Documentazione Trova la Parola

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

2.2.1 Spiegazione elementi tabella dei requisiti: 5

2.3 Use case 5

2.4 Pianificazione 5

2.5 Analisi dei mezzi 5

2.5.1 Software 6

2.5.2 Hardware 6

3 Progettazione 6

3.1 Design dell’architettura del sistema 6

3.2 Design dei dati e database 6

3.3 Design delle interfacce 6

3.4 Design procedurale 6

4 Implementazione 7

5 Test 7

5.1 Protocollo di test 7

5.2 Risultati test 8

5.3 Mancanze/limitazioni conosciute 8

6 Consuntivo 8

7 Conclusioni 8

7.1 Sviluppi futuri 8

7.2 Considerazioni personali 8

8 Glossario 8

9 Bibliografia 9

9.1 Bibliografia per articoli di riviste: 9

9.2 Bibliografia per libri 9

9.3 Sitografia 9

10 Allegati 9

# Introduzione

## Informazioni sul progetto

* Allievo del progetto: Marco Conforti
* Docente Responsabile: Geo Petrini
* Scuola: Scuola d’Arti e Mestieri Trevano
* Sezione: Informatica
* Materia: Modulo 306
* Data Inizio: 01.09.2023
* Termine Consegna: 07.12.2023

## Abstract

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 abstract:

*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 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

Il programma richiesto serve ad una rivista che pubblica giochi di enigmistica, il programma deve generare un gioco chiamato “trova la parola”, dove avendo una lista di parole si dovrà creare una tabella dove saranno presenti tutte le parole della lista, queste parole potranno essere messe in orizzontale in verticale o in senso obliquo, le parole possono essere da sotto verso sopra e viceversa.

## Analisi e specifica dei requisiti

|  |  |
| --- | --- |
| **ID: REQ-001** | |
| **Nome** | Implementazione generatore puzzlecruci |
| **Priorità** | Alta |
| **Versione** | 1.0 |
| **Note** | Dal sito web base si dovrà generare un puzzlecruci |
| **Sotto requisiti** | |
| **001** | Si dovrà avere un funzione per generare parole |
| **002** | Si dovrà avere un funzione per inserire le parole all’interno della griglia in modo corretto (direzione corretta, sovrapposizione no ma incroci si) |
| **003** | Si dovrà avere un funzione per visualizzare la tabella |
| **004** | Si dovrà avere un funzione per visualizzare la lista delle parole |

|  |  |
| --- | --- |
| **ID: REQ-002** | |
| **Nome** | Creazione Export + Possibilità di stampare |
| **Priorità** | Alta |
| **Versione** | 1.0 |
| **Note** | Bisogna creare un export in html e/o txt e la possibilità di stampare il puzzle crucis. |
| **Sotto requisiti** | |
| **001** | Si dovrà avere un funzione per esportare la tabella in html e/o txt |
| **002** | Si dovrà avere un funzione per la stampa |

|  |  |
| --- | --- |
| **ID: REQ-003** | |
| **Nome** | Creazione soluzione |
| **Priorità** | Media |
| **Versione** | 1.0 |
| **Note** | Si dovranno aggiungere la soluzione al puzzlecruci generato, utile anche come test. |
| **Sotto requisiti** | |
| **001** | Si dovrà avere un funzione per rilevare la soluzione. |

|  |  |
| --- | --- |
| **ID: REQ-004** | |
| **Nome** | Creazione sito web di base |
| **Priorità** | Alta |
| **Versione** | 1.0 |
| **Note** | Sito web dove aggiungere funzionalità aggiuntive. |
| **Sotto requisiti** | |
| **001** | Si dovrà aggiungere un funzione per far scegliere all’utente la modalità. |
| **002** | Si dovrà aggiungere un funzione per far scegliere all’utente il font. |
| **003** | Si dovrà aggiungere un funzione per far scegliere all’utente la difficoltà. |
| **004** | Si dovrà aggiungere un funzione per far scegliere all’utente parole nuove tramite un file. |

|  |  |
| --- | --- |
| **ID: REQ-005** | |
| **Nome** | Implementazione generatore puzzlecruci |
| **Priorità** | Media |
| **Versione** | 2.0 |
| **Note** | Implementazione di funzionalità aggiuntive. |
| **Sotto requisiti** | |
| **001** | Si dovrà aggiungere un funzione per cambiare la modalità (bambino, normale, parola nascosta casuale solo per normale, per bambino lettere casuali). |
| **002** | Si dovrà aggiungere un funzione per modificare il font della griglia. |
| **003** | Si dovrà aggiungere un funzione per cambiare la difficoltà della griglia anche in base alla modalità se modalità bambini n. parole se modalità normale grandezza campo. |
| **004** | Si dovrà aggiungere un funzione per far scegliere all’utente parole nuove tramite un file. |

