Documentazione Password Security Checker

1 Introduzione 3

1.1 Informazioni sul progetto 3

1.2 Abstract 3

1.3 Scopo 3

2 Analisi 4

2.1 Analisi del dominio 4

2.2 Analisi e specifica dei requisiti 4

2.3 Use case 6

2.4 Pianificazione 6

2.5 Analisi dei mezzi 6

2.5.1 Software 6

2.5.2 Hardware 6

3 Progettazione 6

3.1 Design dell’architettura del sistema 7

3.2 Design dei dati e database 7

3.3 Design delle interfacce 7

3.4 Design procedurale 7

4 Implementazione 8

5 Test 8

5.1 Protocollo di test 8

5.2 Risultati test 10

5.3 Mancanze/limitazioni conosciute 10

6 Consuntivo 10

7 Conclusioni 10

7.1 Sviluppi futuri 10

7.2 Considerazioni personali 10

8 Bibliografia 10

8.1 Bibliografia per articoli di riviste: 10

8.2 Bibliografia per libri 10

8.3 Sitografia 11

9 Allegati 11

# Introduzione

## Informazioni sul progetto

Allievi coinvolti: Alessandro Castelli

Docenti coinvolti: Luca Muggiasca, Geo Petrini

SAMT Informatica Progetti

Data inizio: 09.09.2021

Data consegna: 23.12.2021

## Abstract

* **Background/Situazione iniziale**
  + In questo progetto è stato creato un programma che permette di forzare una password che contiene qualsiasi tipo di carattere possibile nella testiera Francese-Svizzera. Per forzare la password per prima cosa crea diverse combinazioni con gli argomenti passati da linea di comando (nome, cognome, data di nascita e un argomento facoltativo), se sono stati passati tutti gli argomenti in modo corretto può generare quasi 1’000'000 combinazioni. Se con gli argomenti la password non viene forzata allora passa in rassegna 100'000 password in un file (le 100'000 password più utilizzate) e se anche così non viene trovata il programma esegue un brute force che permette di creare tutte le combinazioni possibili con i caratteri presenti sulla tastiera Francese-Svizzera.
* **Approccio/Metodi**
  + Essendo non necessario avere una parte grafica in questo progetto ho utilizzato java come linguaggio di programmazione passando i diversi parametri necessari da linea di comando.
* **Risultati**
  + Il programma è funzionante.

## Scopo

Lo scopo del progetto è di creare un programma che permette di forzare una qualsiasi password che contiene caratteri presenti sulla tastiera Francese-Svizzera. Cercando di creare combinazioni utilizzando informazione dell’utente (nome, cognome, data di nascita e un argomento facoltativo), passando in rassegna un file con le password più utilizzate ed eseguendo un brute force

## Analisi del dominio

Questo prodotto è utilizzabile da qualsiasi sistema operativo con installato java, viene salvato in un file jar quindi per essere utilizzato basta saper digitare il comando da utilizzare nella linea di comando e passare gli argomenti necessari al funzionamento del programma (password, nome, cognome, data di nascita).

## Analisi e specifica dei requisiti

|  |  |  |
| --- | --- | --- |
| Id | Requisito | Priorità |
| 1 | File csv con le password più utilizzate | Alta |
| 2 | Passaggio di parametri dell’utente (password, nome, cognome, data di nascita, ecc) | Alta |
| 3 | Calcolo del tempo impiegato | Media |
| 4 | Generazione casuale di password con brute force | Alta |
| 5 | Indicazione del numero di tentativi svolti | Media |
| 6 | Deve poter prendere in considerazione ogni tasto digitabile dalla tastiera “Svizzera” | Alta |
| 7 | Passaggio di password come primo parametro | Alta |

