tjMinisterul Educaţiei, Culturii și Cercetării

Universitatea Tehnică a Moldovei

Faculatea Calculatoare, informatică și microelectronică  
Departamentul Ingineria Software și Automatică

**RAPORT**

Lucrare de Laborator nr.3

Disciplina: Analiza și Specificarea Cerințelor Software

Tema :

A efectuat:

st. gr. SI-211 Chihai A.

A verificat:

asist. univ. Tocan A.

Chişinău 2024

Cuprins:

[**Introducere** 1](#_Toc184474579)

[**1.1.** **Obiectivele laboratorului** 3](#_Toc184474580)

[**1.2. Descrierea sarcinilor de laborator** 3](#_Toc184474581)

[**2 Identificarea, Modelarea și Validarea Cerințelor Funcționale** 5](#_Toc184474582)

[**2.1. Definirea cerințelor funcționale** 5](#_Toc184474583)

[**2.2. Metodologia de modelare a cerințelor funcționale** 6](#_Toc184474584)

[**2.3. Exemple de cerințe funcționale** 7](#_Toc184474585)

[**2.4. Identificarea entităților și a relațiilor** 9](#_Toc184474586)

[**2.5. Diagrama Relație - Entitate (ERD)** 10](#_Toc184474587)

[**3 Analiza și Definirea Fluxului de Date** 12](#_Toc184474588)

[**3.1. Definiția Fluxului de Date în Sistemul de Cracking Hash** 13](#_Toc184474589)

[**3.2. Metodologia de Modelare a Fluxului de Date în Sistemul de Cracking Hash** 13](#_Toc184474590)

[**3.3. Scenarii Practice ale Fluxului de Date în Sistemul de Cracking Hash** 14](#_Toc184474591)

[**3.4. Diagrama Fluxului de Date (DFD)** 16](#_Toc184474592)

[**4 Identificarea Cerințelor Nonfuncționale** 18](#_Toc184474593)

[**4.1. Definirea cerințelor nonfuncționale** 18](#_Toc184474594)

[**5 Decompoziția Lucrărilor la Realizarea Cerințelor** 19](#_Toc184474595)

[**5.1. Definirea decompoziției lucrărilor** 20](#_Toc184474596)

[**5.2. Metodologia de decompoziție** 20](#_Toc184474597)

[**5.3. Exemple de decompoziție a sarcinilor** 22](#_Toc184474598)

[Concluzie 24](#_Toc184474599)

[Bibliografie 25](#_Toc184474600)

## **Introducere**

Într-o eră dominată de digitalizare și interconectivitate, securitatea informațională devine o componentă critică pentru protejarea datelor și a sistemelor informatice. Organizațiile, indiferent de dimensiunea lor, se confruntă cu riscuri tot mai complexe, iar tehnicile avansate de atac cibernetic pun la încercare metodele tradiționale de apărare. În acest context, analiza algoritmilor de hashing și identificarea vulnerabilităților acestora sunt esențiale pentru întărirea securității și prevenirea breșelor care pot compromite informații sensibile.

Hash-urile sunt utilizate pe scară largă pentru protecția datelor, fiind o componentă fundamentală în mecanismele de autentificare, criptare și integritate a informației. Cu toate acestea, dacă nu sunt implementate corect, acestea pot deveni puncte de atac pentru infractorii cibernetici. Tehnicile de brute-force, deși considerate consumatoare de resurse, reprezintă o metodă viabilă de evaluare a rezistenței unui algoritm de hashing, oferind perspective valoroase asupra modului în care pot fi îmbunătățite strategiile de apărare.

Lucrarea de față își propune să analizeze și să implementeze tehnici de brute-force pentru decriptarea hash-urilor, integrând aceste procese într-un sistem prietenos și eficient. Soluția va include o interfață grafică intuitivă (GUI), destinată atât specialiștilor, cât și utilizatorilor cu un nivel mediu de cunoștințe, pentru a facilita accesul la funcționalități complexe într-un mod simplu și accesibil. De asemenea, această platformă va oferi instrumente pentru optimizarea procesului de cracking, identificarea tipului de hash, evaluarea performanțelor algoritmilor și analizarea vulnerabilităților de securitate.

**Obiectivele lucrării**

1. **Analiza algoritmilor de hashing**:
   * Studierea mecanismelor utilizate de cei mai populari algoritmi de hashing (MD5, SHA-1, SHA-256) și identificarea caracteristicilor care le determină nivelul de securitate.
2. **Implementarea tehnicilor de brute-force**:
   * Dezvoltarea unor algoritmi practici de decriptare pentru diferite tipuri de hash-uri, optimizați pentru a reduce timpul de execuție și spațiul de căutare.
3. **Dezvoltarea unei interfețe grafice (GUI)**:
   * Crearea unui mediu intuitiv care să permită utilizatorilor interacțiunea cu sistemul, configurarea parametrilor de cracking și vizualizarea progresului în timp real.
4. **Testarea performanței și optimizarea**:
   * Evaluarea soluției propuse prin scenarii diverse de testare, pentru a măsura eficiența tehnicilor aplicate și pentru a identifica posibile îmbunătățiri.
5. **Identificarea vulnerabilităților**:
   * Analizarea punctelor slabe ale algoritmilor de hashing și a metodelor brute-force, pentru a oferi soluții de protecție și reducere a riscurilor.
6. **Documentarea procesului**:
   * Crearea unei documentații detaliate care să includă pașii implementării, analiza tehnologiilor utilizate, rezultatele testării și concluziile privind eficiența soluției.

## **Obiectivele laboratorului**

Obiectivele laboratorului sunt următoarele:

1. **Dezvoltarea unui sistem de cracking pentru hash-uri** – Crearea unui sistem capabil să aplice tehnici de brute-force pentru decriptarea hash-urilor utilizând algoritmi precum MD5, SHA-1 și SHA-256. Sistemul va permite simularea și analiza proceselor de cracking pentru a identifica vulnerabilitățile algoritmilor de hashing.
2. **Implementarea tehnicilor de brute-force și optimizare** – Dezvoltarea unor algoritmi eficienți care să reducă timpul de execuție și spațiul de căutare. Vor fi integrate tehnici de preprocesare a datelor și utilizarea de wordlist-uri pentru creșterea eficienței proceselor de cracking.
3. **Crearea unei interfețe grafice (GUI)** – Dezvoltarea unei interfețe prietenoase, care să faciliteze utilizarea funcționalităților sistemului. GUI-ul va permite utilizatorilor să configureze parametrii tehnicilor brute-force, să vizualizeze progresul în timp real și să analizeze rezultatele proceselor de cracking.
4. **Testarea performanței și eficienței sistemului** – Realizarea de teste pentru a evalua performanța sistemului în decriptarea hash-urilor complexe, cu diferite tipuri de algoritmi. Vor fi analizate viteza, acuratețea și eficiența tehnicilor aplicate în condiții diverse de lucru.
5. **Identificarea vulnerabilităților algoritmilor de hashing** – Analiza detaliată a punctelor slabe ale algoritmilor de hashing utilizați și evaluarea impactului atacurilor brute-force. Se vor propune soluții pentru îmbunătățirea securității prin optimizarea modului în care sunt utilizate hash-urile.
6. **Documentarea procesului de dezvoltare și testare** – Redactarea unui raport cuprinzător care să descrie pașii de implementare a sistemului, tehnologiile utilizate, strategiile de optimizare, rezultatele obținute și concluziile referitoare la performanța și aplicabilitatea soluției.

