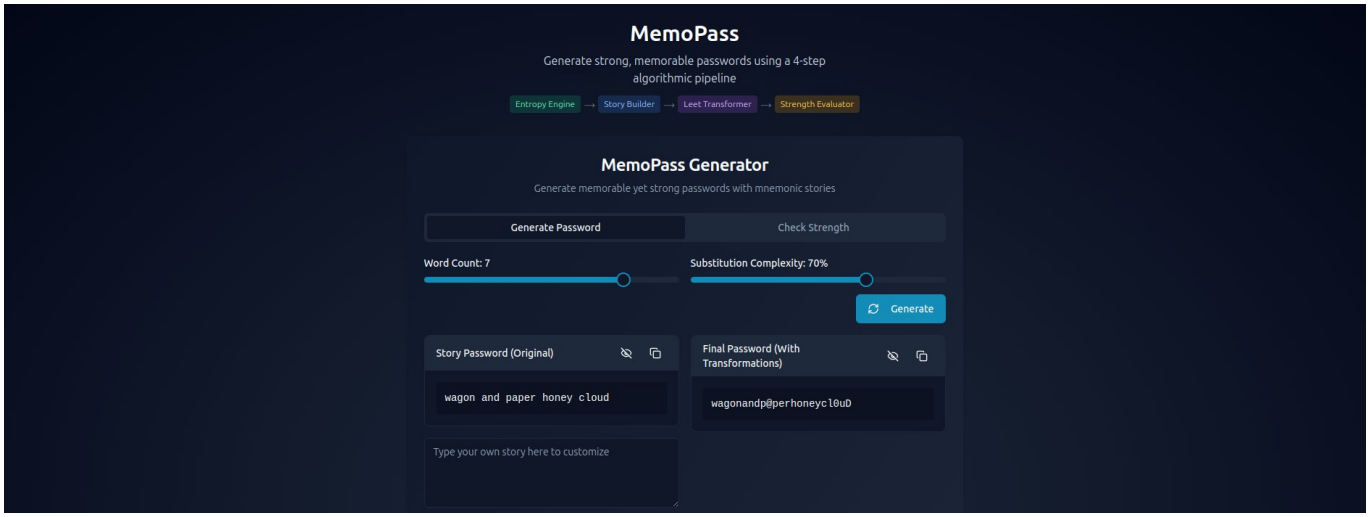

Figura 3.4.1 Interfața principală a aplicației – selectarea opțiunilor de generare

În centrul interfeței aplicației MemoPass sunt amplasate două câmpuri esențiale care reflectă procesul etapizat de creare a unei parole puternice, dar ușor de reținut. Aceste câmpuri nu doar afișează rezultatul operațiilor interne ale aplicației, ci și educă utilizatorul cu privire la modul în care o parolă trece de la o formă simplă, mnemonică, la una complexă și securizată, gata de utilizare.

Primul câmp, intitulat *„Story Password (Original)”*, afișează fraza brută generată de aplicație, alcătuită dintr-o succesiune de cuvinte uzuale, selectate aleator sau furnizate de utilizator. Acest format mnemonic este inspirat din metode precum Diceware, în care o serie de cuvinte comune sunt alese pentru a crea o parolă memorabilă. Utilizarea unor termeni concreți, care pot fi vizualizați mental sub forma unei povești sau a unei scene, contribuie la ușurința cu care parola poate fi reținută. Un exemplu de frază generată ar putea fi: „ocean bloom dance juice”, o combinație care evocă imagini clare și familiare.

Cel de-al doilea câmp, denumit *„Final Password (With Transformations)”*, afișează rezultatul aplicării transformărilor algoritmice asupra frazei inițiale. Aceste transformări sunt concepute pentru a spori securitatea parolei, fără a compromite complet lizibilitatea acesteia. Printre tehnicile utilizate se numără substituții de tip Leet (de exemplu, „o” devine „0”, „a” devine „@”), introducerea de majuscule în poziții

aleatorii și înlocuiri contextuale bazate pe reguli personalizabile. De asemenea, uneori pot fi eliminate spațiile sau adăugate caractere speciale, în funcție de nivelul de complexitate selectat de utilizator. Astfel, fraza „ocean bloom dance juice” poate fi transformată în „oce@nbLo0mdanc3juice”, o parolă care păstrează o parte din structura și sensul propoziției originale, dar devine semnificativ mai dificil de spart prin metode automatizate.