### 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.

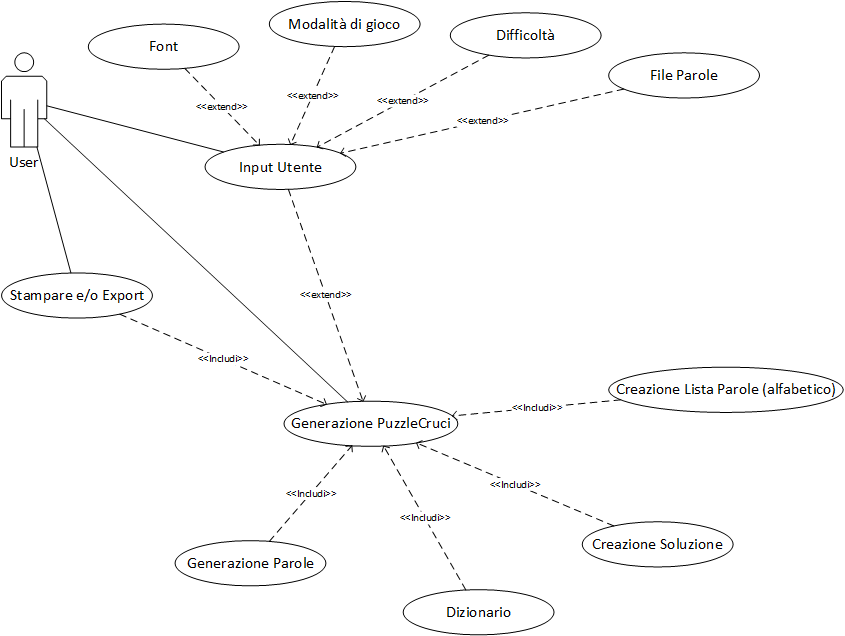
**Versione**: indica la versione del requisito. Ogni modifica del requisito avrà una versione aggiornata.

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.

## Use case



All’interno del mio applicativo ho esclusivamente un tipo di 1 utente.

* User: L’utente in questione avrà accesso a tre tipi di input, uno dove inserire degli input più specifici per la creazione del proprio Puzzle Cruci, quest’ultimo è un’estensione della generazione del Puzzle Curci. Un altro che appunto genera il Puzzle Cruci includendo anche la lista parole, la generazione delle parole, il dizionario e la soluzione. Infine la possibilità di esportare il Puzzle Cruci generato, anche questa funzionalità è inclusa nella generazione del Puzzle Cruci ma dovrà essere espressamente richiesta dall’utente a contrario della soluzione, la lista, la generazione delle parole e del dizionario.

## Pianificazione

|  |
| --- |
| Figura 1: Esempio di diagramma di Gantt. |

## Analisi dei mezzi

### Software

* **Visual Studio Code 1.78.2**
* **HTML 5**
* **Javascript 1.8.5**
  + **Estensione Live Server 5.7.9**
* **CSS 3**
* **AJAX 3.7**
* **Google Chrome 117.0.5938.150**

### Hardware

* **1 Server Web:** L’applicativo avrà bisogno un server web per funzionare, poiché richiama delle funzioni Ajax, durante lo sviluppo viene usata l’estensione Live Server.
* **1 Computer:** Il PC usato avrà le seguenti componenti:
  + CPU: i7-9700
  + RAM: 32 GB
  + SSD: 512 GB
  + SCHEDA VIDEO: NVIDIA GeForce RTX 2060