## **1.2. Descrierea sarcinilor de laborator**

În cadrul acestui laborator, vor fi realizate următoarele sarcini pentru dezvoltarea și implementarea sistemului de analiză și decriptare a hash-urilor utilizând tehnici de brute-force:

**1. Cercetarea tehnicilor de brute-force și algoritmilor de hashing**

* + Analizarea celor mai utilizate tehnici de brute-force, incluzând utilizarea de wordlist-uri, rainbow tables și alte metode de optimizare.
  + Studierea algoritmilor de hashing (MD5, SHA-1, SHA-256) pentru a înțelege structura, modul de funcționare și punctele lor slabe.
  + Identificarea tehnologiilor și bibliotecilor relevante pentru implementarea eficientă a unui sistem de decriptare.

**2. Implementarea algoritmilor de brute-force pentru decriptarea hash-urilor**

* + Dezvoltarea unui sistem capabil să proceseze hash-uri și să aplice tehnici brute-force pentru decriptarea acestora.
  + Crearea unui modul pentru determinarea tipului de hash introdus (de exemplu, identificarea automată a unui hash MD5 sau SHA-1).
  + Integrarea funcționalităților de utilizare a wordlist-urilor predefinite și opțiunea de a încărca wordlist-uri personalizate.

**3. Implementarea unei interfețe grafice (GUI)**

* + Crearea unei interfețe vizuale prietenoase care să permită utilizatorilor:
  + Să introducă hash-uri și să selecteze parametrii de cracking.
  + Să vizualizeze progresul cracking-ului în timp real.
  + Să analizeze și să exporte rezultatele decriptării.
  + Adăugarea unor funcționalități de filtrare și raportare, care să evidențieze performanțele algoritmilor utilizați și timpul necesar procesului de cracking.

**4. Testarea și optimizarea sistemului**

* + Realizarea unor teste de performanță pentru a evalua:
  + Timpul de execuție în funcție de complexitatea hash-urilor și dimensiunea wordlist-urilor.
  + Utilizarea resurselor hardware (CPU, memorie) în timpul proceselor de cracking.
  + Optimizarea algoritmilor pentru reducerea timpului de căutare și îmbunătățirea scalabilității sistemului.

**5. Securitatea și integritatea procesului de cracking**

* + Implementarea măsurilor pentru protecția datelor utilizatorului și a rezultatelor obținute.
  + Introducerea unor funcționalități care să semnalizeze potențiale erori sau utilizări incorecte ale sistemului.
  + Analiza impactului etic și de securitate al utilizării tehnicilor brute-force.

## **2 Identificarea, Modelarea și Validarea Cerințelor Funcționale**

Acest capitol se concentrează pe definirea, modelarea și validarea cerințelor funcționale pentru sistemul de analiză și decriptare a hash-urilor prin tehnici brute-force. Vom identifica cerințele esențiale, entitățile cheie și relațiile dintre acestea, pentru a asigura dezvoltarea unui sistem eficient, care să răspundă nevoilor utilizatorilor.

În această etapă, vom crea o diagramă ERD (Entity-Relationship Diagram) pentru a reprezenta structura bazei de date. Diagrama va evidenția entitățile principale, precum utilizatori, hash-uri, wordlist-uri și rezultate, și modul în care acestea interacționează pentru a susține funcționalitățile sistemului, precum:

* Identificarea tipului de hash și aplicarea tehnicilor brute-force.
* Gestionarea optimizată a proceselor de cracking.
* Vizualizarea progresului și generarea de rapoarte.

Prin modelarea cerințelor și validarea acestora, vom pune bazele unui sistem performant, bine structurat și adaptat scopurilor proiectului.

## **2.1. Definirea cerințelor funcționale**

Cerințele funcționale descriu comportamentul sistemului din perspectiva utilizatorilor și a proceselor interne, stabilind modul în care aplicația trebuie să răspundă la acțiuni, să gestioneze date și să asigure fluxurile de informații. În contextul unui sistem pentru analiza și decriptarea hash-urilor prin tehnici brute-force, cerințele funcționale esențiale includ:

1. **Procesarea și analiza hash-urilor**:
   * Sistemul trebuie să accepte hash-uri de diferite tipuri (MD5, SHA-1, SHA-256) și să determine automat algoritmul folosit.
   * Să aplice tehnici de brute-force pentru decriptarea hash-urilor, utilizând wordlist-uri predefinite sau personalizate.
2. **Monitorizarea progresului cracking-ului**:
   * Utilizatorii trebuie să poată urmări progresul proceselor de cracking în timp real, cu afișarea numărului de combinații testate și timpul estimat rămas.
3. **Gestionarea utilizatorilor**:
   * Sistemul trebuie să permită administrarea utilizatorilor, oferind funcții precum înregistrarea, autentificarea și gestionarea permisiunilor în funcție de roluri (administrator, utilizator obișnuit).
4. **Generarea și exportarea rapoartelor**:
   * Sistemul trebuie să permită crearea de rapoarte detaliate care să includă hash-urile procesate, rezultatele obținute și timpul necesar decriptării.
   * Să ofere funcționalitatea de exportare a rapoartelor în diferite formate (PDF, CSV).
5. **Vizualizarea și filtrarea datelor**:
   * Utilizatorii trebuie să poată vizualiza hash-urile procesate și rezultatele într-o interfață grafică intuitivă, cu opțiuni de filtrare pe baza criteriilor dorite (ex. tip hash, status cracking, dată).

Aceste cerințe asigură că sistemul va răspunde eficient nevoilor utilizatorilor, oferind funcționalități avansate de analiză și raportare, integrate într-un mediu ușor de utilizat.

## **2.2. Metodologia de modelare a cerințelor funcționale**

Pentru proiectul de analiză și decriptare a hash-urilor utilizând tehnici de brute-force, metodologia de modelare a cerințelor funcționale se va concentra pe transformarea nevoilor utilizatorilor într-o soluție tehnică coerentă și eficientă. Procesul va fi structurat în mai multe etape, astfel încât să acopere toate aspectele esențiale ale sistemului.

**1. Colectarea și analiza cerințelor**

* Se va începe prin discuții cu părțile interesate, inclusiv cercetători în securitate cibernetică, administratori IT și utilizatori finali. Interviurile și atelierele vor fi utilizate pentru:
  + Identificarea nevoilor cheie ale utilizatorilor.
  + Stabilirea funcționalităților dorite (ex. identificarea tipului de hash, configurarea tehnicilor brute-force, vizualizarea progresului cracking-ului).
  + Definirea scenariilor de utilizare specifice.

**2. Identificarea entităților și relațiilor**

* Entitățile principale vor fi identificate, incluzând:
  + **Hash-uri** (entități care vor fi analizate și decriptate).
  + **Wordlist-uri** (baze de date utilizate pentru cracking).
  + **Rezultate** (detalierea rezultatelor proceselor de cracking).
  + **Utilizatori** (cu diferite roluri și permisiuni).
* Relațiile dintre aceste entități vor fi definite, de exemplu:
  + „Un utilizator poate adăuga mai multe wordlist-uri.”
  + „Un hash poate genera mai multe rezultate.”
* Cardinalitatea fiecărei relații va fi stabilită și ilustrată prin diagrame.