Prin această abordare în două etape – generare mnemonică urmată de transformare criptografică – MemoPass oferă utilizatorilor o soluție echilibrată, care combină memorabilitatea cu securitatea reală. Interfața vizuală simplifică înțelegerea procesului și încurajează adoptarea unor practici mai sigure de creare a parolelor, fără a impune formule greu de ținut minte sau șiruri arbitrare de caractere.

Ambele câmpuri includ opțiuni utile precum copiere în clipboard, ascundere/afișare și regenerare rapidă. Aceste acțiuni sunt evidențiate vizual cu iconuri intuitive, contribuind la ușurința în utilizare. În partea inferioară a interfeței, sunt prezentate detalii tehnice ale parolei, precum:

- entropia calculată (exprimată în biți);
- estimarea timpului necesar pentru spargere prin brute-force;
- scorul de tărie evaluat pe o scară de la 0 la 4;
- o bară colorată care indică vizual nivelul de siguranță (de la roșu la verde);
- o reprezentare emoji pentru fiecare cuvânt generat, care funcționează ca mnemotehnică vizuală și sprijin pentru memorare.

Interfața aplicației *MemoPass* nu este doar un mediu de interacțiune vizuală, ci un element esențial în procesul de educare, ghidare și protecție a utilizatorului în fața riscurilor asociate parolelor slabe. Prin designul său intuitiv și opțiunile flexibile de personalizare, aplicația reușește să transforme un proces tehnic complex într-o experiență accesibilă și plăcută. Feedback-ul vizual în timp real, împreună cu detaliile tehnice și suportul mnemonic, contribuie la creșterea gradului de conștientizare asupra securității parolelor și la formarea unor obiceiuri sănătoase de autentificare. Astfel, interfața își îndeplinește dublul rol: funcțional și educațional, consolidând viziunea aplicației asupra unui ecosistem digital mai sigur și mai responsabil.

IV TESTAREA ȘI VALIDAREA APLICAȚIEI

Testarea și validarea unei aplicații de securitate, cum este MemoPass, reprezintă un pas esențial în asigurarea încrederii utilizatorilor și a conformității cu bunele practici din domeniul securității cibernetice. O aplicație care generează parole fără un control riguros asupra entropiei, funcționalității sau protejării datelor poate introduce mai multe riscuri decât soluții manuale tradiționale. Prin urmare, validarea riguroasă a fiecărui modul, de la generarea criptografică la evaluarea de tărie, nu doar că confirmă corectitudinea implementării, ci și demonstrează că aplicația este pregătită să funcționeze în contexte reale, oferind protecție eficientă împotriva atacurilor comune. Testarea devine astfel o componentă critică a ciclului de viață al aplicației, în special când scopul este asigurarea unui nivel ridicat de securitate digitală.

4.1 Strategii de testare aplicate

Pentru testarea aplicației MemoPass, au fost aplicate mai multe tipuri de teste, organizate în funcție de obiectivele specifice fiecărei componente a pipeline-ului algoritmic.

Testarea funcțională a urmărit verificarea tuturor scenariilor de interacțiune ale utilizatorului. De la generarea unei parole, evaluarea scorului și până la regenerarea parolei, au fost testate toate controalele din interfață. Scopul a fost confirmarea comportamentului așteptat în funcție de combinațiile de input.

Testarea unită a vizat fiecare modul logic separat – generatorul de entropie, transformatorul Leet și evaluatorul de tărie. Aceste componente au fost testate folosind date de intrare cunoscute, pentru a obține rezultate predictibile. De exemplu, în cazul transformatorului Leet, s-a verificat dacă un caracter este corect înlocuit conform regulilor prestabilite.

