Universitatea Tehnică de Construcții București
Facultatea de Hidrotehnica

Proiect la PCLP III

Profesor Coordonator :
Sandu Ana Maria

Student :
Lite Karina Stefania

Cuprins

1. Introducere..... 3
1.1 Motivarea alegerii temei 3
1.2 Importanța securității digitale 3
2. Concepte și tehnologii folosite în elaborarea proiectului 4
2.1 Visual Studio Code 4
2.2 Java 5
3. Scenariu de funcționare și detalii de implementare.....6
3.1 Arhitectura proiectului 6
3.2 Crearea Interfetei grafice 7
3.3 Logica de creare a parolelor 8
3.4 Tratarea cazurilor de eroare 9
3.5 Salvarea in fisier 10
3.6 Rezultatul final 11
4. Concluzie 13
5. Bibliografie 14

1. Introducere

1.1 Motivarea alegerii temei

Am decis să creez un **generator de parole sigure** deoarece acest proiect reprezintă pentru mine echilibrul perfect între utilitatea practică și explorarea conceptelor de securitate cibernetică. Într-o eră în care atacurile informatice sunt tot mai dese, am considerat că dezvoltarea unui instrument care să protejeze datele personale este o provocare relevantă și necesară.

Deși la prima vedere generarea unui șir de caractere pare o sarcină simplă, proiectul necesită o gândire analitică și o înțelegere bună a modului în care funcționează algoritmul de alegere aleatorie. Mi-am dorit să construiesc un instrument care să nu fie doar funcțional, ci și riguros din punct de vedere tehnic, asigurându-mă că parolele create sunt rezistente în fața tentativelor de atacuri cibernetice. Prin acest proiect, am urmărit să transform o necesitate tehnică abstractă într-o soluție concretă și ușor de utilizat, care să ofere utilizatorului un sentiment real de siguranță în mediul online.

1.2 Importanța securității digitale

Securitatea digitală a devenit una dintre cele mai mari provocări ale lumii moderne, deoarece aproape toată activitatea noastră se desfășoară online. De la rețelele de socializare și adresele de e-mail, până la aplicațiile bancare, toate aceste conturi sunt protejate de o barieră principală: **parola**. Din păcate, cei mai mulți utilizatori aleg parole foarte slabe, cum ar fi numele animalului de companie, data nașterii sau combinații de cifre ușor de ghicit (de exemplu „123456”). Acestea sunt vulnerabile în fața hackerilor, care folosesc programe speciale ce pot testa mii de combinații pe secundă.

O parolă cu adevărat sigură este una care nu are o logică anume și care nu poate fi găsită într-un dicționar. Cu cât o parolă este mai lungă și conține mai multe tipuri de caractere (litere mici, litere mari, cifre și simboluri speciale), cu atât numărul de combinații posibile crește uriaș. Acest lucru face ca un program de spart parole să aibă nevoie de sute de ani pentru a o găsi, în loc de câteva secunde.

Problema apare atunci când trebuie să inventăm noi astfel de parole. Creierul uman tinde să creeze modele repetitive, iar aici intervine utilitatea proiectului de față. Un generator de parole automate elimină orice urmă de previzibilitate. Prin folosirea unor algoritmi de generare aleatorie, aplicația oferă utilizatorului certitudinea că parola sa este unică și extrem de greu de spart. În plus, funcția de salvare într-un fișier text ajută utilizatorul să nu mai fie nevoit să memoreze toate aceste coduri complexe, prevenind astfel situațiile neplăcute în care cineva își pierde accesul la un cont important pentru că a uitat o parolă dificilă. Astfel, aplicația devine un instrument de bază pentru oricine dorește să își protejeze identitatea și datele personale în mediul online .

2. Concepte și tehnologii folosite în elaborarea proiectului

În procesul de realizare a acestei aplicații, am utilizat o serie de instrumente software și limbaje de programare moderne, specifice mediului de dezvoltare Java. Alegerea acestora a fost influențată de nevoia de a crea o interfață grafică stabilă și un cod ușor de întreținut.

