Documentazione Password Security Checker

1 Introduzione 3

1.1 Informazioni sul progetto 3

1.2 Abstract 3

1.3 Scopo 3

Analisi 4

1.4 Analisi del dominio 4

1.5 Analisi e specifica dei requisiti 4

1.6 Use case 6

1.7 Pianificazione 6

1.8 Analisi dei mezzi 6

1.8.1 Software 6

1.8.2 Hardware 6

2 Progettazione 7

2.1 Design dell’architettura del sistema 7

2.2 Design dei dati e database 7

2.3 Design delle interfacce 7

2.4 Design procedurale 7

3 Implementazione 8

4 Test 8

4.1 Protocollo di test 8

4.2 Risultati test 9

4.3 Mancanze/limitazioni conosciute 9

5 Consuntivo 9

6 Conclusioni 9

6.1 Sviluppi futuri 9

6.2 Considerazioni personali 9

7 Bibliografia 9

7.1 Bibliografia per articoli di riviste: 9

7.2 Bibliografia per libri 9

7.3 Sitografia 10

8 Allegati 10

# Introduzione

## Informazioni sul progetto

Sono un allievo della Scuola d’Arti e Mestieri a Trevano, della sezione informatica, nella classe I3AA, e mi chiamo Xavier Horisberger, e ho il ruolo di produttore. Il progetto è supervisionato dai docenti Geo Petrini e Luca Muggiasca che hanno il ruolo di clienti.

Il progetto è iniziato il 09.09.2021 e la consegna è per il 23.12.2021.

In questo capitolo raccogliere le informazioni relative al progetto, ad esempio:

* Allievo/i e docente/i coinvolti nel progetto e rispettivi ruoli,
* scuola, sezione, materia/e,
* date di inizio e termine di consegna,
* …

## Abstract

Questa documentazione contiene tutte le informazioni riguardante il progetto Password Security Checker.

È una breve e accurata rappresentazione dei contenuti di un documento, senza notazioni critiche o valutazioni. Lo scopo di un abstract efficace dovrebbe essere quello di far conoscere all’utente il contenuto di base di un documento e metterlo nella condizione di decidere se risponde ai suoi interessi e se è opportuno il ricorso al documento originale.

Può contenere alcuni o tutti gli elementi seguenti:

* **Background/Situazione iniziale**
* **Descrizione del problema e motivazione**: Che problema ho cercato di risolvere? Questa sezione dovrebbe includere l'importanza del vostro lavoro, la difficoltà dell'area e l'effetto che potrebbe avere se portato a termine con successo.
* **Approccio/Metodi**: Come ho ottenuto dei progressi? Come ho risolto il problema (tecniche…)? Quale è stata l’entità del mio lavoro? Che fattori importanti controllo, ignoro o misuro?
* **Risultati**: Quale è la risposta? Quali sono i risultati? Quanto è più veloce, più sicuro, più economico o in qualche altro aspetto migliore di altri prodotti/soluzioni?

Esempio di ab:

*As the size and complexity of today’s most modern computer chips increase, new techniques must be developed to effectively design and create Very Large-Scale Integration chips quickly. For this project, a new type of hardware compiler is created. This hardware compiler will read a C++ program, and physically design a suitable microprocessor intended for running that specific program. With this new and powerful compiler, it is possible to design anything from a small adder, to a microprocessor with millions of transistors. Designing new computer chips, such as the Pentium 4, can require dozens of engineers and months of time. With the help of this compiler, a single person could design such a large-scale microprocessor in just weeks.*

## Scopo

Lo scopo di questo progetto è creare un applicativo che permetta di forzare delle password, ovvero, andando a tentativi, scoprire che password ha inserito l’utente.

Lo scopo del progetto (scopi didattici/scopi operativi). Dovrebbe descrivere il mandato, ma non vanno ricopiate le informazioni del quaderno dei compiti (che va invece allegato).

## Analisi

## Analisi del dominio

Questo prodotto può essere utilizzato da chi ha java installato su pc, e ha una conoscenza minima di come eseguire un programma java da linea di comando, ed è anche utilizzabile su qualsiasi computer, non è importante che sistema operativo usa.