Testarea de entropie a implicat o analiză statistică a parolelor generate, pentru a evalua nivelul de entropie exprimat în biți. Aceste valori au fost comparate cu lungimea și complexitatea parolelor selectate de utilizator. Rezultatele au confirmat că parolele mai complexe, generate cu transformări suplimentare și un număr mai mare de cuvinte, ating un scor de entropie peste pragul recomandat de 70 de biți.

Testarea de securitate locală a urmărit verificarea faptului că nicio informație generată nu este transmisă către server. S-au utilizat instrumente precum DevTools și Network Inspector pentru monitorizare. Totodată, au fost investigate potențiale vulnerabilități comune în contextul aplicațiilor web ce rulează exclusiv pe partea clientului.

Pentru a susține procesul de testare descris, în anexa A a raportului sunt incluse mai multe scripturi de testare JavaScript utilizate pentru verificarea funcționalității modulelor esențiale ale aplicației MemoPass. Aceste scripturi demonstrează modul de funcționare al generatorului criptografic, al transformatorului Leet, al evaluatorului de tărie, precum și proceduri de verificare a execuției locale fără trafic de rețea. Prin aceste exemple practice, se evidențiază aplicabilitatea principiilor de testare unitară și comportamentală asupra unei aplicații de securitate client-side, oferind totodată o bază reutilizabilă pentru extinderea viitoare a soluției.

4.2 Teste comparative cu alte soluții existente

Pentru a evalua relevanța și eficiența aplicației MemoPass în contextul instrumentelor actuale de generare a parolelor, au fost efectuate o serie de teste comparative cu soluții deja consacrate în industrie. Analiza s-a concentrat pe patru dimensiuni esențiale: personalizare, memorabilitate, siguranță algoritmică și experiența de utilizare.

Una dintre soluțiile comparate a fost zxcvbn, biblioteca dezvoltată de Dropbox, foarte populară pentru evaluarea tăriei parolelor. Deși oferă o analiză detaliată a structurii unei parole și returnează scoruri bazate pe predictibilitate, zxcvbn nu oferă un mecanism de generare a parolelor, ci doar o validare a celor deja introduse. Astfel, utilizatorul rămâne responsabil pentru compunerea parolei, ceea ce lasă loc erorilor umane.

O altă abordare bine cunoscută este reprezentată de sistemul Diceware, care utilizează o listă mare de cuvinte comune și le selectează aleator pentru a compune parole ușor de reținut. Deși oferă memorabilitate crescută, acest sistem nu permite personalizarea nivelului de complexitate și nici nu include transformări suplimentare (ex: Leet, simboluri), ceea ce poate afecta entropia parolei. În plus, Diceware este adesea utilizat prin linii de comandă sau interfețe austere, care nu oferă feedback vizual sau interactiv.

În ceea ce privește aplicațiile comerciale precum *LastPass Generator* și *Bitwarden Generator*, ilustrate in figura 4.2.1, acestea pun accent pe generarea aleatoare a parolelor complexe, dar fără a oferi utilizatorului o legătură semantică sau vizuală cu rezultatul. Aceste soluții sunt eficiente pentru utilizatorii avansați care folosesc manageri de parole, dar nu sunt gândite pentru uz general și nu încurajează înțelegerea sau memorarea parolei. De asemenea, fiind parte a unor platforme online, procesul de generare presupune, uneori, comunicare cu serverul, ceea ce ridică probleme de confidențialitate în anumite contexte.