2.1 Visual Studio Code

Fig.1 – logo-ul aplicatiei Visual Studio Code

Visual Studio Code (prescurtat VS Code) este un editor de cod sursă dezvoltat de Microsoft, lansat oficial în anul 2015. Este gratuit, open-source și cross-platform, ceea ce înseamnă că poate fi utilizat pe Windows, macOS și Linux. VS Code s-a impus rapid ca unul dintre cele mai populare editoare de cod din lume datorită simplității sale, performanței excelente și ecosistemului extins de extensii.

Editorul oferă suport încorporat pentru un număr mare de limbaje de programare, cum ar fi JavaScript, Python, C++, Java, HTML/CSS și altele. În plus, prin extensii din marketplace, utilizatorii pot adăuga suport pentru tehnologii și limbaje suplimentare, pot integra debuggere, terminale, controlul versiunilor (Git) sau chiar medii de dezvoltare virtuale.

Visual Studio Code este folosit frecvent în mediul universitar, mai ales în cadrul facultăților cu profil informatic. Studenții îl utilizează pentru scrierea și testarea codului în diverse limbaje de programare, fiind preferat pentru interfața intuitivă, instalarea rapidă și compatibilitatea cu platforme educaționale. De asemenea, multe cursuri și laboratoare recomandă sau solicită folosirea acestui editor, datorită integrării sale eficiente cu GitHub, terminalul integrat și suportul pentru debugging, facilitând astfel procesul de învățare și dezvoltare.

2.2 Java

Fig.2 – logo-ul limbajului de programare Java

Am ales limbajul Java pentru acest proiect deoarece oferă o construcție logică și eficientă, fapt care a facilitat separarea clară între codul responsabil de generarea parolelor și elementele ferestrei grafice. Această organizare a codului pe clase și obiecte face ca programul să fie mult mai ușor de înțeles și de modificat pe viitor. Un alt beneficiu major al utilizării Java este sistemul său de gestionare automată a resurselor, care asigură o funcționare stabilă și previne erorile care ar putea bloca aplicația în timpul execuției. Prin utilizarea versiunii JDK 17, am beneficiat de un mediu de lucru modern și sigur, capabil să proceseze rapid cerințele utilizatorului.

Pentru a transforma acest utilitar dintr-un simplu script într-o aplicație desktop completă, am utilizat bibliotecile grafice Swing și AWT. Java Swing ne-a permis să construim toate elementele vizuale pe care le vede utilizatorul, de la butoanele de comandă până la bifele pentru opțiuni și listele derulante. În completare, biblioteca AWT a fost esențială pentru a da viață acestor elemente,

permițând programului să reacționeze la click-urile mouse-ului și să schimbe culorile sau fonturile textului. Această interacțiune directă între utilizator și interfață este gestionată prin „ascultători de evenimente”, care fac legătura între butoanele vizibile și logica matematică din spatele lor.

Un aspect tehnic foarte important în designul ferestrei a fost utilizarea sistemului GridBagLayout. Acesta funcționează ca o rețea invizibilă de celule care ne permite să așezăm componentele într-o ordine precisă, evitând astfel suprapunerea butoanelor sau apariția unor spații goale inestetice. Prin configurarea marginilor și a modului de aliniere, am obținut o fereastră compactă și ergonomică, unde toate comenzile sunt grupate logic. Această îmbinare între stabilitatea limbajului Java și flexibilitatea componentelor grafice rezultă într-o aplicație profesională, gata să fie utilizată pe orice sistem de operare.

3. Scenariu de funcționare și detalii de implementare

3.1 Arhitectura proiectului

Aplicația a fost concepută urmând principiul separării logicii de interfață, fiind structurată în două clase Java distincte:

Clasa PasswordGenerator - reprezintă baza aplicației. Aici este stocată toată logica algoritmică, fără elemente vizuale. Rolul ei este de a construi parolele pe baza seturilor de caractere definite.

Clasa PasswordApp - gestionează interfața grafică . Aceasta apelează metodele din prima clasă și se ocupă de tot ce înseamnă interacțiunea cu utilizatorul: butoane, ferestre de dialog și afișarea rezultatelor.