Questo capitolo dovrebbe descrivere il contesto in cui il prodotto verrà utilizzato, da questa analisi dovrebbero scaturire le risposte a quesiti quali ad esempio:

* Background/Situazione iniziale
* Quale è e come è organizzato il contesto in cui il prodotto dovrà funzionare?
* Come viene risolto attualmente il problema? Esiste già un prodotto simile?
* Chi sono gli utenti? Che bisogni hanno? Come e dove lavorano?
* Che competenze/conoscenze/cultura posseggono gli utenti in relazione con il problema?
* Esistono convenzioni/standard applicati nel dominio?
* Che conoscenze teoriche bisogna avere/acquisire per poter operare efficacemente nel dominio?
* …

## Analisi e specifica dei requisiti

|  |  |  |
| --- | --- | --- |
| ID | Requisito | Importanza |
| 1 | Possibilità di inserire nome, cognome, data di nascita, parola bonus e la password da controllare (solo la password è obbligatoria da inserire). | Alta |
| 2 | Non ci sono vincoli sulla password, può essere di qualsiasi lunghezza e può essere composta solamente dai caratteri premibili dalla tastiera “Svizzera”. | Medio |
| 3 | Controllo con le password più conosciute (usando file di testo) | Medio |
| 4 | Cercare di forzare la password inserita | Alta |
| 5 | Indicare i numeri di tentativi svolti | Medio |
| 6 | Indicare il tempo impiegato per forzare la password | Medio |
| 7 | Stampa un delle informazioni mentre il programma lavora per far capire che non si è bloccato | Medio |

Il progettista, dopo aver ricevuto il mandato, in collaborazione con il committente redige una lista di requisiti. Durante questi incontri, tramite interviste (da inserire nei diari), il progettista deve cercare di rispondere alle seguenti domande:

* Quali sono i bisogni del committente?
* Quali funzioni deve svolgere il prodotto?
* Come devono essere implementate?
* L’utente, come vorrebbe/dovrebbe interagire con il prodotto?
* Come verrà utilizzato il prodotto?
* Che tipo di interfaccia si immagina?
* Che prestazioni minime deve fornire il prodotto?
* Che grado di sicurezza deve avere il prodotto?
* …

In base alla lista dei requisiti e all’analisi degli stessi, il progettista redige una *specifica dei requisiti* in cui elenca e descrive in modo dettagliato quali sono le funzionalità che il prodotto fornirà. La specifica dovrebbe essere abbastanza dettagliata da poter essere utilizzata come base per lo sviluppo, ma non troppo; ad esempio non dovrebbe contenere dettagli di implementazione, o definizioni dettagliate dell’interfaccia grafica a meno che questi non siano considerati cruciali. Non si deve scordare che i requisiti non rappresentano delle attività bensì delle caratteristiche che il prodotto dovrà possedere.

|  |  |
| --- | --- |
| **ID: REQ-012** | |
| **Nome** | Creazione interfaccia banca dati |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | Si necessitano i permessi di root / Dipende dal requisito REQ-001 (Creazione DB) |
| **Sotto requisiti** | |
| **001** | Si necessita una maschera di login |
| **002** | Si dovranno poter immettere nuovi allievi |
| **003** | Dovrà essere possibile la ricerca di allievi |

**Spiegazione elementi tabella dei requisiti:**

**ID**: identificativo univoco del requisito

**Nome**: breve descrizione del requisito

**Priorità**: indica l’importanza di un requisito nell’insieme del progetto, definita assieme al committente. Ad esempio, poter disporre di report con colonne di colori diversi ha priorità minore rispetto al fatto di avere un database con gli elementi al suo interno. Solitamente si definiscono al massimo di 2-3 livelli di priorità.