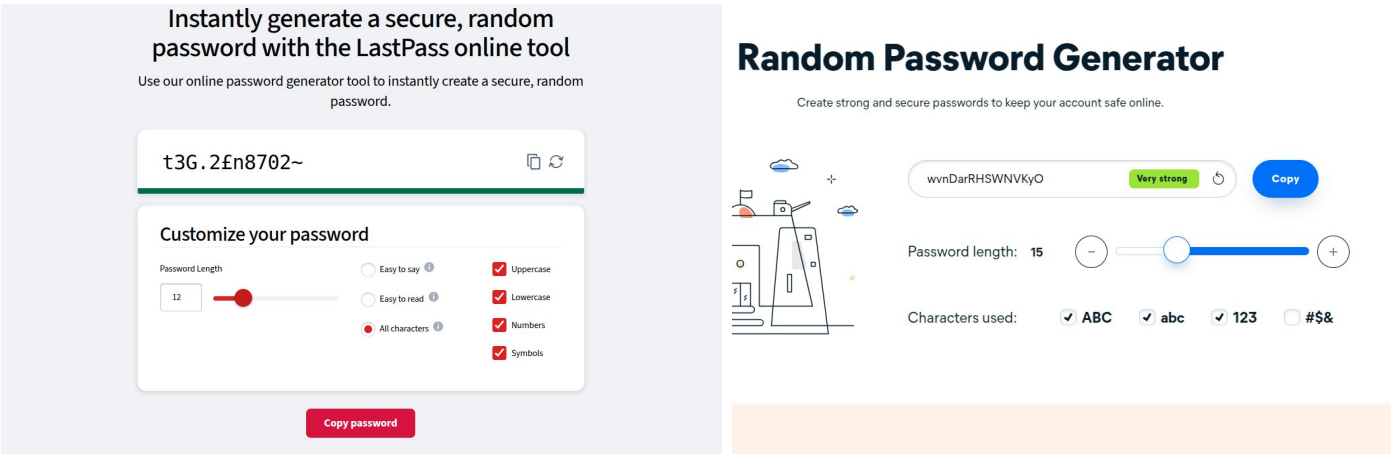

Figura 4.2.1 Analiza comparativa a instrumentelor existente pentru generarea parolelor

Comparativ cu aceste soluții, *MemoPass* se remarcă printr-un echilibru inovator între securitate și experiența utilizatorului. Aplicația oferă un grad ridicat de personalizare, utilizatorul poate alege numărul de cuvinte, nivelul de transformare Leet și poate introduce chiar propria frază mnemonică. Datorită com-

binației între generarea entropică, construcția mnemonică și transformările simbolice, parolele rezultate au atât un nivel ridicat de entropie, cât și caracter memorabil.

Un alt avantaj important îl reprezintă interfața educațională: MemoPass nu doar generează parole, ci oferă informații despre entropia obținută, scorul de tărie, estimarea timpului de spargere și vizualizarea parolei prin emoji, toate aceste elemente ajutând utilizatorul să înțeleagă de ce o parolă este sigură sau nesigură. Acest aspect transformă aplicația dintr-un simplu instrument de generare într-un mijloc de educare digitală. În plus, MemoPass funcționează 100% local, ceea ce înseamnă că nicio informație despre parole nu este transmisă prin rețea sau stocată în cloud. Acest lucru asigură un nivel superior de confidențialitate, fiind ideal pentru utilizatorii care doresc să evite riscurile asociate cu soluțiile online.

Deși aplicația MemoPass oferă o soluție inovatoare și eficientă pentru generarea de parole sigure și memorabile, există și anumite limitări care trebuie recunoscute. În forma sa actuală, aplicația nu include un sistem automat de stocare sau sincronizare a parolelor generate, ceea ce poate fi un inconvenient pentru utilizatorii care doresc gestionarea centralizată a credențialelor. De asemenea, fiind o aplicație 100% client-side, nu beneficiază de integrare directă cu browsere sau sisteme de autentificare existente. În plus, interfața actuală, deși intuitivă, poate fi îmbunătățită în viitor prin introducerea feedback auditiv sau asistență pentru utilizatori cu nevoi speciale (accesibilitate). Aceste aspecte oferă însă un cadru clar pentru extinderea funcționalității aplicației în versiunile viitoare, în funcție de cerințele utilizatorilor și evoluțiile din domeniul securității digitale.

CONCLUZII

Proiectul MemoPass a avut ca scop dezvoltarea unei aplicații web moderne care să contribuie la creșterea securității digitale prin generarea de parole sigure, memorabile și personalizabile. În contextul în care parolele continuă să fie unul dintre cele mai utilizate și expuse mecanisme de autentificare, proiectul propus răspunde unei nevoi reale și stringente – aceea de a oferi utilizatorilor instrumente accesibile care să asigure confidențialitatea, integritatea și disponibilitatea datelor (conform modelului CIA).