3.2 Crearea Interfeței Grafice (GUI Design)

Interfața a fost creată folosind biblioteca javax.swing. Am ales un design minimalist, dar funcțional, punând accent pe ușurința în utilizare. Am utilizat **GridBagLayout** pentru a organiza componentele într-o grilă invizibilă. Acest lucru mi-a permis să controlez exact distanța dintre rânduri (folosind Insets) și să evit aspectul de elemente împrăștiate. Pentru a oferi un aspect modern, am setat culori personalizate butoanelor (nuanțe de gri deschis) și am utilizat fontul „Segoe UI" de dimensiune 14-18, asigurând o lizibilitate crescută. Am inclus JCheckBox-uri pentru opțiunile parolei și un JComboBox pentru a permite utilizatorului să vizualizeze și să aleagă dintre cele 3 variante generate.

```
public class PasswordApp extends JFrame {
    private JTextField txtLungime, txtAplicatie;
    private JCheckBox lMari, lMici, Cifre, Simboluri;
    private JComboBox<String> comboParole;
    private PasswordGenerator generator = new PasswordGenerator();

    private final String HINT_LUNGIME = "Ex: 10";
    private final String HINT_APP = "Introdu numele aplicației : ";
    private final Font fontPrincipal = new Font(name: "Segoe UI", Font.BOLD, size: 14);

    public PasswordApp() {
        setTitle(title: "PasswordGenerator");
        setSize(width: 500, height: 600);
        setDefaultCloseOperation(EXIT_ON_CLOSE);
        setLocationRelativeTo(c: null);

        JPanel panelPrincipal = new JPanel(new GridBagLayout());
        GridBagConstraints gbc = new GridBagConstraints();
        gbc.insets = new Insets(top: 8, left: 8, bottom: 8, right: 8);
        gbc.fill = GridBagConstraints.HORIZONTAL;
        gbc.gridx = 0;
```

Fig.3 - Inițializarea interfeței grafice și configurarea ferestrei principale

```
JLabel lblL = new JLabel(text: "Lungime parolă:");
lblL.setFont(fontPrincipal);
gbc.gridy = 0; panelPrincipal.add(lblL, gbc);
```

Fig.4 - Implementarea etichetelor de text și poziționarea pe grilă

```
lMari = new JCheckBox(text: "Litere Mari (A-Z)");
lMici = new JCheckBox(text: "Litere Mici (a-z)", selected: true);
Cifre = new JCheckBox(text: "Cifre (0-9)");
Simboluri = new JCheckBox(text: "Simboluri (!@#)");
```

Fig.5 - Configurarea opțiunilor prin JCheckBox

3.3 Logica de creare a parolelor

Procesul de generare este unul dinamic. În momentul în care utilizatorul apasă butonul, programul parcurge următorii pași tehnici:

Se creează un obiect de tip StringBuilder în care se colectează toate caracterele permise (litere, cifre, simboluri), în funcție de bifele selectate de utilizator în interfață. Folosind clasa Random, programul extrage un caracter la întâmplare din acest set construit anterior și îl adaugă la parola finală. Acest pas de selecție se repetă în interiorul unei instrucțiuni de tip for până când parola ajunge la lungimea cerută de utilizator.

PasswordGenerator.java

```
1  import java.util.Random;
2
3  public class PasswordGenerator {
4      private static  String MICI = "abcdefghijklmnopqrstuvwxyz";
5      private static  String MARI = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
6      private static  String CIFRE = "0123456789";
7      private static  String SIMBOLURI = "!@#$%^&*()-_=+[]{}|<>?";
8
```

Fig.6 - Definirea seturilor de caractere

```java
public String genereaza(int lungime, boolean cuMari, boolean cuMici, boolean cuCifre, boolean cuSimboluri) {
    StringBuilder setCaractere = new StringBuilder();
    if (cuMici) setCaractere.append(MICI);
    if (cuMari) setCaractere.append(MARI);
    if (cuCifre) setCaractere.append(CIFRE);
    if (cuSimboluri) setCaractere.append(SIMBOLURI);

    if (setCaractere.length() == 0) return "Selectează opțiuni";

    Random random = new Random();
    StringBuilder parola = new StringBuilder();
    for (int i = 0; i < lungime; i++) {
        int index = random.nextInt(setCaractere.length());
        parola.append(setCaractere.charAt(index));
    }
    return parola.toString();
}
}
```