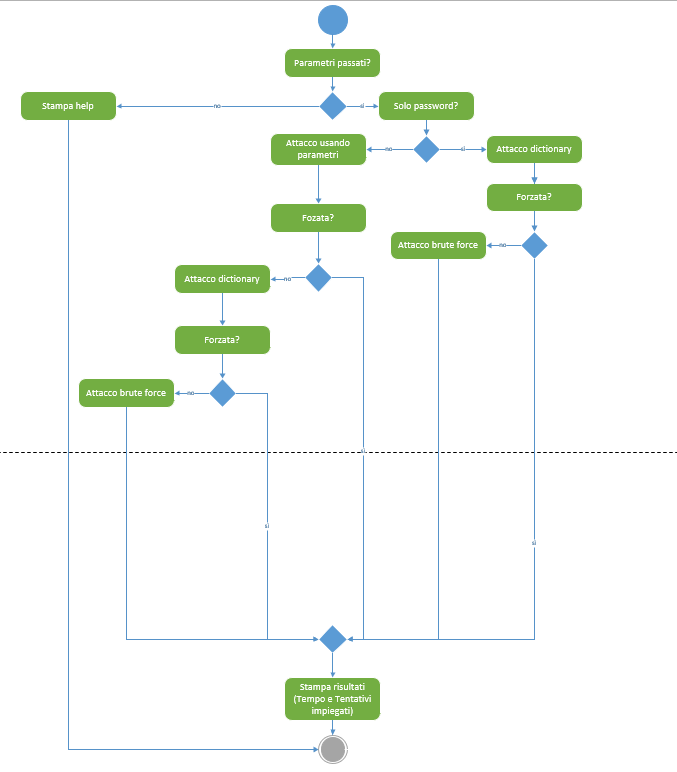
**Versione**: indica la versione del requisito. Ogni modifica del requisito avrà una versione aggiornata.

Sulla documentazione apparirà solamente l’ultima versione, mentre le vecchie dovranno essere inserite nei diari.

**Note**: eventuali osservazioni importanti o riferimenti ad altri requisiti.

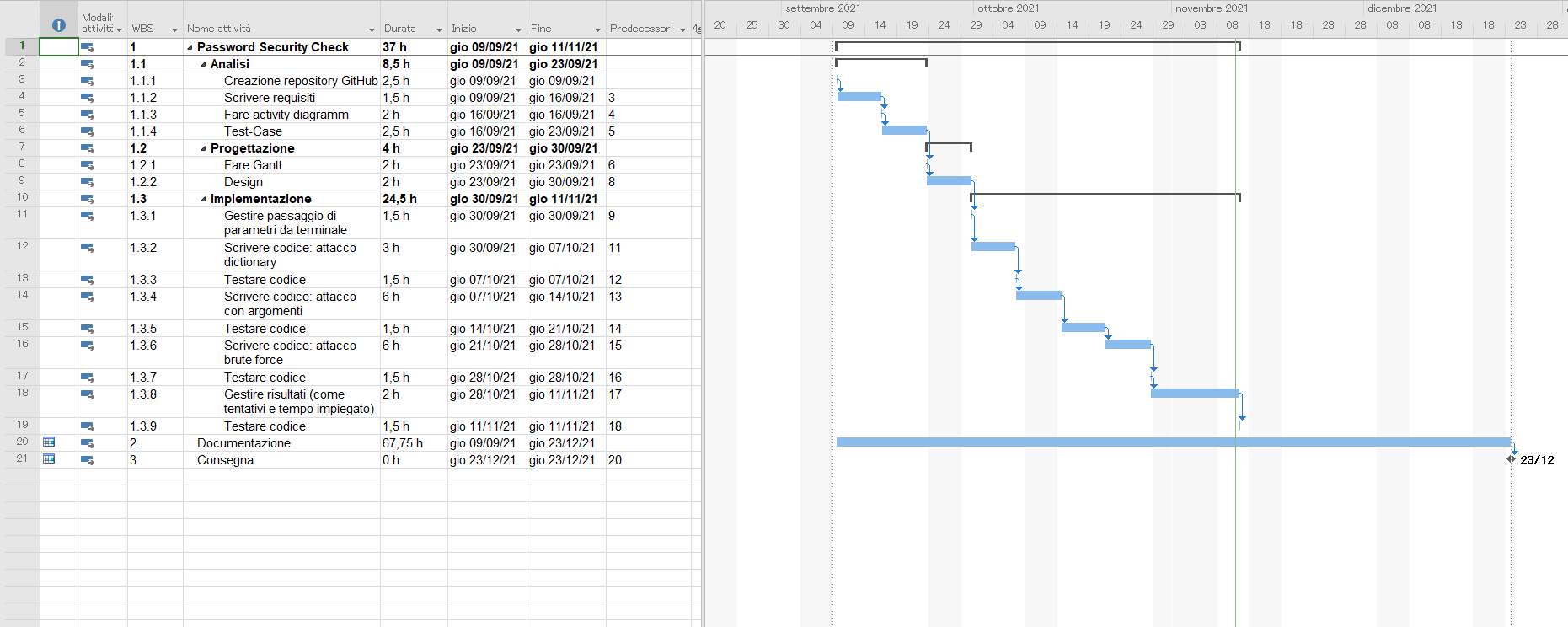
**Sotto requisiti**: elementi che compongono il requisito.