# 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

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

## Design delle interfacce

## 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

## GUI

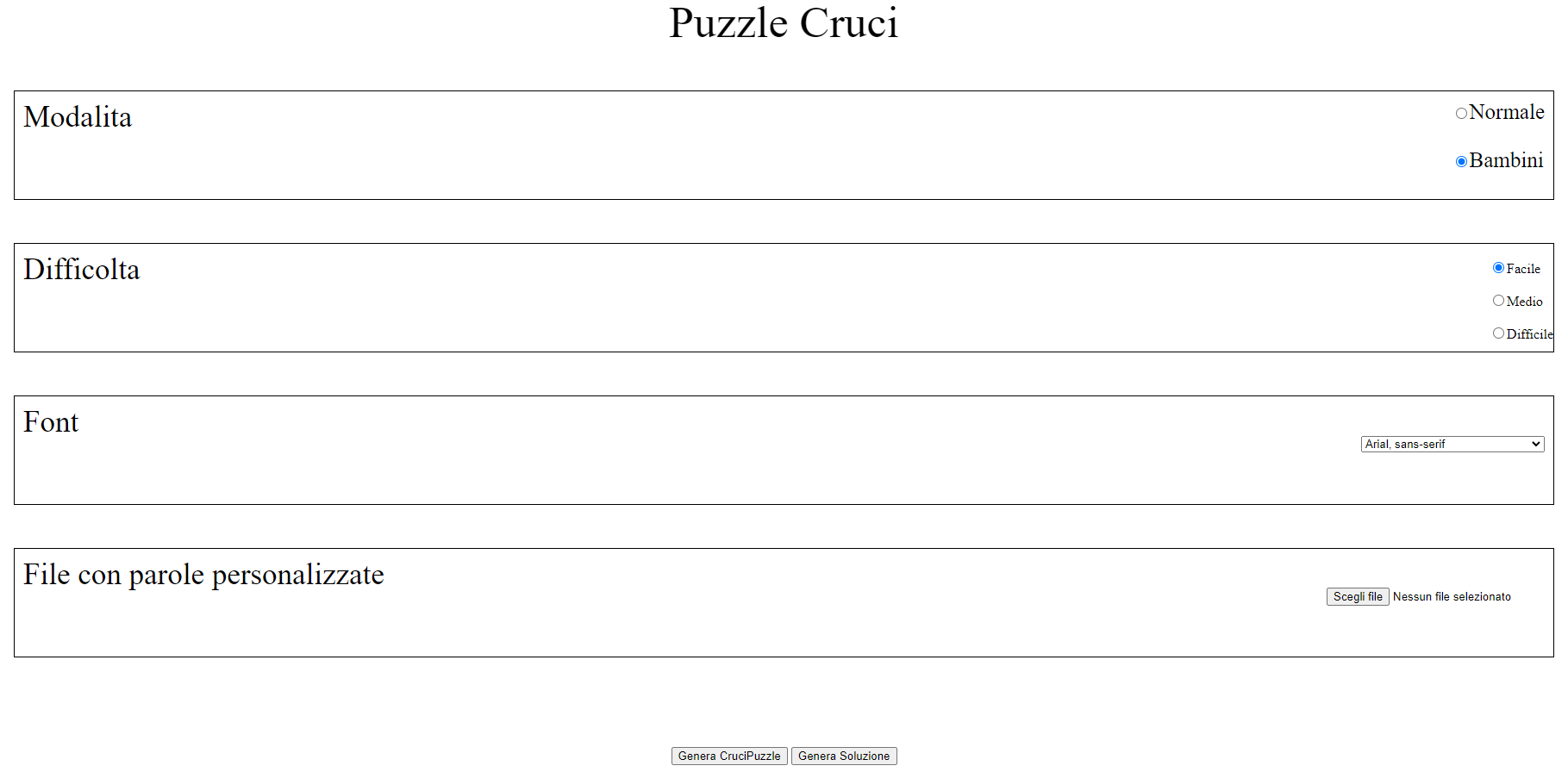
****

Figura 2 – GUI

Nella figura sopra è raffigurata la GUI finale del progetto. Possiamo notare che all’interno della GUI del sito ci sono due modalità una per bambini che è quella di default e quella normale, la differenza tra le due è che la prima non ha la parola nascosta ma lettere casuali al suo interno mentre la modalità normale contiene la parola nascosta che si potrà trovare una volta trovate tutte le parole della lista. Sotto questa funzionalità possiamo trovare la difficoltà che può essere di tre tipi facili, medio o difficili, la difficoltà cambia anche in base alla modalità infatti se abbiamo la modalità bambini e mettiamo facile avremo 15 parole da trovare in una matrice 10x10, con difficoltà media avremo 50 parole da cercare in 18x18, con difficoltà difficile avremo 70 parole da cercare in una matrice 22x22, se abbiamo invece la modalità normale possiamo e difficoltà facile avremo un numero indefinito di parole da cercare in una matrice 15x15, se inseriamo la difficoltà media avremo sempre un numero indefinito di parole da cercare in una matrice 20x20, se inseriamo la difficoltà difficile avremo un numero indefinito di parole in una matrice 25x25. Come penultima funzionalità da inserire troviamo il font fa modificare si possono scegliere 13 font diversi compreso il default. L’ultima funzione consente di importare un file di estensione “.txt”, all’interno del file si dovranno avere delle parole da inserire all’interno della matrice del gioco, maggiori informazioni sono nel capitolo dedicato al file con parole.

## Classe Gioco

La classe gioco si occupa della generazione del cruci puzzle e la sua soluzione aiutato dalle varie classi difficoltà, modalità e Parole. Oltre alla classe Font che funziona in modo autonomo, la classe Gioco è la sola classe che comunica con il file script.js che interagisce con index.html, quindi è la classe centrale del progetto.