Fig.7 - Logica algoritmului de generare

3.4 Tratarea cazurilor de eroare

Programul a fost conceput să prevină blocarea sau închiderea neașteptată în cazul în care utilizatorul introduce date greșite. Am implementat următoarele mecanisme de siguranță:

Controlul lungimii parolei: În situația în care utilizatorul introduce litere în locul cifrelor sau lasă câmpul gol, am utilizat un bloc de tip try-catch. Acesta interceptează eroarea de format (NumberFormatException) și afișează un mesaj de avertizare prin JOptionPane, ghidând utilizatorul să introducă un număr valid.

Verificarea opțiunilor de selecție: Programul verifică dacă cel puțin o căsuță de caractere este bifată înainte de a începe generarea. Dacă nu este selectată nicio opțiune, aplicația detectează că setul de caractere este gol și avertizează utilizatorul că trebuie să aleagă tipul de caractere dorit.

Validarea datelor înainte de salvare: Butonul de salvare include o verificare suplimentară pentru a se asigura că există o parolă generată în lista derulantă și că numele aplicației a fost completat. Astfel, se evită scrierea unor informații incomplete în fișierul extern.

```java
btnGenerare.addActionListener(e -> {
    String textLungime = txtLungime.getText();

    if (textLungime.equals(HINT_LUNGIME) || textLungime.isEmpty()) {
        JOptionPane.showMessageDialog(this, message: "Te rog introdu un număr pentru lungime!");
        return;
    }

    try {
        int lungime = Integer.parseInt(textLungime);


        if (!lMari.isSelected() && !lMici.isSelected() && !Cifre.isSelected() && !Simboluri.isSelected()) {
            JOptionPane.showMessageDialog(this, message: "Selectează cel puțin o opțiune de caractere!");
            return;
```

Fig.8 -Validarea datelor și gestionarea erorilor în interfață

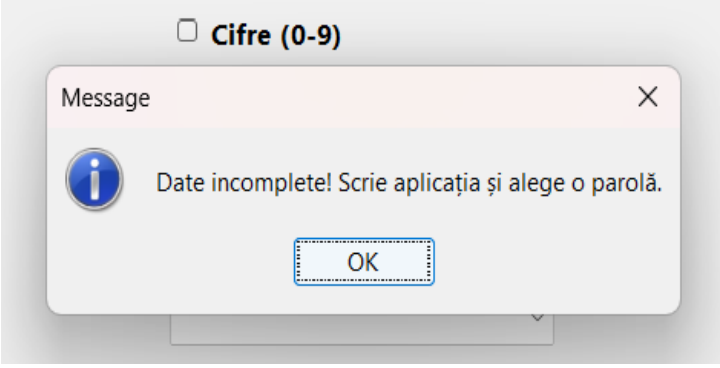

Fig.9 - Interfața de avertizare pentru date incomplete

3.5 Salvarea în fișier

Salvarea se face automat în fișierul parole.txt. Am ales să folosesc BufferedWriter împreună cu FileWriter în modul „append”. Aceasta este o funcționalitate critică, deoarece permite utilizatorului să își construiască o listă lungă de parole în timp, fără a pierde datele salvate anterior.

```
    }
    try (BufferedWriter bw = new BufferedWriter(new FileWriter(fileName: "parole.txt", append: true))) {
        bw.write("Aplicatie: " + app + " | Parola: " + parola);
        bw.newLine();
        JOptionPane.showMessageDialog(this, message: "Salvat cu succes în parole.txt!");
    } catch (IOException ex) {
        JOptionPane.showMessageDialog(this, message: "Eroare la scriere!");
    }
});
```