## Diagrammi



I casi d’uso rappresentano l’interazione tra i vari attori e le funzionalità del prodotto.

## Pianificazione



Prima di stabilire una pianificazione bisogna avere almeno una vaga idea del modello di sviluppo che si intende adottare. In questa sezione bisognerà inserire il modello concettuale di sviluppo che si seguirà durante il progetto. Gli elementi di riferimento per una buona pianificazione derivano da una scomposizione top-down della problematica del progetto.

La pianificazione può essere rappresentata mediante un diagramma di Gantt:

Se si usano altri metodi di pianificazione (p.es. scrum), dovranno apparire in questo capitolo.

## Analisi dei mezzi

Elencare e descrivere i mezzi disponibili per la realizzazione del progetto. Ricordarsi di sempre descrivere nel dettaglio le versioni e il modello di riferimento.

### Software

SDK, librerie, tools utilizzati per la realizzazione del progetto e eventuali dipendenze.

### Hardware

Su quale piattaforma dovrà essere eseguito il prodotto? Che hardware particolare è coinvolto nel progetto? Che particolarità e limitazioni presenta? Che HW sarà disponibile durante lo sviluppo?

# Progettazione

Questo capitolo descrive esaustivamente come deve essere realizzato il prodotto fin nei suoi dettagli. Una buona progettazione permette all’esecutore di evitare fraintendimenti e imprecisioni nell’implementazione del prodotto.

## Design dell’architettura del sistema

Diagramma delle classi:



Descrive:

* La struttura del programma/sistema lo schema di rete...
* Gli oggetti/moduli/componenti che lo compongono.
* I flussi di informazione in ingresso ed in uscita e le relative elaborazioni. Può utilizzare *diagrammi di flusso dei dati* (DFD).
* Eventuale sitemap

## Design dei dati e database

Descrizione delle strutture di dati utilizzate dal programma in base agli attributi e le relazioni degli oggetti in uso.

Schema E-R, schema logico e descrizione.

Se il diagramma E-R viene modificato, sulla doc dovrà apparire l’ultima versione, mentre le vecchie saranno sui diari.

## Design delle interfacce

Descrizione delle interfacce interne ed esterne del sistema e dell’interfaccia utente. La progettazione delle interfacce è basata sulle informazioni ricavate durante la fase di analisi e realizzata tramite mockups.

## Design procedurale

Descrive i concetti dettagliati dell’architettura/sviluppo utilizzando ad esempio:

* Diagrammi di flusso e Nassi.
* Tabelle.
* Classi e metodi.
* Tabelle di routing
* Diritti di accesso a condivisioni …

Questi documenti permetteranno di rappresentare i dettagli procedurali per la realizzazione del prodotto.

# Implementazione

## L’inizio

All’inizio del lavoro ho creato un progetto con NetBeans creando la classe PasswordSecurityChecker, ovvero il programma principale del progetto, e il file 100000-most-known-passwords.txt che conterrà le 100000 password più conosciute al mondo.

Dopo ciò ho cercato su internet le 100000 password più conosciute al mondo trovando un file su github a questo indirizzo: <https://github.com/danielmiessler/SecLists/blob/master/Passwords/Common-Credentials/10-million-password-list-top-100000.txt#L100000> che conteneva esattamente quello che cercavo. Ho preso il suo contenuto e l’ho messo nel mio file file 100000-most-known-passwords.txt.