**3. Definirea fluxurilor de date și proceselor**

* Vor fi descrise fluxurile esențiale, cum ar fi:
  + **De la introducerea hash-ului la obținerea rezultatului:** Hash-ul este procesat, tipul său este identificat, iar cracking-ul este efectuat.
  + **De la încărcarea unui wordlist la utilizarea sa:** Datele din wordlist sunt integrate în procesul de cracking.
* Vor fi detaliate procesele critice, precum analiza hash-urilor și generarea rapoartelor.

**4. Crearea diagramelor de modelare**

* **Diagramele de cazuri de utilizare (Use Case Diagrams):**
  + Vor ilustra interacțiunile dintre utilizatori (de exemplu, administratori, utilizatori finali) și sistem.
* **Diagrama fluxului de date (DFD):**
  + Va descrie modul în care datele sunt procesate și circulă între entitățile sistemului.
* **Diagrama entitate-relație (ERD):**
  + Va detalia structura bazei de date, incluzând entități și relații.
* **Diagrame de stare:**
  + Vor descrie evoluția entităților, cum ar fi stările unui hash („introdus”, „procesat”, „decriptat”)

**5. Validarea și verificarea cerințelor**

* Modelele și cerințele vor fi revizuite împreună cu părțile interesate pentru a confirma:
  + Corectitudinea cerințelor față de nevoile utilizatorilor.
  + Fezabilitatea tehnică a implementării.
* Vor fi testate prototipuri pentru a valida că sistemul răspunde cerințelor specificate și este funcțional în scenarii reale.

Această metodologie asigură o abordare structurată și comprehensivă, orientată spre dezvoltarea unui sistem robust, scalabil și capabil să răspundă cerințelor utilizatorilor și scopului propus.

## **2.3. Exemple de cerințe funcționale**

În cadrul proiectului, cerințele funcționale definesc comportamentul sistemului destinat analizării și decriptării hash-urilor prin tehnici de brute-force. Aceste cerințe sunt fundamentale pentru dezvoltarea unui sistem eficient, capabil să răspundă nevoilor utilizatorilor și să asigure performanță și accesibilitate.

**1. Procesarea hash-urilor**

Sistemul trebuie să accepte introducerea de hash-uri, identificând automat tipul acestora (ex. MD5, SHA-1, SHA-256). Hash-urile introduse vor fi analizate și procesate utilizând tehnici brute-force optimizate, cu posibilitatea utilizării wordlist-urilor predefinite sau personalizate de utilizator.

**2. Monitorizarea progresului cracking-ului**

Sistemul trebuie să afișeze în timp real progresul procesului de cracking, incluzând:

* Numărul de combinații testate.
* Timpul estimat rămas.
* Rezultatul final (dacă parola a fost identificată).

**3. Generarea alertelor și raportarea rezultatelor**

Sistemul va genera automat alerte în cazul în care procesul de cracking întâmpină probleme (ex. un algoritm de hashing necunoscut sau un set de date corupt). Aceste alerte vor fi însoțite de detalii relevante, precum:

* Tipul problemei.
* Soluții sugerate. De asemenea, utilizatorii autorizați vor putea genera rapoarte detaliate care să includă informații despre:
* Hash-urile procesate.
* Timpul și resursele utilizate.
* Rezultatele obținute.

**4. Gestionarea utilizatorilor și permisiunilor**

Sistemul va permite crearea și gestionarea conturilor utilizatorilor, cu diferite niveluri de acces:

* **Administrator**: acces complet la configurarea și monitorizarea sistemului.
* **Utilizator standard**: acces la funcțiile de analiză și raportare.
* Gestionarea permisiunilor și monitorizarea activităților utilizatorilor vor fi implementate pentru a asigura securitatea sistemului

**5. Interfață grafică (GUI) intuitivă**

Sistemul va include o interfață prietenoasă, care să permită utilizatorilor să:

* Configureze parametrii proceselor de cracking.
* Vizualizeze progresul și rezultatele în timp real.
* Acceseze rapoarte și istoricul proceselor.

**6. Notificări și alerte în timp real**

Sistemul va trimite notificări utilizatorilor prin:

* Email sau mesaje interne în aplicație.
* Alertarea rapidă a administratorilor despre evenimente critice sau rezultate importante.

## **2.4. Identificarea entităților și a relațiilor**

În cadrul proiectului destinat analizei și decriptării hash-urilor, identificarea entităților cheie și a relațiilor dintre acestea este esențială pentru modelarea bazei de date și implementarea corectă a funcționalităților sistemului. Entitățile sunt elementele principale care vor fi gestionate de sistem, iar relațiile dintre ele definesc interdependențele esențiale pentru fluxurile de lucru.

**Entitățile principale ale sistemului**

**Hash**:

Reprezintă datele care trebuie analizate și decriptate.

**Atribute**:

* ID-ul hash-ului (cheia primară).
* Valoarea hash-ului.
* Tipul hash-ului (MD5, SHA-1, SHA-256).
* Data introducerii.

**Wordlist**:

Se referă la listele de cuvinte utilizate pentru cracking-ul hash-urilor.

**Atribute**:

* ID-ul wordlist-ului (cheia primară).
* Numele wordlist-ului.
* Dimensiunea (numărul de cuvinte).
* Data încărcării.

**Rezultat Cracking**:

Conține informațiile generate după decriptarea unui hash.

**Atribute**:

* ID-ul rezultatului (cheia primară).
* ID-ul hash-ului (cheia străină).
* Statusul cracking-ului (succes/eșec).
* Parola descoperită.
* Timpul de execuție.

**Utilizator**:

Reprezintă persoanele care interacționează cu sistemul.

**Atribute**:

* ID-ul utilizatorului (cheia primară).
* Numele utilizatorului.
* Email-ul.
* Rolul (administrator/utilizator standard).

**Raport**:

Documentează activitățile de cracking și rezultatele obținute.

**Atribute**:

* ID-ul raportului (cheia primară).
* ID-ul utilizatorului creator (cheia străină).
* Data generării.
* Descrierea raportului.

**Relațiile dintre entități**

**Utilizator → Raport**:

* Un utilizator poate genera mai multe rapoarte.
* Relație de tip **1:N**.

**Hash → Rezultat Cracking**:

* Un hash poate avea un singur rezultat după procesare, însă acesta poate fi legat de mai multe încercări de cracking.
* Relație de tip **1:N**.

**Wordlist → Rezultat Cracking**:

* Un rezultat de cracking poate folosi un singur wordlist, însă un wordlist poate fi reutilizat pentru multiple hash-uri.
* Relație de tip **1:N**.

**Rezultat Cracking → Raport**:

* Un raport poate include rezultatele mai multor procese de cracking.
* Relație de tip **1:N**.