Obiectivele stabilite la începutul lucrării au fost atinse în mod coerent. A fost analizat cadrul teoretic privind vulnerabilitățile parolelor și s-au sintetizat cele mai bune practici din domeniul securității cibernetice. Ulterior, s-a proiectat și implementat o aplicație care urmează un model de procesare algoritmică de tip pipeline, structurat în patru pași esențiali: generare aleatoare criptografică, construcție mnemonică, transformări Leet și evaluare euristică de tărie. Fiecare dintre acești pași a fost implementat modular, testat individual și integrat într-o interfață prietenoasă și interactivă.

Selecția algoritmilor potriviți pentru generarea și evaluarea parolelor joacă un rol esențial în asigurarea unui echilibru între securitate, eficiență și experiența utilizatorului. Într-un context în care atacurile cibernetice devin tot mai sofisticate, utilizarea unor soluții generice sau simplificate, bazate pe reguli rigide sau pe validări superficiale, nu mai este suficientă. Alegerea unui algoritm inadecvat poate duce la parole ușor de intuit sau de spart, afectând nu doar securitatea individuală, ci și integritatea sistemelor informatice mai largi.

În cadrul acestui proiect, s-a analizat și comparat o serie de metode reprezentative: Diceware, zxcvbn și soluția propusă prin aplicația MemoPass. Fiecare dintre acestea are puncte forte specifice. Diceware se remarcă prin simplitatea sa conceptuală și prin capacitatea de a genera parole memorabile cu o entropie teoretic ridicată. Totuși, lipsa transformărilor suplimentare face ca parolele să fie mai previzibile în fața unui atac automatizat bazat pe modele de limbaj. LastPass Generator oferă un grad ridicat de personalizare, dar funcționează într-un mod opac, fără feedback educativ sau vizibil asupra procesului de generare. În schimb, zxcvbn excelează în evaluarea tăriei parolelor, dar nu oferă un mecanism de generare propriu.

Soluția integrată oferită de MemoPass se distinge tocmai prin faptul că reunește cele mai utile elemente ale acestor metode într-un flux coerent și extensibil. Aceasta combină generarea mnemonică inspirată din Diceware, validarea euristică de tip zxcvbn și un sistem propriu de transformări adaptabile, oferind totodată un grad ridicat de transparență și control utilizatorului. Prin această abordare modulară, MemoPass demonstrează importanța utilizării algoritmilor nu doar puternici din punct de vedere tehnic, ci și potriviți contextului în care sunt aplicați.

Unul dintre cele mai importante aspecte ale aplicației MemoPass este faptul că reușește să echilibreze cu succes trei dimensiuni esențiale: securitate, uzabilitate și educație digitală. Prin combinația dintre algoritmi avansați și o interfață intuitivă, aplicația permite utilizatorilor să genereze parole nu doar sigure

din punct de vedere tehnic, ci și ușor de reținut. Funcționalitatea de vizualizare mnemonică prin emoji, împreună cu evaluarea entropiei și estimarea timpului de spargere, transformă procesul de generare a parolei într-o experiență conștientă. Utilizatorul învață în timp real ce înseamnă o parolă puternică și de ce anumite combinații sunt considerate nesigure. În acest sens, MemoPass nu este doar un instrument tehnic, ci și un instrument de formare în cultura securității digitale.

Mai mult, faptul că toate operațiile sunt executate local, fără transmiterea datelor prin rețea, aduce un avantaj semnificativ în termeni de confidențialitate, mai ales în comparație cu alte soluții existente care implică sincronizare online sau stocare în cloud.

Aplicația a fost supusă unui set riguros de teste funcționale, unitare și de securitate. Rezultatele au confirmat stabilitatea sistemului și comportamentul corect al fiecărui modul în parte. Testele comparative cu alte generatoare de parole, precum zxcvbn, Diceware sau LastPass Generator, au evidențiat avantajele MemoPass în ceea ce privește personalizarea, interacțiunea vizuală și autonomia completă. În plus, entropia parolelor generate și scorurile de tărie au fost constante și ridicate în comparație cu parolele introduse manual de utilizatori. Evaluările de tip brute-force și estimările de timp de spargere au demonstrat că parolele generate respectă standardele moderne de securitate.