## Scirvere la classe PasswordSecurityChecker

In seguito ho iniziato a scivere il contenuto della classe PasswordSecurityChecker.

### Attirbuti

Ho creato i seguenti attributi:

* arguments: lista di stringhe che conterrà i primi 4 argomenti passati dall’utente escludendo il primo (quindi gli argomenti da args[1] a args[4]).
* argumentCombos: lista di stringhe che conterrà le diverse combinazioni tra gli argomenti passati dall’utente.
* argumentCombosTemp: lista di stringhe che conterrà delle combinazioni tra argomenti che verranno poi aggiunte ad argumentCombos, essendo che non si possono fare aggiunte ad una lista mentre la si percorre, il che causerebbe una java.util.ConcurrentModificationException.
* mostKnownPasswords: lista di stringhe che conterrà le 100000 password più conosciute al mondo.
* MAX\_LEN: constante int che contiene la lunghezza massima possibile della password inserita dall’utente.
* password: questa variabile di tipo stringa conterrà il primo argomento passato dall’utente nell’array args, ovvero la password da trovare.
* foundPassword: questa variabile di tipo stringa conterrà la password scoperta dal codice.
* time: questa variabile conterrà il tempo complessivo che il programma ha impiegato a scoprire la password in millisecondi.
* tries: questa variabile conterrà il numero di tentativi svolti dal programma per scoprire la password.
* found: variabile booleana che indica se la password è stata scoperta (true) o no (false).
* characters: questa lista di stringhe conterrà tutti i caratteri necessari per trovare la password usando l’algoritmo brute force, ovvero tutti i caratteri della tastiera (più altri).
* help: questa variabile di tipo stringa contiene varie indicazioni utili a chi usa il programma nel caso non sappia come funziona e viene rilevata un’eccezione, in tal caso verrà stampato il contenuto di questa variabile.

### Metodi

In seguito agli attributi ho creato i seguenti metodi:

#### PasswordSecurityChecker(String[] argumetns)

Il costruttore della classe, che serve ad istanziare un oggetto di tipo PasswordSecurityChecker.

Per prima cosa viene assegnato un contenuto all’attributo help.

Questo metodo riceve come parametro un array di stringhe, il primo elemento verrà assegnato all’attributo password, se non è presente verrà rilevata un IllegalArgumentException, se invece è presente il suo valore verrà assegnato all’attributo password.

In seguito viene richiamato il metodo loadMostKnownPasswords() (che varrà descritto in seguito).

Dopodiché vengono salvati nell’attributo arguments i primi 4 elementi presenti dopo il primo, scartando quelli in più se presenti o prendendo quelli rimanenti se gli elementi dell’array sono minori di 5.

Dopo ciò se ci sono elementi nell’attributo arguemnts viene richiamato il metodo makeCombos() (che varrà descritto in seguito).

Infine l’attributo characters viene riempito con i caratteri della tabella ascii dal 33 al 255.

#### loadMostKnownPasswords()

Questo metodo legge il contenuto del file 100000-most-known-passwords.txt che su ogni riga contiene una password e ognuna di queste password verrà inserite nella lista mostKnownPasswords.

#### addIfValid(String word)

Questo metodo controlla se la parola word è presente in argumentCombos o in argumentCombosTemp se non è presente in nessuno dei due aggiunge word a argumentCombosTemp. La parola per essere aggiunta deve anche rispettare il limite di lunghezza impostato nella constante MAX\_LEN.

#### addCombos(String name, String surname, String word)

Questo metodo serve a creare varie combinazioni tra i parametri passati, se il name e il surname sono delle stringhe vuote allora a argumentCombosTemp verranno aggiunte solo le varie versioni di word (originale, maiuscolo e minuscolo).

Se solo surname è una stringa vuota verranno aggiunte a argumentCombosTemp name concatenato davanti a word (con tutte le varie versioni di name, originale, maiuscolo e minuscolo) e la stessa cosa viene fatta concatenando name alla fine di word.