## **2.5. Diagrama Relație - Entitate (ERD)**

În cadrul procesului de modelare a cerințelor funcționale pentru sistemul de cracking hash, este esențială definirea corectă a structurii bazei de date pentru a asigura o organizare eficientă și o interconectare adecvată a datelor. Diagrama de tip ERD (Entity-Relationship Diagram) oferă o reprezentare vizuală a entităților implicate și a relațiilor dintre acestea, fiind esențială pentru înțelegerea arhitecturii sistemului.

În diagrama prezentată, am identificat următoarele entități-cheie:

* **utilizator**: gestionează informațiile despre utilizatori, inclusiv numele, emailul și rolul lor în sistem.
* **hash**: stochează date despre tipul de hash, valoarea sa și data la care a fost creat.
* **wordlist**: definește sursele utilizate pentru procesul de cracking, cu detalii despre nume, sursă și numărul de cuvinte.
* **setări\_cracking**: specifică algoritmul, limitele de timp și metodele de optimizare utilizate pentru procesul de cracking.
* **rezultat\_cracking**: înregistrează statusul fiecărui proces de cracking, timpul de execuție și relația cu utilizatorul și hash-ul procesat.

Relațiile dintre aceste entități sunt clar definite prin cardinalități și chei străine, reflectând natura interacțiunilor dintre componentele sistemului. De exemplu:

* Un utilizator poate avea multiple setări și rezultate de cracking.
* Fiecare rezultat este asociat cu un hash și un utilizator specific.
* Wordlist-urile sunt utilizate ca surse pentru cracking-ul hash-urilor.

Această structură permite o gestionare eficientă a proceselor de cracking și o trasabilitate clară a datelor generate.

**Fluxul de Date în Sistem**

Fluxul de date descrie modul în care informațiile circulă între componentele sistemului:

1. Utilizatorul configurează setările pentru procesul de cracking.
2. Datele despre hash-ul vizat sunt asociate cu wordlist-ul relevant.
3. Sistemul procesează hash-ul conform algoritmului și setărilor definite.
4. Rezultatele sunt stocate și asociate cu utilizatorul și hash-ul procesat.

Prin această reprezentare, diagrama ERD contribuie la optimizarea proceselor și asigură o bază solidă pentru implementarea funcționalităților sistemului de cracking hash.

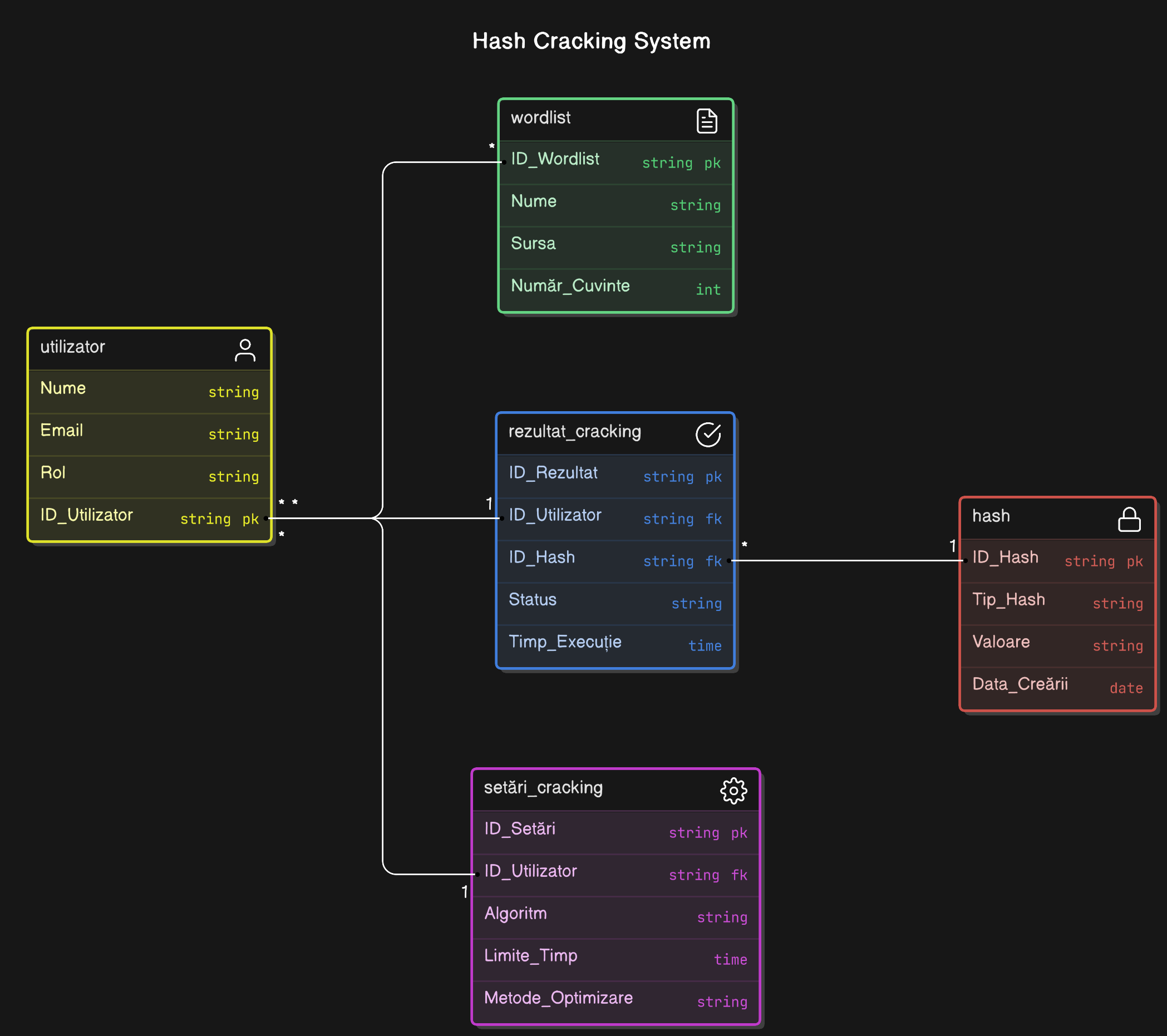


Figura 1 – Diagrama ERD

## **3 Analiza și Definirea Fluxului de Date**

Fluxul de date reprezintă modul în care informațiile circulă între diferitele componente ale sistemului, cum ar fi utilizatorii, hash-urile procesate, wordlist-urile utilizate, setările definite și rezultatele generate. O înțelegere clară a acestui flux este esențială pentru a asigura eficiența și performanța sistemului, precum și pentru a integra corect toate procesele implicate în cracking-ul hash-urilor.

În această etapă, vom analiza detaliat pașii prin care datele sunt colectate, procesate și stocate. Vom dezvolta o diagramă de flux de date (DFD) care va reprezenta vizual procesele și entitățile implicate, precum și interacțiunile și transferurile de informații între acestea. Această diagramă va evidenția modul în care utilizatorii configurează setările, hash-urile sunt identificate, wordlist-urile sunt aplicate, iar rezultatele sunt generate și analizate.

Identificarea clară a fluxului de date va contribui la optimizarea resurselor, reducerea timpului de procesare și facilitarea implementării unor mecanisme eficiente pentru cracking-ul hash-urilor. De asemenea, aceasta va sprijini integrarea funcționalităților sistemului într-un mod scalabil și intuitiv pentru utilizatori.