**Spiegazione elementi tabella dei requisiti:**

**ID**: identificativo univoco del requisito

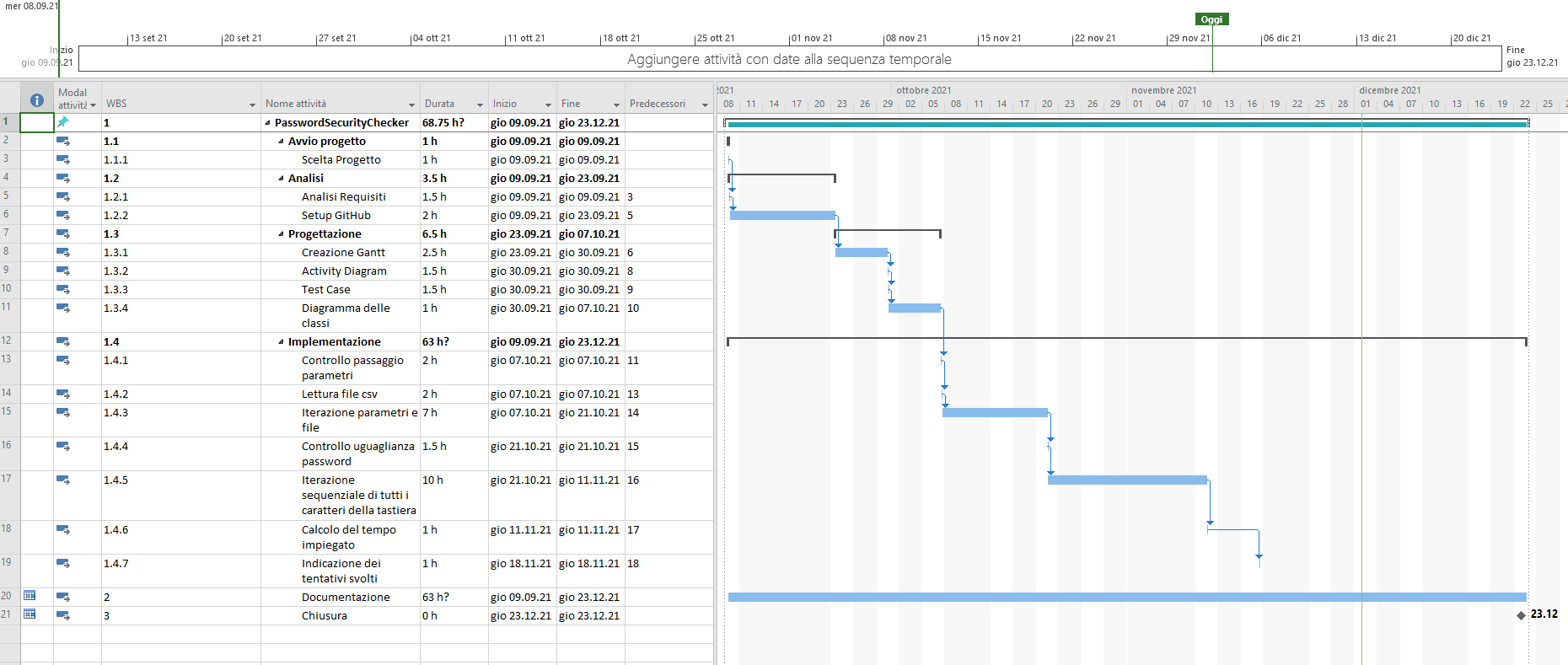
**Requisito**: breve descrizione del requisito

**Priorità**: indica l’importanza di un requisito nell’insieme del progetto, definita assieme al committente. I livelli di priorità sono: bassa, media e alta.