Se tutti i parametri contengono qualcosa vengono concatenate davanti a word tutte le combinazioni possibili tra name e surname (ovvero name + name, name + surname, surname + name, surname + surname) e le stesse concatenazioni vengono fatte alla fine di word, e infine vengono aggiunte a argumentCombosTemp.

Per ogni aggiunta alla lista argumentCombosTemp fatta in questo metodo viene utilizzato il metodo addIfValid, per evitare che alla fine in argumentCombos ci saranno ripetizioni e che le combinazioni rispettino il limite di lunghezza.

#### addSubStrings(String word)

Questo metodo ritorna una lista contenente se possibile il primo carattere di word in prima posizione, in seconda posizione se possibile vengono aggiunti i primi due caratteri di word e in terza posizione se possibile vengo aggiunti i primi tre caratteri di word. Il contenuto di questa lista viene poi aggiunto a argumetnCombosTemp usando il metodo addCombos(“”,””,i).

#### addTempCombosToCombos()

Questo metodo serve ad aggiungere il contenuto di argumentCombosTemp a argumentCombos e svuotare argumentCombosTemp.

#### makeCombos()

Questo metodo serve a creare diverse combinazioni tra i vari argomenti passati dall’utente e salvarli nell’attributo argumentCombos.

Per prima cosa vengono salvate in argumentCombos tutti le varianti degli argomenti passati, tutti in maiuscolo, in minuscolo e le versioni originali, in seguito vengono aggiunte tutte te combinazioni (tra 2 argomenti) tra le varie versioni degli argomenti.

In seguito se il nome e il cognome sono presenti su di essi vengono usati i metodi addSubString(), e concatenando insieme i due risultati otteniamo la lista chiamata firstLettersNames, oppure se è presente solo il nome, nella lista firstLettersNames ci saranno solo le substring del nome, per poi richiamare il metodo addTempCombosToCombos(). A questo punto all’interno di argumentCombos avremmo tutti gli argomenti con le loro varie versioni, più tutte le combinazioni tra di essi e le substrings del nome e se possibile del cognome.

Dopo questo punto verranno aggiunte a argumentCombos tutte le combinazioni fatte con addCombos passando ogni elemento in firstLettersNames e ogni elemento in argumentCombos , poi verrà fatta la stessa cosa passando però a addCombos in entrambi i primi due parametri ogni elemento in firstLettersNames e ogni elemento in argumentCombos, in questo modo:

*for(String i : argumentCombos){*

*for(String j : firstLettersNames){*

*addCombos(j,"",i);*

*for(String h : firstLettersNames){*

*addCombos(j,h,i);*

*}*

*}*

*}*

Dopo ciò se presente verrà preso il terzo elemento di arguments, ovvero la dati di nascita e in un array verranno messe le varie parti di essa, giorno, mese, anno, e verrà creata una variabile contenente solo le ultime due cifre dell’anno di nascita, in seguito per ogni elemento all’interno di argumentCombos verranno aggiunti questi quattro numeri, una versione dove si aggiungono tutte e tre i numeri della data di nascita e una dove vengono aggiunte gionro+mese.

Infine vengono aggiunte singolarmente a argumentCombos giorno+mese e giorno+mese+anno se possibile.

Per ogni aggiunta alla lista argumentCombosTemp fatta in questo metodo viene utilizzato il metodo addIfValid, per evitare che alla fine in argumentCombos ci siano ripetizioni, ma principalmente per evitare che le combinazioni superino il limite di lunghezza.

#### printTriesAndTime()

Questo metodo serve a stampare i tentativi svolti al momento che il metodo viene richiamato e il tempo che ha attualmente impiegato a fare questi tentativi. Questo metodo viene verrà usato nei metodi per scoprire la password così che l’utente che usa il programma vedrà che il programma sta lavorando e non avrà il dubbio che si è bloccato.

#### finalPrint()

Questo metodo viene richiamato una volta che la password è stata scoperta e il programma è terminato, stampando la foundPassword, tries e time.

#### endForce(String s)

Questo metodo viene richiamato per terminare un force, ovvero gli algoritmi per cercare di scoprire la password, impostando l’attributo foundPassword a s, aumentando di uno l’attributo tries e impostando a true l’attributo found.

