5/2/2021

Fereshteh Azadi

Chalmers

HashMaker

LIA2

# Contents

[1 Contents 1](#_Toc70850198)

[2 Bakgrund 2](#_Toc70850199)

[3 Uppgift 2](#_Toc70850200)

[4 Programmet 2](#_Toc70850201)

[5 Databas 2](#_Toc70850202)

[5.1 PasswordHashes 2](#_Toc70850203)

[5.2 ProcessingInfo 2](#_Toc70850204)

[6 Klasser 2](#_Toc70850205)

[**6.1** PasswordGenerator 3](#_Toc70850206)

[6.2 ThreadManager: 3](#_Toc70850207)

[6.3 HashGenerator: 3](#_Toc70850208)

[6.4 DatabaseManager 3](#_Toc70850209)

[6.5 PasswordHashData 3](#_Toc70850210)

[6.6 Database 3](#_Toc70850211)

[6.6.1 HashingContext 3](#_Toc70850212)

[6.6.2 PasswordHash 3](#_Toc70850213)

[6.6.3 ProcessingInfo 3](#_Toc70850214)

[7 Klassdiagram 4](#_Toc70850215)

[8 Databasdiagram 6](#_Toc70850216)

# Bakgrund

Normalt sparar man inte lösenorden som användarna använder för att logga in. Det man istället sparar är en hash. Det finns flera olika hash algoritmer och varje algoritm översätter ett lösenord till sin kombination av tecken.

Om man ska ta reda på vilket lösenrod en hash motsvarar så kan man inte använda en hash algoritm då den enbart kan översätta från lösenrod till hash. Tillbaka går inte. Detta gör att man får använda en brute force metod, alltså att man får beräkna olika påhittade lösnenord med en hash algoritm tills man hittat rätt lösenord. Detta tar lång tid och istället använder man en databas där man söker på en hash för att hitta rätt lösenord.

# Uppgift

Min uppgift blir att skapa ett program som skapar hash:ar och lägger dessa i en databas.

1.       Trådning skall tillämpas för att kunna räkna fram flera olika hashar parallellt enligt formatat antal kärnor i CPU:n (kallat n) minus en, alltså n-1 trådar ska köras parallet. (Minus en för att operativsystemet skall ha en kärna ledig för att hantera övriga program i windows, mus, tangentbord, mm.

2.       Databasen skall ha formatet

a.       Tabell medLösenord \* Hash enligt algoritm 1 \* Hash enligt algoritm 2 \* etc.  
b.       Tabell som håller reda på hur långt programmet hunnit i beräkningen av hashar (den börjar med att beräkna hash för alla möjliga tecken för ett lösenord av längden 1, sedan längden 2, etc)

# Programmet

När programmet startar hämtas första lösenordet från databasen.   
Sen skapas trådar för att generara hashars. Dessa trådar beräknar 100(går att konfigurera) hashar åt gången. När alla trådarna är klara med sina beräkningar så sparas resultatet från dessa i en tabell (PasswordHashes) i databasen. Samtidigt sparas även hur långt programmet har kommit i beräkningarna (ProcessingInfo).  
Detta upprepas tills programmet avbryts genom att användaren trycker på valfi tangent.

# Databas

Databasen är skapad enligt ”Code First” med hjälp av EntityFrameworkCore.

Den består av två tabeller:

## PasswordHashes

I denna sparas information om alla genererade lösenord och tre typer av hashar (SHA256, SHA385, SHA512)

## ProcessingInfo

Här sparas information om hur långt programmet kommit i beräkningarna så att det kan fortsätta med rätt lösenord när det startas nästa gång.

# Klasser

Programmet består av följande klasser

## PasswordGenerator

Klassen hanterar skapandet av lösenord som programmet behöver för att skapa hashar.  
Det består av en konstruktor som initierar klassen med det första lössenordet som programmet begär.  
*GetCurrentPassword()* anropas för att hämta nuvarande lösenord .  
*NextPassword()* anropas för att generera nästa lösenord.

## ThreadManager:

Huvudklassen som håller ihop allt som programmet ska göra.  
Skapar och kör trådar för att skapa hashar. Sparar sedan dessa till databasen.

## HashGenerator:

class HashGenerator räknar ut hashar för ett lösenord. Skapar hashar av typen sha256,sha384,sha512

## DatabaseManager

Den klassen hanterar att spara lösenorden och hasharna till databasen.Sparar också nästa lösenord som skall skapas när programmet startas.

*init(String defaultPassword)* är tänkt att anropas när man startar programmet.  
Hämta nästa lösenord från databasen. Fins det inget sparat lösenord så kommer defaultPassword att användas.

*save(List<PasswordHashData> passwordHashDataList, String nexPassword*) sparar lista av PasswordHashData och nextpassword till databasen

## PasswordHashData

klass för information om password och dess olika hashar

## Database

Innehåller tre klasser :

### HashingContext

Definitionen av databasen (connectionString och tabeller)

### PasswordHash

klass för databastabellen PasswordHash

### ProcessingInfo

klass för databastabellen ProcessingInfo

# Graphical user interface, text, application Description automatically generatedKlassdiagram

Graphical user interface, text, application

Description automatically generated

# Databasdiagram