## **3.1. Definiția Fluxului de Date în Sistemul de Cracking Hash**

Fluxul de date reprezintă modul în care informațiile sunt transferate între componentele sistemului de cracking hash. Acesta descrie procesul prin care datele sunt colectate, procesate, stocate și transmise între entitățile implicate, precum utilizatorii, hash-urile procesate, wordlist-urile utilizate, setările definite și rezultatele generate.

Definirea fluxului de date implică stabilirea unui traseu clar al circulației informațiilor, identificând sursele (de exemplu, utilizatorii sau wordlist-urile), destinațiile (precum rezultatele cracking-ului), procesele de transformare a datelor (cum ar fi aplicarea algoritmilor brute-force) și modalitățile prin care acestea sunt stocate sau distribuite. Totodată, fluxul de date detaliază interacțiunile dintre diversele entități, asigurând un sistem coerent și eficient.

O diagramă de flux de date (DFD) este utilizată pentru a reprezenta vizual aceste interacțiuni, facilitând înțelegerea și analiza proceselor sistemului. În această diagramă, procesele sunt ilustrate prin elemente grafice precum cercuri sau dreptunghiuri, iar fluxul de date este reprezentat prin săgeți care conectează procesele și entitățile. Astfel, fluxul de date oferă o imagine clară asupra modului în care informațiile sunt gestionate și transferate în cadrul sistemului de cracking hash.

## **3.2. Metodologia de Modelare a Fluxului de Date în Sistemul de Cracking Hash**

Metodologia de modelare a fluxului de date implică definirea și reprezentarea vizuală a modului în care informațiile circulă în cadrul sistemului de cracking hash, identificând procesele cheie, sursele de date, destinațiile acestora și relațiile dintre componente. În acest context, modelarea fluxului de date se desfășoară în următorii pași esențiali:

* Identificarea proceselor cheie  
  Primul pas constă în identificarea proceselor fundamentale ale sistemului. Acestea pot include operații precum selectarea hash-urilor pentru cracking, aplicarea algoritmilor brute-force, utilizarea wordlist-urilor, optimizarea proceselor și generarea rezultatelor.
* Determinarea entităților externe și a fluxului de date  
  După definirea proceselor interne, este esențială identificarea entităților externe cu care sistemul interacționează, precum utilizatorii, wordlist-urile importate sau alte sisteme de stocare. Se analizează datele transmise și primite, stabilindu-se clar traseul informațiilor între entitățile externe și sistem.
* Crearea diagramei de flux de date (DFD)  
  Utilizând simboluri standardizate (cercuri pentru procese, săgeți pentru fluxurile de date și dreptunghiuri pentru entități externe), se construiește o diagramă care ilustrează fluxurile de informații între procesele interne și entitățile externe. Această diagramă oferă o imagine clară asupra interacțiunilor și logicii operaționale ale sistemului.
* Definirea nivelurilor de detaliu  
  Modelarea fluxului de date este realizată pe mai multe niveluri, începând cu o diagramă generală care oferă o vedere de ansamblu și continuând cu diagrame detaliate care descriu procesele și fluxurile specifice. Această abordare permite o înțelegere progresivă a arhitecturii și funcționalităților sistemului.
* Validarea fluxurilor de date  
  După realizarea diagramei de flux de date, aceasta trebuie validată pentru a asigura corectitudinea și coerența. Validarea include verificarea respectării cerințelor funcționale, identificarea eventualelor lacune și asigurarea că toate fluxurile și interacțiunile sunt clar definite și complete.

Această metodologie ajută la optimizarea și clarificarea arhitecturii sistemului, contribuind la implementarea unei soluții eficiente și bine structurate.

## **3.3. Scenarii Practice ale Fluxului de Date în Sistemul de Cracking Hash**

Exemplele de fluxuri de date din cadrul sistemului de cracking hash pot include următoarele scenarii operaționale:

1. **Încărcarea unui Wordlist**  
   Fluxul de date începe cu încărcarea unui wordlist de către utilizator în platforma de cracking hash. Wordlist-ul este stocat în baza de date corespunzătoare, urmând a fi utilizat în procesele de cracking.
   * **Entitate externă**: Utilizator
   * **Proces intern**: Încărcarea wordlist-ului
   * **Flux de date**: Fișier wordlist (nume, sursă, număr de cuvinte)
   * **Destinație**: Baza de date a wordlist-urilor
2. **Configurarea setărilor de cracking**  
   Utilizatorul configurează setările pentru procesul de cracking, specificând algoritmul, limitele de timp și metodele de optimizare. Aceste setări sunt stocate pentru a fi utilizate în procesul de cracking.

* **Entitate externă**: Utilizator
* **Proces intern**: Configurarea setărilor de cracking
* **Flux de date**: Preferințe utilizator (algoritm, limite de timp, metode de optimizare)
* **Destinație**: Baza de date a configurațiilor utilizatorului

1. **Identificarea tipului de hash**  
   Sistemul primește un hash introdus de utilizator și determină automat tipul acestuia (MD5, SHA-1, SHA-256 etc.). Această informație este stocată pentru a ghida procesul de cracking.

* **Entitate externă**: Hash-ul introdus de utilizator
* **Proces intern**: Identificarea tipului de hash
* **Flux de date**: Tip hash identificat (ex. SHA-256)
* **Destinație**: Baza de date a hash-urilor

1. **Aplicarea tehnicilor de cracking**  
   Platforma utilizează setările definite de utilizator și wordlist-ul încărcat pentru a aplica tehnicile brute-force asupra hash-ului. Rezultatul cracking-ului (dacă reușește) este stocat.

* **Entitate externă**: Wordlist și hash identificat
* **Proces intern**: Aplicarea tehnicilor de cracking
* **Flux de date**: Rezultatul cracking-ului (parola recuperată sau statusul procesului)
* **Destinație**: Baza de date a rezultatelor cracking-ului

1. **Generarea rapoartelor de activitate**  
   După finalizarea procesului de cracking, utilizatorul poate genera un raport detaliat care sumarizează setările utilizate, timpul de execuție și rezultatele obținute. Raportul poate fi stocat sau descărcat de utilizator.

* **Entitate externă**: Rezultatul cracking-ului și configurațiile utilizatorului
* **Proces intern**: Generarea raportului
* **Flux de date**: Raport detaliat (configurații, rezultate, statistici)
* **Destinație**: Raport generat