Deși aplicația oferă funcționalități solide și o interfață bine gândită, există câteva limitări care pot fi abordate în versiunile viitoare. În prezent, MemoPass nu integrează un sistem de stocare a parolelor și nu oferă funcții de export securizat sau sincronizare cu browsere sau aplicații externe. Pe termen mediu, se propune adăugarea de funcționalități precum: export al parolei în format criptat, generare coduri 2FA sau integrarea cu soluții de autentificare biometrică. În plus, arhitectura modulară a aplicației o face potrivită pentru a fi transformată într-o librărie reutilizabilă, integrabilă în alte aplicații sau platforme de securitate.

Un aspect fundamental care reiese din dezvoltarea aplicației MemoPass este importanța studierii aprofundate a algoritmilor și selectarea lor în funcție de scopul propus. În domeniul securității cibernetice, alegerea unui algoritm nepotrivit, fie pentru generarea, transformarea sau validarea datelor, poate compromite întregul sistem, indiferent de cât de bine este construit din punct de vedere vizual sau arhitectural. În cadrul acestui proiect, s-a demonstrat că algoritmi precum window.crypto.getRandomValues() pentru entropie, transformările Leet selective, evaluarea euristică și aplicarea unor șabloane mnemonice contribuie împreună la obținerea unui rezultat echilibrat între siguranță și utilizabilitate. Astfel, alegerea algoritmilor potriviți nu este doar o decizie tehnică, ci una strategică, care trebuie bazată pe înțelegerea profundă a riscurilor, avantajelor și limitărilor fiecărui mecanism în parte.

Aplicația MemoPass demonstrează cum o abordare algoritmică, modernă și orientată pe utilizator poate adresa una dintre cele mai mari provocări ale securității digitale: alegerea și utilizarea unei parole sigure. Prin utilizarea unor algoritmi riguroși, o interfață educațională și o arhitectură sigură, aplicația reușește să transforme un proces perceput adesea ca dificil sau frustrant într-o experiență controlată, eficientă și chiar intuitivă.

Lucrarea dovedește că securitatea nu trebuie să fie complicată sau inaccesibilă, din contră, poate fi gândită în mod uman-centric, astfel încât să ofere protecție fără a compromite uzabilitatea. MemoPass oferă o astfel de soluție și poate reprezenta baza pentru o serie de produse și servicii viitoare dedicate protejării identității digitale într-un mod inteligent și personalizat.

BIBLIOGRAFIE

[1] MENEZES, A. J., van OORSCHOT (1996). Manual de criptografie aplicată. CRC Press

[2] Fundația OWASP. (2023). Fișă de recomandări pentru stocarea parolelor.

[3] FERGUSON, N., SCHEINER, B., & KOHNO, T. (2010). Ingineria criptografiei: Principii de proiec tare și aplicații practice. Wiley.

[4] NIST - Institutul Național pentru Standarde și Tehnologie. (2017). Ghiduri pentru identitate digitală: Autentificare și managementul ciclului de viață

[5] CIOBANU, G. (2021). Bazele securității informației. Editura Polirom.

[6] CERT-RO (2022). Recomandări privind utilizarea parolelor sigure. Centrul Național de Răspuns la Incidente de Securitate Cibernetică.

[7] MARINESCU, M. (2019). Criptografie și securitatea informației.