#### dictionaryForce()

Questo metodo percorre ogni elemento della lista mostKnownPasswords e le confronta con la password passata dall’utente, se essa combacia a una di queste password verrà invocato il metodo endForce passando la password in questione. Ogni 100 tentativi viene richiamato printTriesAndTime().

#### argumentsForce()

Questo metodo percorre ogni elemento della lista argumentCombos e le confronta con la password passata dall’utente, se essa combacia a una di queste password verrà invocato il metodo endForce passando la password in questione. Ogni 10000 tentativi viene richiamato printTriesAndTime().

#### bruteForce(String keys)

Questo metodo riceve inizialmente una stringa vuota, poi controlla che la lunghezza di keys non superi la lunghezza massima stabilita dalla costante. Poi inizia a percorrere l’attributo characters, controlla se l’attributo found sia false e se keys concatenato all’elemento corrente di characters sia uguale a password, se si verrà richiamato il metodo endForce e il programma terminerà, altrimenti verrà richiamato bruteForce passando keys + l’elemento corrente di characters, entrando in un ciclo che terminerà solo quando la password verrà trovata.

#### findPassword()

Questo è il metodo più importante, che mette insieme tutti gli altri metodi creati fino ad adesso. Per prima cosa viene richiamato il metodo argumentsForce, se lui trova la password viene richiamato finalPrint(), altrimenti si prova con il dictionaryForce(), se lui trova la password viene richiamato il finalPrint(), altrimenti viene richiamato il bruteForce(), e una volta che lui trova la password viene richiamato finalPrint(), altrimenti il programma non finirà.

In questo capitolo dovrà essere mostrato come è stato realizzato il lavoro. Questa parte può differenziarsi dalla progettazione in quanto il risultato ottenuto non per forza può essere come era stato progettato.

Sulla base di queste informazioni il lavoro svolto dovrà essere riproducibile.

In questa parte è richiesto l’inserimento di codice sorgente - Print Screen - di maschere solamente per quei passaggi particolarmente significativi e/o critici.

Inoltre, dovranno essere descritte eventuali varianti di soluzione o scelte di prodotti con motivazione delle scelte.

Non deve apparire nessuna forma di guida d’uso di librerie o di componenti utilizzati. Eventualmente questa va allegata.

Per eventuali dettagli si possono inserire riferimenti ai diari.

# Test

## Protocollo di test

Definire in modo accurato tutti i test che devono essere realizzati per garantire l’adempimento delle richieste formulate nei requisiti. I test fungono da garanzia di qualità del prodotto. Ogni test deve essere ripetibile alle stesse condizioni.

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-001  REQ-1 | **Nome:** | Test inserimento password |
| **Descrizione:** | Controllo che sia possibile inserire una password | | |
| **Prerequisiti:** |  | | |
| **Procedura:** | Aprire il terminale.  Utilizzare il programma da linea di commando inserendo una password per provare. (Esempio: Admin01!). Poi premere “enter” per far partire il programma. | | |
| **Risultati attesi:** | Il programma funziona senza problemi e alla fine stampa i tentativi e il tempo che ha impiegato a forzare la password. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-002  REQ-2  REQ-3  REQ-4  REQ-5  REQ-6 | **Nome:** | Test inserimento password e argomenti |
| **Descrizione:** | Controllo che sia possibile inserire una password e fino a 4 argomenti | | |
| **Prerequisiti:** |  | | |
| **Procedura:** | Aprire il terminale.  Utilizzare il programma da linea di commando inserendo una password come primo argomento, e poi inserire 4 parole per riempire 4 altri argomenti. (Esempio: Admin01! Xavier Horisberger Acquario Dormire). Poi premere enter per far partire il programma. | | |
| **Risultati attesi:** | Il programma funziona senza problemi e alla fine stampa i tentativi e il tempo che ha impiegato a forzare la password e quanto è sicura. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-003  REQ-7 | **Nome:** | Test stampa di informazioni nel mentre il programma lavora |
| **Descrizione:** | Mentre il programma funziona, deve stampare su una riga del terminale i tentativi svolti al momento e quanto tempo è passato in secondi. | | |
| **Prerequisiti:** |  | | |
| **Procedura:** | Aprire il terminale.  Utilizzare il programma da linea di commando inserendo una password come primo argomento, e poi inserire 4 parole per riempire 4 altri argomenti. (Esempio: Admin01! Xavier Horisberger Acquario Dormire). Poi premere enter per far partire il programma. | | |
| **Risultati attesi:** | Mentre il programma lavora vengono stampati i tentativi correnti e il tempo passato (su una sola riga, viene anche aggiornata dopo ogni tentativo). | | |