## **3.4. Diagrama Fluxului de Date (DFD)**

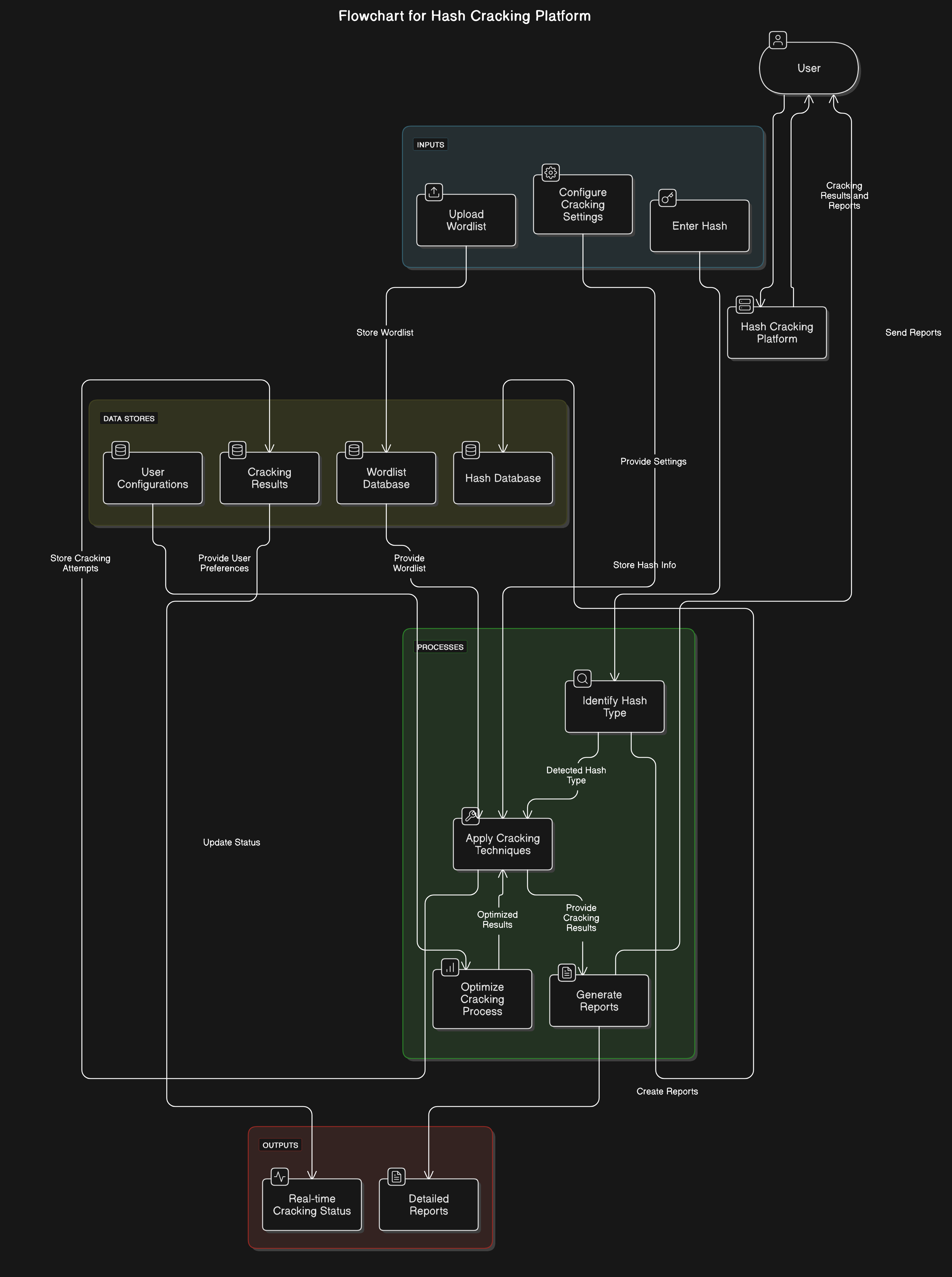


Figura 2 – Diagrama DFD

Această diagramă de flux de date (DFD) oferă o reprezentare vizuală a proceselor și interacțiunilor dintre diferitele componente ale platformei de cracking hash. În contextul proiectului, sistemul funcționează astfel:

1. **Încărcarea Wordlist-urilor**  
   Utilizatorul încarcă un wordlist care este stocat în baza de date dedicată. Acesta va fi utilizat ulterior în procesul de cracking. Wordlist-urile furnizate de utilizatori sunt gestionate pentru a asigura optimizarea și reducerea spațiului de căutare.
2. **Introducerea și Detectarea Hash-ului**  
   Utilizatorul introduce un hash pe care dorește să-l decripteze. Sistemul analizează și identifică automat tipul hash-ului (de exemplu, MD5, SHA-256) și stochează această informație în baza de date a hash-urilor pentru utilizare ulterioară.
3. **Configurarea Setărilor pentru Cracking**  
   Platforma permite utilizatorului să definească parametri esențiali pentru cracking, precum algoritmul dorit, timpul maxim permis și metodele de optimizare. Aceste setări sunt salvate pentru a ghida execuția procesului de cracking.
4. **Aplicarea Algoritmilor Brute-Force**  
   Pe baza hash-ului și a setărilor definite, platforma inițiază procesul de cracking folosind tehnici brute-force și wordlist-ul încărcat. Algoritmii sunt optimizați în funcție de setările utilizatorului pentru a reduce timpul de execuție.
5. **Generarea și Vizualizarea Rezultatelor**  
   Rezultatul cracking-ului este procesat și stocat în baza de date. Utilizatorul poate vizualiza statusul în timp real sau poate genera un raport detaliat, care sumarizează setările utilizate, timpul de execuție și rezultatele obținute.
6. **Generarea Rapoartelor**  
   Sistemul permite utilizatorului să creeze rapoarte care includ informații complete despre proces, rezultatele cracking-ului și metodele utilizate. Aceste rapoarte pot fi descărcate sau păstrate pentru audituri ulterioare.
7. **Stocarea și Accesarea Datelor**  
   Toate datele procesate, inclusiv configurațiile, rezultatele și rapoartele, sunt stocate în baze de date specifice, accesibile pentru utilizatori în orice moment.

Utilizarea unei diagrame DFD facilitează vizualizarea modului în care datele sunt procesate și transferate în cadrul sistemului, oferind o perspectivă clară asupra fluxului de informații și a interacțiunilor dintre componente. Această reprezentare grafică este esențială pentru o înțelegere detaliată a proceselor, contribuind la identificarea unor posibile puncte de optimizare și la asigurarea unei arhitecturi sistemice clare, eficiente și bine structurate.

## **4 Identificarea Cerințelor Nonfuncționale**

Cerințele nonfuncționale definesc caracteristici esențiale ale sistemului, care nu descriu comportamentele directe, dar sunt cruciale pentru asigurarea performanței, fiabilității și calității generale a acestuia. Aceste cerințe includ aspecte precum performanța, scalabilitatea, securitatea, interoperabilitatea, disponibilitatea și ușurința de utilizare. În acest capitol, vom analiza impactul acestor cerințe asupra designului și implementării sistemului, cu un accent special pe cerințele specifice de performanță și securitate în cadrul unui sistem de cracking hash.

Pentru a ilustra implementarea cerințelor nonfuncționale, vom folosi diagrame de tip Use Case. Aceste diagrame vor oferi o reprezentare clară a interacțiunilor utilizatorului cu sistemul, evidențiind scenariile care influențează direct performanța și securitatea. De exemplu, un scenariu poate detalia modul în care un utilizator configurează setările pentru cracking în timp real, iar cerințele de performanță vor specifica intervalul maxim permis pentru realizarea acestui proces.

## **4.1. Definirea cerințelor nonfuncționale**

Cerințele nonfuncționale reprezintă specificații care stabilesc modul în care sistemul își îndeplinește funcționalitățile, axându-se pe parametri de performanță, utilizare a resurselor, scalabilitate, securitate și disponibilitate. Spre deosebire de cerințele funcționale care descriu ce face sistemul, cele nonfuncționale determină cum trebuie să funcționeze pentru a asigura o experiență eficientă și stabilă pentru utilizatori.