## Pianificazione

Pianificazione utilizzata: WaterFall

Ho rappresentato la pianificazione mediante un diagramma di Gantt:



## Analisi dei mezzi

### Software

Apache Netbeans IDE 12.4

Java 16.0.2

Librerie importate:

* java.io.BufferedReader
* java.io.IOException
* java.io.InputStream
* java.io.InputStreamReader
* java.util.ArrayList
* java.util.Arrays
* java.util.List

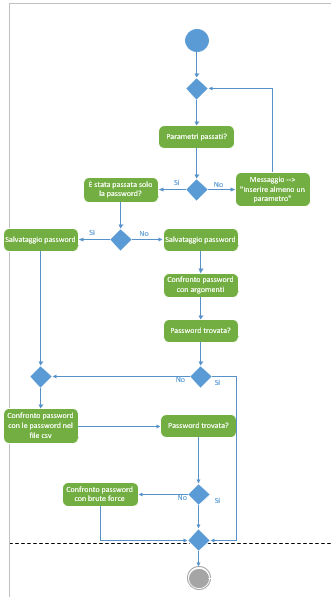
### Hardware

L’hardware utilizzato per questo progetto è il pc nelle aule scolastiche (Sistema operativo: Windows 10)

# Progettazione

## Design dell’architettura del sistema

Per descrivere l’architettura del sistema ho creato un activity diagram:

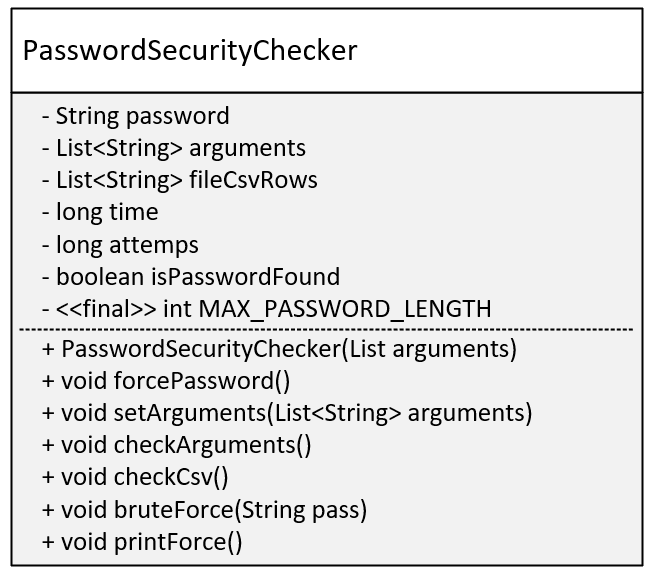


## Design dei dati e database

Gli unici dati salvati sono le 100’000 password più utilizzate. Sono salvate in un file csv “Most100kUsedPasswords.csv”

## Design procedurale

Nel progetto viene utilizzata una singola classa che ho descritto con un diagramma UML:



Classe:

* PasswordSecurityChecker: Unica classe presente nel progetto, la classe che permette la forzatura di una password passata dall’utente.

Attributi:

* private String password: attributo che contiene la password passata come primo argomento dall’utente.
* private List<String> arguments: la lista che contiene tutti gli argomenti passati dall’utente.
* private List<String> fileCsvRows: la lista che contiene tutte le password scritte sul file “Most100kUsedPasswords.csv” (le password più utilizzate).
* private long time: attributo che contiene il tempo impiegato dal programma per forzare la password.
* private long attemps: attributo che contiene il numero di tentativi svolti dal programma per forzare la password.
* private boolean isPasswordFound: attributo che permette di uscire dai cicli di forzatura della password se è stata trovata.
* private final int MAX\_PASSWORD\_LENGTH = 20: attributo che definisce la lunghezza massima della password passata dall’utente, impostata a 20.

Metodi:

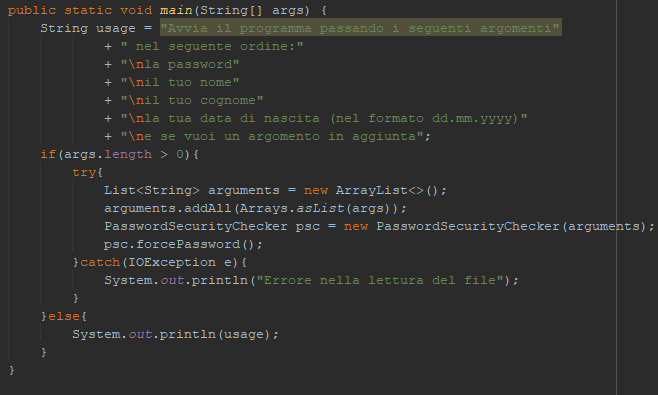
* public PasswordSecurityChecker(List arguments): il metodo costruttore della classe PasswordSecurityChecker a cui viene passata la lista degli argomenti immessi dall’utente.
* public void forcePassword(): è il metodo che gestisce il tempo impiegato e fa partire i metodi per eseguire i controlli sugli argomenti, il file csv e il brute force.
* public void setArguments(List<String> arguments) : è il metodo che genera possibili combinazioni di password con gli argomenti passati dall’utente.
* public void checkArguments(): è il metodo che fa passare tutti gli argomenti e le combinazioni degli argomenti per confrontarli con la password passata dall’utente.
* public void checkCsv(): è il metodo che fa passare tutte le password contenute nel file csv per confrontarli con la password passata dall’utente.
* public void bruteForce(String pass) : è il metodo che esegue il brute force, cioè la generazione di tutte le combinazioni possibili con tutti i carattareri (nel caso di questo progetto con i caratteri digitabili dalla tastiera Francese-Svizzera).
* public void printForce(): è il metodo che stampa una stringa che contiene il tempo impiegato e il numero di tentativi svolti quando la password viene forzata.

# Implementazione

## Avvio programma

Questo programma parte controllando se all’avvio vengono passati degli argomenti e se non vengono passati argomenti il programma stamperà un messaggio di che spiega in breve come far partire il programma.

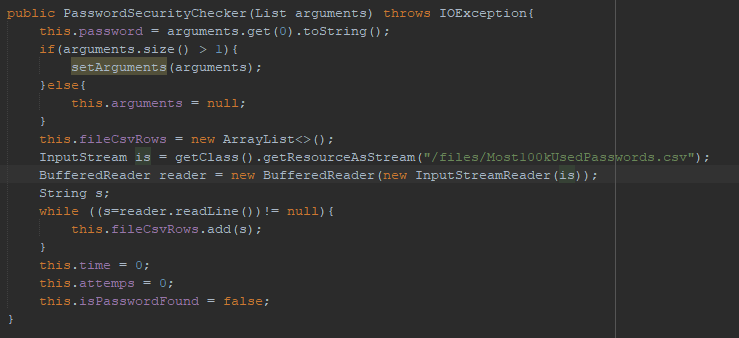
Se sono stati passati argomenti allora li prenderà e li inserirà in una lista che verrà passata come parametro al costruttore.



## Costruttore

All’interno del costruttore viene subito inserito il primo argomento nell’attributo password e successivamente viene controllato se ci sono ulteriori argomenti e se ci sono richiama il metodo “setArguments” (vedi punto 3.3) che crea le diverse combinazione tra gli argomenti passati e le inserisce nell’attributo “arguments” che è una lista in cui verranno contenute tutte le combinazioni, se oltre alla password non vengono passati argomenti il programma metterà l’attributo “arguments” come null.

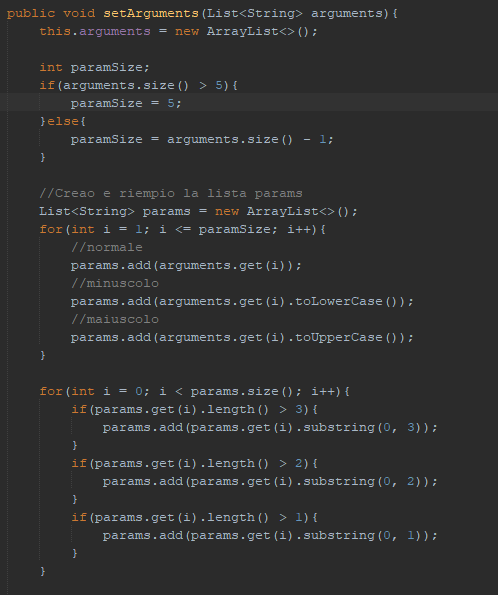
In seguito il programma andrà a leggere il file csv contenente le 100'000 password più utilizzate e le inserirà all’interno di una lista, l’attributo fileCsvRows. Per fare questo ho dovuto utilizzare un BufferedReader a cui viene passato un InputStream che contiene la stringa del percorso del file, ho dovuto utilizzare questo metodo di lettura dei file per poter leggere all’interno di un file jar.