Fig.10 - Salvarea datelor în fișier extern

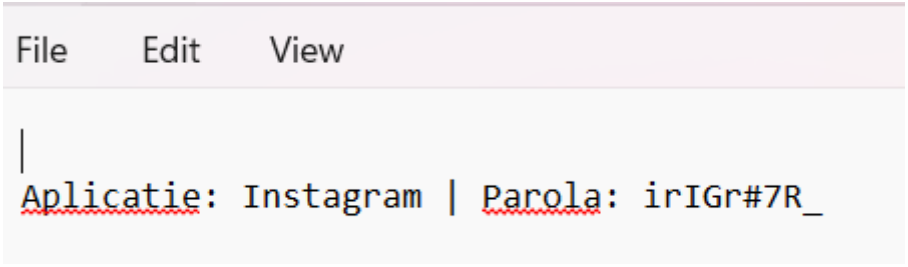

Fig.11 – Datele salvate in fisierul parole.txt

3.6 Rezultatul final

Rezultatul final al acestui proiect este o aplicație desktop care îmbină funcționalitatea cu un design intuitiv. Interfața grafică a fost proiectată să fie cât mai simplă, permițând utilizatorului să configureze rapid lungimea și tipul caracterelor pentru noile parole. Odată apăsat butonul de generare, aplicația oferă trei variante de parole într-o listă derulantă, oferind flexibilitate în alegerea celei mai bune opțiuni. Procesul se încheie prin salvarea parolei selectate, împreună cu numele aplicației corespunzătoare, într-un fișier extern de tip text. Succesul operațiunii este confirmat printr-un mesaj de alertă, iar datele pot fi verificate imediat în fișierul parole.txt, unde sunt stocate într-un format organizat și ușor de citit.