În cadrul proiectului nostru, cerințele nonfuncționale includ următoarele exemple:

* **Timpul de răspuns**: Sistemul trebuie să proceseze fiecare hash într-un interval de timp specificat, optimizat în funcție de tipul algoritmului utilizat (ex.: MD5, SHA-256).
* **Utilizarea resurselor**: Sistemul trebuie să funcționeze eficient fără a consuma excesiv resursele hardware, optimizând utilizarea CPU și memorie.
* **Scalabilitate**: Capacitatea de a manipula simultan mai mulți utilizatori și seturi mari de wordlist-uri, fără a compromite performanța.
* **Securitate**: Asigurarea că datele utilizatorilor, inclusiv hash-urile și rezultatele cracking-ului, sunt protejate împotriva accesului neautorizat și a scurgerilor de informații.

Un exemplu concret care ilustrează cerințele nonfuncționale ar putea fi reprezentat de un **Use Case** care detaliază procesul de cracking al unui hash sub sarcină mare. În acest scenariu:

* Cerințele de **performanță** impun ca cracking-ul să fie finalizat într-un timp rezonabil, indiferent de dimensiunea wordlist-ului.
* Cerințele de **scalabilitate** cer ca sistemul să suporte mai mulți utilizatori care rulează procese de cracking simultan.
* Cerințele de **securitate** impun criptarea hash-urilor stocate și protejarea datelor de utilizator.

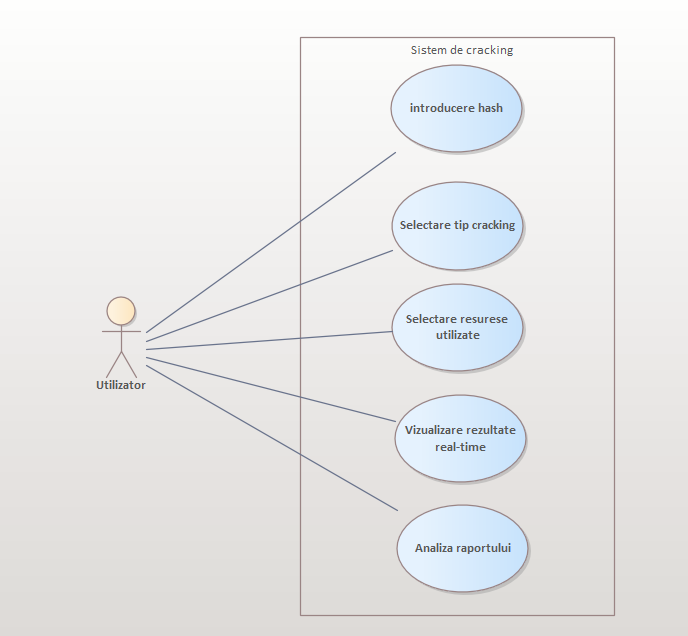


Figura 3 – Use Case diagrama

## **5 Decompoziția Lucrărilor la Realizarea Cerințelor**

Pentru realizarea cerințelor definite în cadrul proiectului, este esențială împărțirea lucrărilor în etape clare și bine structurate. Fiecare etapă contribuie la implementarea funcționalităților și cerințelor nonfuncționale într-un mod eficient și scalabil. Principalele activități includ:

1. **Analiza cerințelor** – Identificarea și clarificarea tuturor cerințelor funcționale și nonfuncționale, precum performanța, securitatea și scalabilitatea, pentru a defini specificațiile detaliate ale sistemului.
2. **Proiectarea arhitecturii** – Crearea unei arhitecturi care să sprijine implementarea eficientă a funcționalităților, asigurându-se că toate componentele sistemului sunt bine integrate.
3. **Dezvoltarea modulului funcțional** – Implementarea cazurilor de utilizare, cum ar fi introducerea hash-urilor, selectarea resurselor și generarea rapoartelor.
4. **Optimizarea și testarea performanței** – Testarea timpului de răspuns și utilizării resurselor pentru a se asigura că cerințele de performanță sunt respectate.
5. **Asigurarea securității** – Implementarea măsurilor de criptare a datelor și protejarea împotriva accesului neautorizat.
6. **Testarea finală și validarea** – Realizarea testelor funcționale și de stres pentru a verifica dacă cerințele definite sunt îndeplinite.

Această decompoziție ajută la gestionarea eficientă a proiectului și la monitorizarea progresului în fiecare etapă.

## **5.1. Definirea decompoziției lucrărilor**

Decompoziția sarcinilor reprezintă procesul de divizare a unui proiect complex în componente mai mici, ușor de gestionat, planificat și implementat. Scopul acestui proces este de a organiza activitățile într-o structură clară, facilitând monitorizarea progresului și reducerea riscurilor asociate.

În cadrul proiectului nostru de cracking hash, decompoziția sarcinilor implică divizarea cerințelor funcționale și nonfuncționale în etape distincte și atribuirea acestora unor echipe sau persoane responsabile. Această abordare permite:

* **Planificarea eficientă**: Stabilirea pașilor necesari pentru implementarea fiecărei cerințe.
* **Urmărirea progresului**: Monitorizarea realizării fiecărei etape în raport cu obiectivele stabilite.
* **Identificarea riscurilor**: Observarea rapidă a eventualelor probleme sau blocaje în procesul de implementare.

Această metodologie asigură că toate aspectele proiectului sunt bine definite și executate în mod eficient, contribuind la succesul final al implementării.

## **5.2. Metodologia de decompoziție**

Metodologia utilizată pentru dezvoltarea sistemului de cracking al hash-urilor și integrarea unui GUI este bazată pe tehnici de descompunere ierarhică, care permit împărțirea unui obiectiv complex în sarcini mai mici și gestionabile. Aceste etape sunt clar definite pentru a sprijini organizarea eficientă a lucrărilor și implementarea etapizată a funcționalităților.

**Etapele decompoziției:**

1. **Definirea obiectivelor principale**  
   Această etapă identifică activitățile majore necesare pentru finalizarea proiectului. Exemple de obiective principale includ:
   * Analiza cerințelor tehnice și funcționale.
   * Dezvoltarea componentelor pentru cracking-ul hash-urilor.
   * Crearea interfeței grafice intuitive (GUI).
   * Implementarea și testarea tehnicilor brute-force.
2. **Împărțirea obiectivelor principale în sarcini specifice**  
   Obiectivele sunt împărțite în sarcini și sub-sarcini care pot fi distribuite echipei de dezvoltare. De exemplu:
   * Pentru **analiza cerințelor tehnice**:
     + Identificarea algoritmilor de hashing (MD5, SHA-1, SHA-256).
     + Studierea vulnerabilităților și limitărilor algoritmilor.
   * Pentru **dezvoltarea funcționalităților de cracking**:
     + Implementarea generatorului de parole brute-force.
     + Dezvoltarea funcționalității de validare a tipului de hash.
     + Integrarea wordlist-urilor personalizabile.
3. **Detalierea sarcinilor**  
   Fiecare sarcină este descompusă în pași clari, cu termene limită, resurse și responsabilități definite. De exemplu:
   * În cadrul **creării GUI**:
     + Proiectarea layout-ului pentru input-ul utilizatorului.
     + Implementarea afișării rezultatelor în timp real.
     + Adăugarea funcționalității pentru exportul rapoartelor.