## setArguments

Questo metodo permette di creare diverse combinazioni tra gli argomenti passati dall’utente, inizialmente controlla la quntità di argomenti passati e se superano i 5 parametri (password, nome, cognome, data di nascita e un argomento in più a scelta) quelli in più non vengono presi in considerazione.

Viene creata una lista “params” che conterrà gli argomenti passati così come sono che verranno poi aggiunti anche in minuscolo e quelli in maiuscolo. Una volta che il programma ha gli argomenti normali, in minuscolo e in maiuscolo, vengono presi e vengono create delle substring dell’inizio della stringa di 1, 2 e 3 caratteri e anche questi vengono aggiunti alla lista “params”.



Una volta che la lista “params” è completa il programma dovrà aggiungere gli elementi della lista e altre combinazioni all’interno dell’attributo “arguments”. Come prima cosa verranno aggiunti ad “arguments” tutti gli elementi della lista “params”. Poi inizierà a creare le combinazioni, verranno cretate combinazioni di 2 e 3 stringhe tra tutti gli elementi all’interno di “params”. Successivamente verrà controllato se è stata passata la data di nascita dall’utente e se essa è stata passata nel formato richiesto (dd.mm.yyyy), se è stata passata correttamente allora verranno presi i valori del giorno, del mese e dell’anno e verranno create combinazioni con uno e 2 elementi della lista “params”.



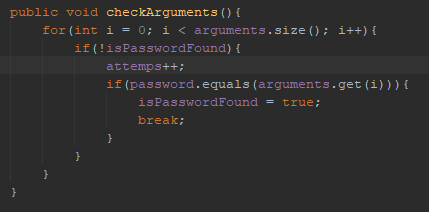
## forcePassword

Questo metodo dopo aver controllato che la password non superi la lunghezza massima, che è definita dalla costante MAX\_PASSWORD\_LENGTH prende il tempo di inizio, fa partire a uno a uno, se la password non è ancora stata trovata (controllando con l’attributo “isPasswordFound”), i metodi che controllano gli argomenti (punto 3.5), il file csv (punto 3.6) e il brute force (punto 3.7). Una volta finito il brute force viene preso il tempo di fine e viene calcolato il tempo impiegato per forzare la password, salvandolo nell’attributo “time”. Infine viene richiamato il metodo “printForce” (punto 3.8).



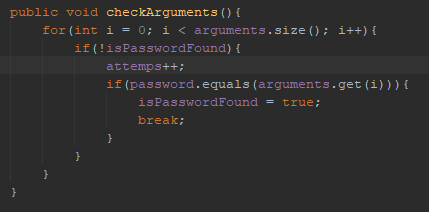
## checkArguments

Questo metodo fa passare tramite un ciclo for tutti gli elementi contenuti nella lista “arguments” e li confronta con la password, se viene trovata il metodo metterà la variabile “isPasswordFound” a true e uscirà dal ciclo.