### Costruttore

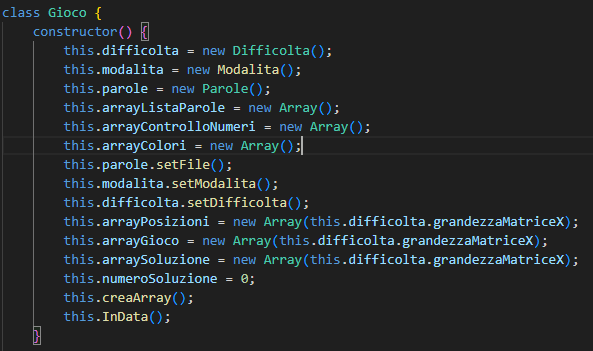


Figura 3 - Costruttore Classe Gioco

Nel seguente costruttore vediamo l’instanziazione della Classe *Difficolta* che serve soprattutto per la grandezza della matrice, inoltre vediamo l’instanziazione della Classe *Modalita* che serve principalmente per appunto sapere la modalità e quindi quale funzione deve fare la Classe, in seguito vediamo l’instanziazione della classe *Parole* che serve principalmente a prendere le parole dai dizionari e metterle dentro un array o anche per recuperare le possibili soluzioni stampabili e per la stampa tramite file. Inoltre verranno inizializzati gli array *ListaParole* che è appunto un array dove all’interno troviamo le parole del dizionario selezionato, l’array *arrayControlloNumeri* che salva al suo interno salva le parole che sono già state usate all’interno della matrice in modo che non ci possono essere doppioni, l’array *arrayColori* che serve appunto a salvare i colori usati nella soluzione in modo che non possono essere più utilizzati per altre soluzioni, l’array *arrayPosizioni* che ha al suo interno la direzione delle parole, questo serve a fare in modo che non ci siano due parole sovrapposte con la stessa direzione, l’array *arrayGioco* che al suo interno ha tutte le lettere del gioco ,l’array *arraySoluzione* che serve a tenere traccia delle parole dove si trovano all’interno della matrice grazie all’aiuto della variabile *numeroSoluzione* che per ogni parola ha un numero diverso e all’interno dell’array citato prima vengono inseriti questi numeri così da sapere dove si trovano le lettere della parola. Inoltre vengono invocati i seguenti metodi *setFile* dalla classe *parole*, *setModalita* dalla classe *Modalita*, *setDifficolta* dalla classe *Difficolta* e i metodi della classe Gioco *creaArray* e *InData*.

### Funzioni

All’interno di questo capitolo si vedranno i metodi della Classe Gioco.