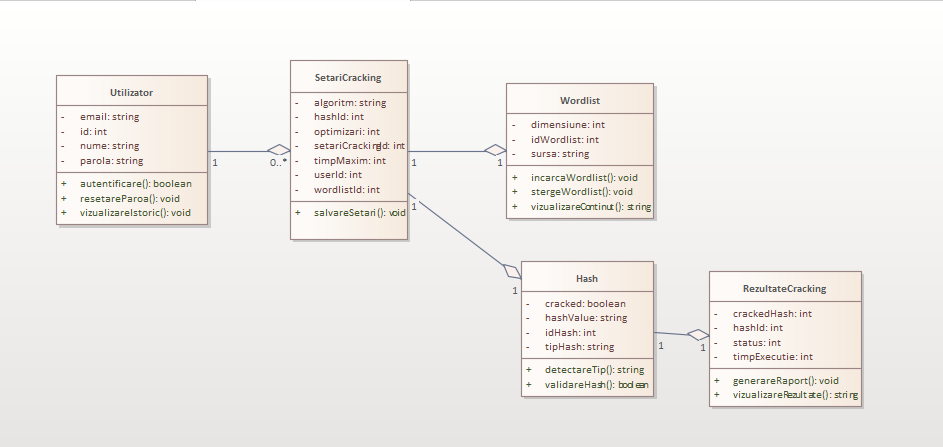


Figura 4 – Diagrama de clasa

Diagrama de clasă prezentată mai sus descrie structura internă a sistemului de cracking al hash-urilor, incluzând principalele entități, atributele acestora și relațiile dintre ele. Aceasta oferă o vedere clară asupra arhitecturii sistemului și modului în care componentele interacționează între ele.

**Interpretarea diagramei:**

* **Clase principale**:
  + Utilizator: Reprezintă utilizatorii sistemului și include funcționalități precum autentificarea și vizualizarea istoricului.
  + SetariCracking: Permite configurarea algoritmului, limitelor de timp și optimizărilor utilizate în procesul de cracking.
  + Wordlist: Gestionarea fișierelor cu liste de parole utilizate pentru atacurile brute-force.
  + Hash: Conține detalii despre hash-ul procesat, inclusiv valoarea, tipul și starea acestuia.
  + RezultatCracking: Stochează rezultatele obținute, cum ar fi parola descoperită și timpul necesar pentru cracking.
* **Relații**:
  + Utilizator este asociat cu mai multe SetariCracking.
  + Fiecare SetariCracking este conectat la un Hash și un Wordlist.
  + Fiecare Hash are asociat un singur RezultatCracking, care poate genera un raport detaliat.

**Exemplu de utilizare a diagramei**

Această diagramă este esențială pentru:

* **Planificarea implementării**: Ajută la identificarea dependențelor între componente.
* **Vizualizarea arhitecturii**: Oferă o imagine clară asupra fluxului de date și a responsabilităților fiecărei clase.
* **Testarea funcționalităților**: Asigură că toate clasele sunt corect implementate și interacționează conform specificațiilor.

## **5.3. Exemple de decompoziție a sarcinilor**

Un exemplu de decompoziție a sarcinilor din cadrul proiectului poate fi procesul de **cracking al hash-urilor**. Această activitate principală poate fi împărțită în subactivități mai mici, cum ar fi determinarea tipului de hash, selectarea tehnicii brute-force adecvate, încărcarea unui wordlist pentru cracking și procesarea propriu-zisă a hash-ului. Fiecare dintre aceste subactivități este tratată ca o unitate separată de muncă, având scopuri și livrabile bine definite.

De asemenea, **procesul de configurare a setărilor de cracking** poate fi decompozit în mai multe etape, cum ar fi selectarea algoritmului utilizat, configurarea limitelor de timp, aplicarea optimizărilor și validarea configurației. Fiecare etapă implică interacțiunea cu anumite clase, cum ar fi SetariCracking, Wordlist și Hash, iar decompoziția sarcinilor permite o alocare clară a responsabilităților și o implementare eficientă.

În mod similar, **generarea rapoartelor** poate fi descompusă în activități precum colectarea datelor din rezultatele cracking-ului, formatarea conținutului raportului, alegerea formatului de export (PDF, CSV) și vizualizarea raportului generat. Aceste activități implică clase precum RezultatCracking și Raport, iar procesul de decompoziție contribuie la identificarea interacțiunilor dintre componente și la asigurarea unui flux de lucru coerent.

Un alt exemplu este **gestionarea utilizatorilor**, care poate fi descompusă în sarcini precum autentificarea utilizatorilor, alocarea de permisiuni, vizualizarea istoricului activităților și resetarea parolelor. Fiecare activitate poate fi implementată separat, dar interacțiunile dintre ele trebuie modelate și validate pentru a garanta funcționarea corectă a sistemului.

Decompoziția sarcinilor este esențială pentru a organiza și coordona dezvoltarea sistemului, asigurând astfel o implementare coerentă, eficientă și ușor de gestionat.

# Concluzie

În cadrul acestui laborator, am abordat dezvoltarea și implementarea unui sistem de cracking al hash-urilor utilizând tehnici brute-force, cu scopul de a analiza eficiența acestor metode și de a integra o interfață grafică intuitivă (GUI). Procesul a fost împărțit în mai multe etape, fiecare având un obiectiv clar și fiind susținută de un flux de lucru bine structurat. Am început prin analiza diferitelor tipuri de hash-uri și a metodelor brute-force utilizate pentru cracking-ul acestora, urmate de implementarea algoritmilor pentru decriptarea efectivă a acestora.

Un aspect esențial al acestui laborator a fost optimizarea procesului de cracking, prin utilizarea wordlist-urilor personalizabile și a diferitelor tehnici de preprocesare, ce au redus semnificativ timpul necesar pentru descifrarea hash-urilor. Interacțiunea între componentele sistemului, cum ar fi gestionarea utilizatorilor, configurarea setărilor de cracking și generarea rapoartelor de securitate, a fost facilitată de diagramarea clară a structurilor și relațiilor dintre clase, ceea ce a contribuit la o implementare coerentă a sistemului.

În final, dezvoltarea unui GUI prietenos pentru utilizatori nespecializați a fost un pas important, permițând unui public larg să acceseze funcționalitățile sistemului fără a avea cunoștințe tehnice avansate. Testarea performanței și evaluarea eficienței soluției au confirmat că tehnicile brute-force, deși eficiente, pot reprezenta o vulnerabilitate semnificativă pentru securitatea cibernetică, iar măsurile de protecție trebuie să fie implementate corespunzător pentru a reduce riscurile. Astfel, acest laborator a oferit o oportunitate valoroasă de învățare despre securitatea informațiilor și despre implementarea unor soluții de protecție a datelor.

# Bibliografie

1. <https://sparxsystems.com/enterprise_architect_user_guide/17.0/modeling_languages/classdiagram.html>
2. <https://sparxsystems.com/resources/tutorials/ea-tutorials.html>
3. <https://sparxsystems.com/enterprise_architect_user_guide/17.0/modeling_languages/umldiagrams.html>