ANEXA A

Algoritm pentru construcția propoziției mnemonice și reprezentarea emoji

```
     export function buildMnemonicStory(tokens: string[]): { sentence: string;
emojis: string } {
       const template = sentenceTemplates[Math.floor(Math.random() * senten-
ceTemplates.length)];

       const parts: Record<string, string> = {
         subject: '',
         verb: '',
         object: '',
         location: '',
         time: ''
       };

       const unusedTokens = [...tokens];

       for (const [category, words] of Object.entries(wordCategories)) {
         for (let i = 0; i < unusedTokens.length; i++) {
           if (words.includes(unusedTokens[i])) {
             parts[category] = unusedTokens[i];
             unusedTokens.splice(i, 1);
             break;
           }
         }
       }

       const    missingCategories    =    Object.keys(parts).filter(category
=> !parts[category] && template.includes(`[${category}]`));

      for (let i = 0; i < missingCategories.length && i < unusedTokens.length;
i++) {
         parts[missingCategories[i]] = unusedTokens[i];
         unusedTokens.splice(i, 1);
       }

       let sentence = template;
       for (const [category, word] of Object.entries(parts)) {
         if (word) {
           sentence = sentence.replace(`[${category}]`, word);
         }
       }

       sentence   =   sentence.replace(/\[\w+\]/g,   '').replace(/\s+/g,   '
').trim();

       const words = sentence.split(' ');
       const emojiSentence = words.map(word => wordToEmoji[word] || '').fil-
ter(emoji => emoji).join('');

       return {
         sentence,
         emojis: emojiSentence
       };
     }
```

ANEXA B

Algoritm de conversie Leet pentru întărirea parolelor

```
     export function applyLeetTransformations(text: string, complexity: num-
ber = 0.5): string {
       if (!text || text.trim() === '') {
         return '';
       }

       const words = text.split(' ');
       let result = '';

       for (let i = 0; i < words.length; i++) {
         const word = words[i];
         let leetWord = '';

         for (let j = 0; j < word.length; j++) {
           const char = word[j];
           const before = j > 0 ? word.substring(j - 1, j) : '';
           const after = j < word.length - 1 ? word.substring(j + 1, j +
2) : '';

           const matchingTransformations = leetTransformations.filter(
             transform => transform.from === char.toLowerCase() && trans-
form.condition(before, after)
           );

            if (matchingTransformations.length > 0 && Math.random() < complexi-
ty * 0.6) {

             leetWord += matchingTransformations[0].to;
           } else {
             leetWord += char;
           }
         }

         result += leetWord + (i < words.length - 1 ? '' : '');
       }

       if (result.length > 0) {
         const chars = result.split('');
         const randomIndex = Math.floor(Math.random() * chars.length);
         if (chars[randomIndex] && /[a-z]/.test(chars[randomIndex])) {
           chars[randomIndex] = chars[randomIndex].toUpperCase();
         }

         return chars.join('').replace(/\s+/g, '');
       }

       return result;
     }
```

ANEXA C

Algoritm de evaluare a complexității și rezistenței parolelor

```
export function evaluatePasswordStrength(password: string): {
  score: number;
  entropy: number;
  timeToCrack: string;
  strengthText: string;
} {
  const uniqueChars = new Set(password.split('')).size;
  const entropy = password.length * Math.log2(Math.max(uniqueChars, 2));

  const isCommonPassword = commonPasswords.some(common =>
    password.toLowerCase().includes(common.toLowerCase())
  );

  const meetsLengthRequirement = password.length >= 8;
  const hasLowercase = /[a-z]/.test(password);
  const hasUppercase = /[A-Z]/.test(password);

  const hasNumbers = /[0-9]/.test(password);
  const hasSpecialChars = /[^a-zA-Z0-9\s]/.test(password);

  const characterVariety =
    (hasLowercase ? 1 : 0) +
    (hasUppercase ? 1 : 0) +
    (hasNumbers ? 1 : 0) +
    (hasSpecialChars ? 1 : 0);

  const guessesPerSecond = 10000000000;
  const possibleCombinations = Math.pow(2, entropy);
  const secondsToCrack = possibleCombinations / guessesPerSecond;

  let timeToCrack: string;
  if (secondsToCrack < 60) {
    timeToCrack = `${Math.round(secondsToCrack)} seconds`;
  } else if (secondsToCrack < 3600) {
    timeToCrack = `${Math.round(secondsToCrack / 60)} minutes`;
  } else if (secondsToCrack < 86400) {
    timeToCrack = `${Math.round(secondsToCrack / 3600)} hours`;
  } else if (secondsToCrack < 31536000) {
    timeToCrack = `${Math.round(secondsToCrack / 86400)} days`;
  } else {
    timeToCrack = `${Math.round(secondsToCrack / 31536000)} years`;
  }

  let score = 0;

  if (entropy > 60) score += 1.5;
  else if (entropy > 40) score += 1;
  else if (entropy > 28) score += 0.5;

  if (password.length >= 12) score += 1;
  else if (password.length >= 8) score += 0.5;

  score += characterVariety * 0.25;
```

```
  if (isCommonPassword) score = Math.max(0, score - 2);

  score = Math.round(score * 2) / 2;

  let strengthText = "";
  if (score >= 3.5) strengthText = "Very Strong";
  else if (score >= 3) strengthText = "Strong";
  else if (score >= 2) strengthText = "Medium";
  else if (score >= 1) strengthText = "Weak";
  else strengthText = "Very Weak";

  return {
    score,
    entropy,
    timeToCrack,
    strengthText
  };
}
```