## Risultati test

Tabella riassuntiva in cui si inseriscono i test riusciti e non del prodotto finale. Se un test non riesce e viene corretto l’errore, questo dovrà risultare nel documento finale come riuscito (la procedura della correzione apparirà nel diario), altrimenti dovrà essere descritto l’errore con eventuali ipotesi di correzione.

## Mancanze/limitazioni conosciute

La limitazione più grande di questo programma è il fatto che se la password non viene trovata dai metodi argumentsForce o dictionaryForce e quindi potrà solo essere trovata con il bruteForce, e la password è più lunga di 4 caratteri il bruteForce non finirà mai, o per lo meno ci metterà troppo tempo a terminare.

Descrizione con motivazione di eventuali elementi mancanti o non completamente implementati, al di fuori dei test case. Non devono essere riportati gli errori e i problemi riscontrati e poi risolti durante il progetto.

# Consuntivo

Consuntivo del tempo di lavoro effettivo e considerazioni riguardo le differenze rispetto alla pianificazione (cap. 1.7) (ad esempio Gantt consuntivo).

# Conclusioni

Quali sono le implicazioni della mia soluzione? Che impatto avrà? Cambierà il mondo? È un successo importante? È solo un’aggiunta marginale o è semplicemente servita per scoprire che questo percorso è stato una perdita di tempo? I risultati ottenuti sono generali, facilmente generalizzabili o sono specifici di un caso particolare? ecc.

## Sviluppi futuri

Thread

Migliorie o estensioni che possono essere sviluppate sul prodotto.

## Considerazioni personali

Cosa ho imparato in questo progetto? ecc.

# Bibliografia

## Bibliografia per articoli di riviste:

1. Cognome e nome (o iniziali) dell’autore o degli autori, o nome dell’organizzazione,
2. Titolo dell’articolo (tra virgolette),
3. Titolo della rivista (in italico),
4. Anno e numero
5. Pagina iniziale dell’articolo.

## Bibliografia per libri

1. Cognome e nome (o iniziali) dell’autore o degli autori, o nome dell’organizzazione,
2. Titolo del libro (in italico),
3. ev. Numero di edizione,
4. Nome dell’editore,
5. Anno di pubblicazione,
6. ISBN.

## Sitografia

Sito: <https://stackoverflow.com/questions/20389255/reading-a-resource-file-from-within-jar>

Consultato il: 2021-09-23

Sito: <https://stackoverflow.com/questions/11377113/java-printing-lines-from-buffered-reader>

Consultato il: 2021-09-23

Sito: <https://github.com/danielmiessler/SecLists/blob/master/Passwords/Common-Credentials/10-million-password-list-top-100000.txt#L100000>

Consultato il: 2021-09-16

Sito: https://gist.github.com/jwoschitz/1129249 + <https://replit.com/@CyanCoding/C-Brute-Force-Password-Cracker>

Consultato il: 2021-09-30

1. URL del sito (se troppo lungo solo dominio, evt completo nel diario),
2. Eventuale titolo della pagina (in italico),
3. Data di consultazione (GG-MM-AAAA).

**Esempio:**

* http://standards.ieee.org/guides/style/section7.html, *IEEE Standards Style Manual*, 07-06-2008.

# Allegati

Elenco degli allegati, esempio:

* Diari di lavoro
* Codici sorgente/documentazione macchine virtuali
* Istruzioni di installazione del prodotto (con credenziali di accesso) e/o di eventuali prodotti terzi
* Documentazione di prodotti di terzi
* Eventuali guide utente / Manuali di utilizzo
* Mandato e/o QdC
* Prodotto
* …