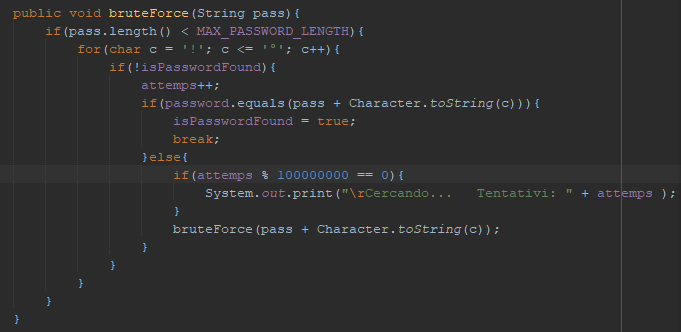
## checkCsv

Questo metodo fa passare tramite un ciclo for tutti gli elementi della lista “fileCsvRows”, che contiene le password contenute nel file csv, e li confronta con la password, se viene trovata il metodo metterà la variabile “isPasswordFound” a true e uscirà dal ciclo.



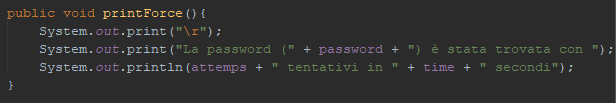
## bruteForce

Il brute force utilizza un ciclo for che passa tutti i caratteri compresi tra “!” e “°” che appunto contengono tutti i caratteri digitabili dalla tastiera Francese-Svizzera. Viene confrontata la password passata dall’utente con la password passata come parametro al metodo + il carattere a cui si trova il ciclo. La password passata come parametro al metodo non è altro che la combinazione di caratteri a cui ci troviamo nel ciclo + il carattere a cui ci troviamo nel ciclo attuale. Se la password viene trovata il metodo metterà la variabile “isPasswordFound” a true e uscirà dal ciclo, se non viene trovata richiemarà di nuovo il metodo “bruteForce” passando la combinazione di caratteri con in aggiunta il carattere del ciclo attuale, in oltre per mostrare a terminale che il programma sta andando e non si è bloccato ogni 100'000'000 di tentativi il metodo stamperà una stringa che contiene il numero di tentativi attuali.



## printForce

Questo metodo è semplicemete una stampa che viene eseguita quando la password è stata trovata. Stampa a terminale la password, il tempo impiegato e il numero di tentativi svolti per forzare la password



# Test

## Protocollo di test

Ecco i test case che ho definito:

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-001  REQ-02 | **Nome:** | Controllo passaggio di parametri dell’utente (password e altri parametri) |
| **Descrizione:** | Fare un controllo per vedere se è possibile passare I parametri e se il programma li salva nelle variabili giuste | | |
| **Prerequisiti:** |  | | |
| **Procedura:** | Avviare il programma e passare come primo parametro la password e in seguito altri parametri, salvare la password in una variabile globale e gli altri in una lista.  Stampare la variabile e gli elementi della lista.  Inoltre provare a non passare parametri per controllare se effettivamente stampa un messaggio di errore. | | |
| **Risultati attesi:** | Stampa dei parametri corretta. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-002  REQ-01 | **Nome:** | Lettura file csv |
| **Descrizione:** | Lettura del file csv inserendo tutti gli elementi in una lista | | |
| **Prerequisiti:** |  | | |
| **Procedura:** | Controllare se la lunghezza della lista corrisponde con la quantità di elementi nel file e poi stampare la lista per confrontarli con gli elementi del file e vedere se corrispondono. | | |
| **Risultati attesi:** | Lunghezza della lista corretta, le password corrispondono | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-003  REQ-04  REQ-06 | **Nome:** | Controllo funzionamento brute force |
| **Descrizione:** | Fare un controllo per vedere se viene trovata la password utilizzando più caratteri della tastiera possiblie | | |
| **Prerequisiti:** |  | | |
| **Procedura:** | Avviare il programma e vedere se trova le password che passi | | |
| **Risultati attesi:** | Stampa della password trovata | | |