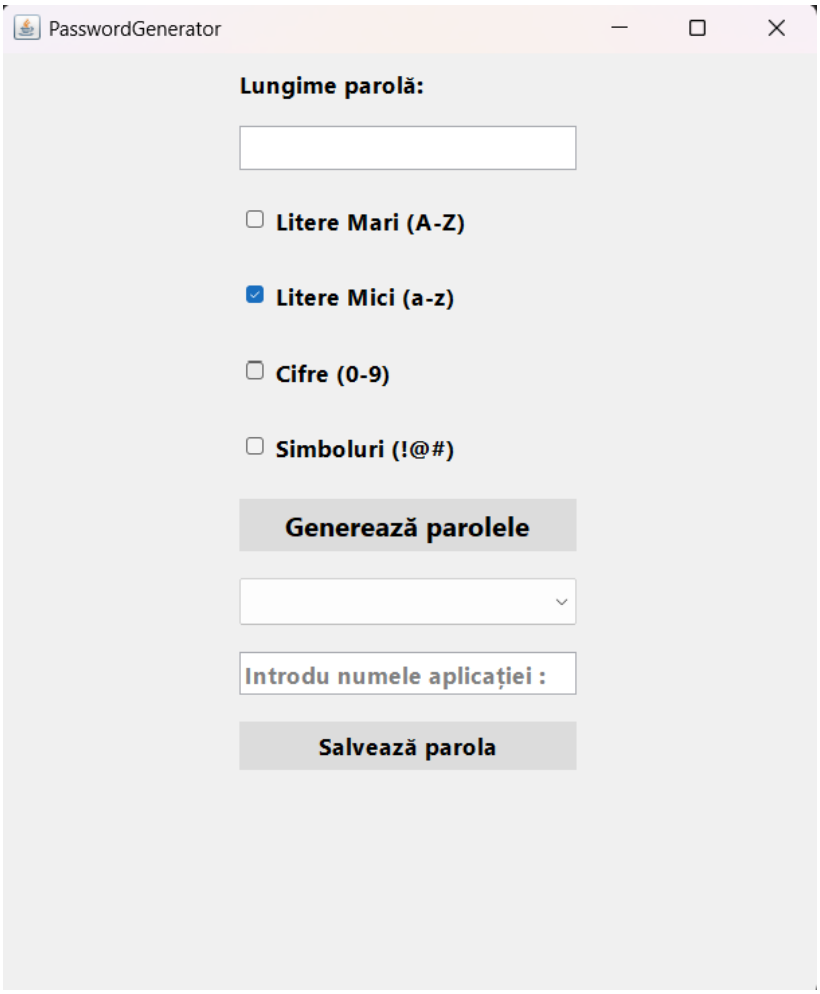

Fig.12 - Aspectul general al interfeței grafice

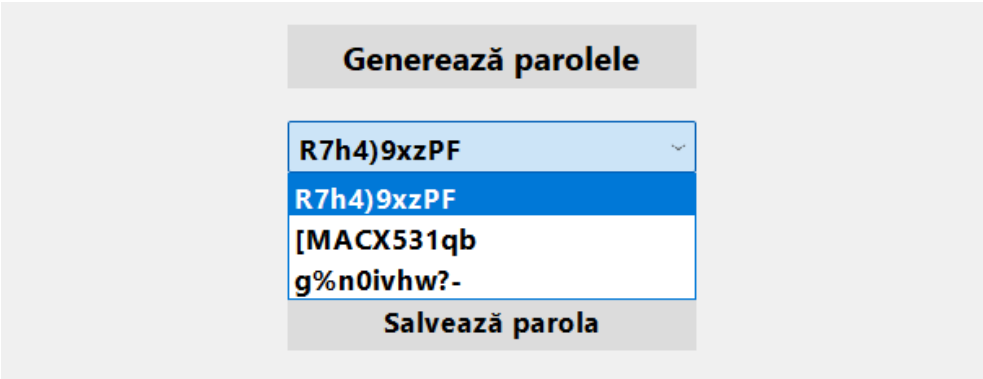

Fig.13 - Selectarea unei parole din cele trei variante generate

4. Concluzie

Realizarea proiectului „Generator de Parole Sigure” a reprezentat o experiență practică foarte utilă, care mi-a permis să aplic cunoștințele de programare dobândite pe parcursul acestui semestru la PCLP . Obiectivul principal, acela de a crea un instrument capabil să genereze și să stocheze parole complexe, a fost îndeplinit cu succes, rezultând o aplicație stabilă și ușor de utilizat de către orice persoană.

Din punct de vedere tehnic, acest proiect mi-a oferit ocazia de a aprofunda utilizarea limbajului Java într-un context real, dincolo de simplele exerciții de algoritmica. Am învățat cum să construiesc o interfață grafică funcțională folosind biblioteca Swing și cum să gestionez evenimentele declanșate de utilizator prin intermediul butoanelor. Un aspect esențial a fost gestionarea corectă a erorilor . Am învățat că o aplicație bună trebuie să fie pregătită pentru orice tip de input greșit din partea utilizatorului și să ofere mesaje de ghidare clare, în loc să se blocheze.

De asemenea, faptul că aplicația poate salva informațiile într-un document text pe hard disk transformă un simplu algoritm de calcul într-un utilitar practic, care poate fi folosit zi de zi pentru organizarea securității conturilor online. Separarea programului în două clase distincte (una pentru logică și una pentru interfață) m-a ajutat să înțeleg mai bine conceptul de modularitate, făcând codul mult mai ușor de citit și de corectat.

Ca perspective de viitor, aplicația poate fi dezvoltată în mai multe direcții. O îmbunătățire importantă ar fi implementarea unui sistem de criptare pentru fișierul în care sunt salvate parolele, astfel încât acestea să nu poată fi citite de oricine deschide documentul text. De asemenea, s-ar putea adăuga o funcție de copiere automată a parolei în clipboard sau posibilitatea de a edita și șterge parolele direct din interfața grafică. În concluzie, acest proiect constituie o bază solidă pentru dezvoltarea unor aplicații de securitate mai avansate și a reprezentat un pas important în formarea mea ca programator.

5. Bibliografie

https://docs.oracle.com/en/java/javase/17/

https://docs.oracle.com/javase/8/docs/api/java/lang/StringBuilder.html

https://docs.oracle.com/javase/8/docs/api/java/util/Random.html

https://docs.oracle.com/javase/tutorial/uiswing/

https://www.w3schools.com/java/java_files_create.asp

https://www.tutorialspoint.com/java/index.htm

Suport de curs PCLP III (Teams , GitHub)