ANEXA D

Proces automatizat de generare, transformare și validare a parolelor mnemonic-securizate

```
     export function generatePassword(wordCount: number = 5, complexity: nu-
mber = 0.5): {
       originalWords: string[];
       sentence: string;
       emojiSentence: string;
       transformedPassword: string;
       strength: {
         score: number;
         entropy: number;
         timeToCrack: string;
         strengthText: string;
       };
       stages: {
         stage1: string;
         stage2: string;
         stage3: string;
         stage4: string;
       };
     } {
       let score = 0;
       let originalWords: string[];
       let story: { sentence: string; emojis: string };
       let transformedPassword: string;
       let strengthEvaluation: {
         score: number;
         entropy: number;
         timeToCrack: string;
         strengthText: string;
       };

       let stages = {
         stage1: "Starting entropy generation...",
         stage2: "Building mnemonic story...",
         stage3: "Applying leet transformations...",
         stage4: "Evaluating password strength..."
       };

       do {
         stages.stage1 = `Generated ${wordCount} random tokens`;
         originalWords = generateRandomTokens(wordCount);
        stages.stage2 = `Created story: "${originalWords.join(', ')}" → sen-
tence`;
         story = buildMnemonicStory(originalWords);

         stages.stage3 = `Applied transformations (complexity: ${Math.ro-
und(complexity * 100)}%) to "${story.sentence}"`;
         transformedPassword = applyLeetTransformations(story.sentence, com-
plexity);

         strengthEvaluation = evaluatePasswordStrength(transformedPassword);
         stages.stage4 = `Evaluated strength: ${strengthEvaluation.score}/4
(${strengthEvaluation.strengthText})`;
         score = strengthEvaluation.score;
```

```
    if (score < 3) {
      stages.stage1 = "Restarting due to low strength score...";
    }
  } while (score < 3);

  return {
    originalWords,
    sentence: story.sentence,
    emojiSentence: story.emojis,
    transformedPassword,
    strength: strengthEvaluation,
    stages
  };
}ntropy,
    timeToCrack,
    strengthText
  };
```

ANEXA E

Scripturi de testare utilizate pentru verificarea funcționalității modulelor aplicației

```
function generateRandomToken(length) {
  const    charset    =    "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVW-
XYZ0123456789";
  const values = new Uint32Array(length);
  window.crypto.getRandomValues(values);

  return Array.from(values).map(v => charset[v % charset.length]).join('');
}

function leetTransform(input, intensity = 100) {
  const map = { a: '@', e: '3', i: '1', o: '0', s: '$', t: '7' };
  return input.split('').map(char => {
    const lower = char.toLowerCase();
    if (map[lower] && Math.random() * 100 < intensity) {
      return map[lower];
    }
    return char;
  }).join('');
}

function evaluateStrength(password) {
  const lengthScore = password.length >= 12 ? 2 : 1;
  const hasDigits = /[0-9]/.test(password);
  const hasSymbols = /[^a-zA-Z0-9]/.test(password);
  const diversityScore = hasDigits + hasSymbols;

  const totalScore = lengthScore + diversityScore;
  return totalScore >= 3 ? "strong" : totalScore === 2 ? "medium" : "weak";
}
```