## Risultati test

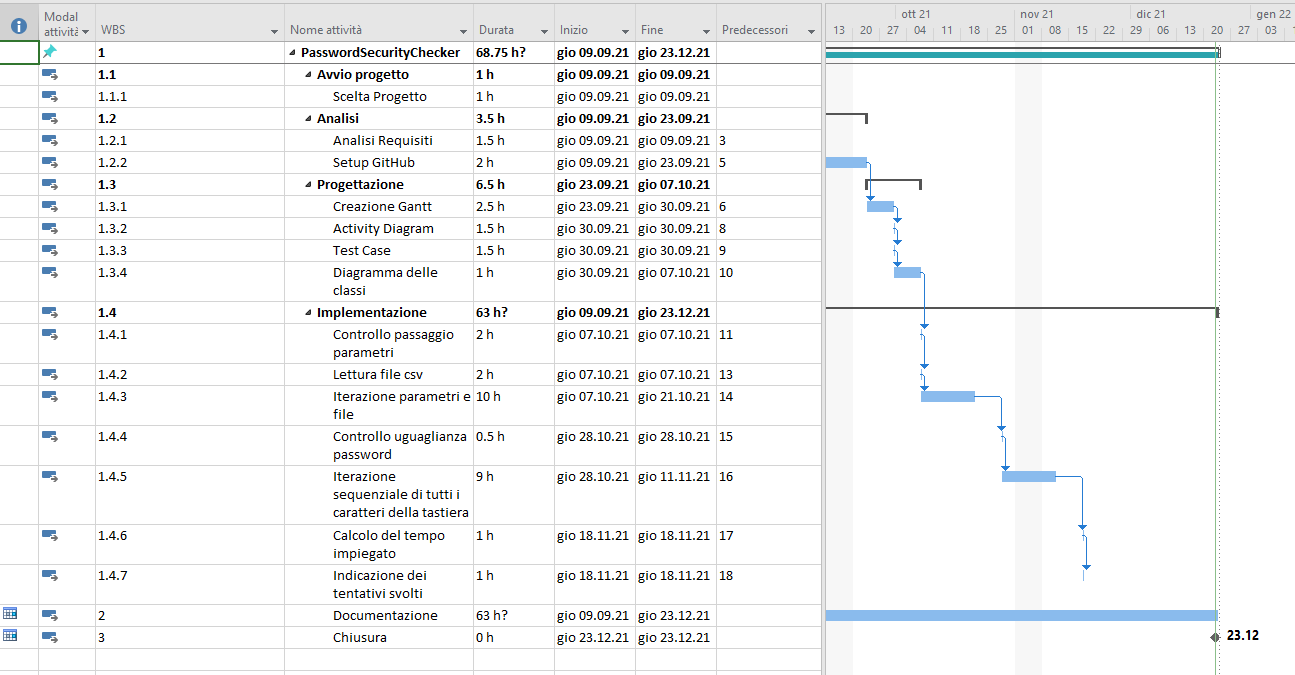
|  |  |
| --- | --- |
| Numero test | Risultato |
| TC-001 | Test funzionante,vengono stampati tutti i parametri correttamente |
| TC-002 | Test funzionante, viene letto il file correttamente e la lista viene riempita correttamente |
| TC-003 | Le password che ho provato sono state trovate tutte |

## Mancanze/limitazioni conosciute

L’unico problema del progetto è che il brute force è molto lento e con password di 20 caratteri ci mette giorni.

# Consuntivo

Gantt consultivo:



# Conclusioni

## Sviluppi futuri

Per velocizzare il brute force si potrebbe ricorrere all’utilizzo di thread così da avere cicli che funzionano in contemporanea per velocizzare la forzatura della password.

## Considerazioni personali

Personalmente questo progetto non l’ho trovato complicato e non ho imparato nulla di nuovo, per realizzarlo ho utilizzato le competenze che già avevo.

# Bibliografia

## Sitografia

Lista di password più utilizzate: <https://github.com/danielmiessler/SecLists/edit/master/Passwords/Common-Credentials/10-million-password-list-top-100000.txt>

Come leggere un file in un jar: <https://stackoverflow.com/questions/20389255/reading-a-resource-file-from-within-jar>

# Allegati

Elenco degli allegati:

* Diari di lavoro
* Codici sorgente
* Qdc