#### Funzione creaArray

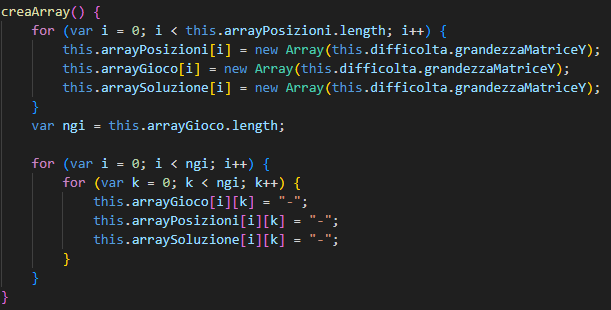


Figura 4 - Funzione creaArray

La Funzione creaArray serve appunto a inizializzare gli array mettendo un ulteriore array all’interno degli array già inizializzati nel costruttore, per ogni cella inserita nel costruttore viene messo un array, così facendo diventa la tabella che poi verrà stampata. Infine ho deciso di riempire gli array con il carattere *“-”* così da capire quali celle rimanevano vuote.

#### Funzione getDirezione

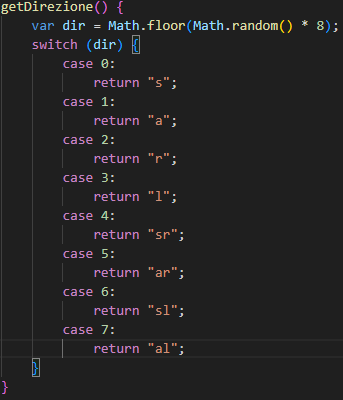


Figura 5 - Funzione getDirezione

La funzione getDirezione genera una posizione casuale per come inserire la parola se la funzione ritornerà **“s”** significa che la parola andrà dal lato destro a quello sinistro, mentre se ritornerà **“a”** significa che la parola andrà da destra verso sinistra, mentre se ritornerà **“r”** significa che la parola andrà dall’alto verso il basso, mentre se ritornerà **“l”** significa che la parola andrà da sotto verso sopra, mentre se ritornerà **“sr”** significa che la parola andrà dal alto sinistra verso il basso destra in obliquo, mentre se ritornerà **“ar”** significa che la parola andrà dal basso sinistra verso l’alto destra in obliquo, mentre se ritornerà **“sl”** significa che la parola andrà dal basso destra verso l’alto sinistra in obliquo, mentre se ritornerà **“al”** significa che la parola andrà dal alto destra verso basso sinistra in obliquo.

#### Funzione getPositionParola



Figura 6 - Funzione getPositionParola

La funzione getPositionParola serve a dare una coordinata (x;y) dove inizia la parola da inserire. Per ogni direzione la funzione gestisce la generazione dei due punti grazie alla lunghezza della parola che dovrà essere passata come parametro. Se la direzione è dall’altro verso il basso la coordinata y sarà limitata alle posizioni superiori mentre la x è liberà stessa cosa contraria se la direzione è dal basso verso l’alto. Il contrario sarà con la direzione da destra verso sinistra infatti la y rimarrà casuale mentre la x sarà limitata a dipendenza della direzione. Discorsi simili per le altre 4 posizioni, semplicemente nessuna coordinata sarà libera ma tutte limitate.

#### Funzione controllaParolaInGrid

#### Funzione InData



Figura 7 - Funzione InData - Modalità Normale

La Funzione InData serve principalmente a verificare che le parole sono state inserite tutte in base alla modalità inserita dall’utente e dopo che è stato verificato vengono invocati metodi per la buona riuscita del programma. Inizialmente carica il dizionario in un array, inoltre effettua la verifica che la parola possa essere effettivamente inserita all’interno dell’array senza essere troppo grande e alla verifica che la parola non sia effettivamente già stata inserita all’interno della matrice se rispetta queste due condizioni procede verso controlli specifici e se vengono rispettati si inserirà all’interno del array che controlla che non ci siano parole ripetute il numero della parola così non potrà essere ripetuta. All’interno dell’immagine vediamo la modalità normale, quella con la parola nascosta, infatti vediamo che non c’è un limite di parole all’interno del ciclo ma controlla se la parola magica può essere inserita nel caso che può essere inserita invocherà altri metodi per le varie stampe.

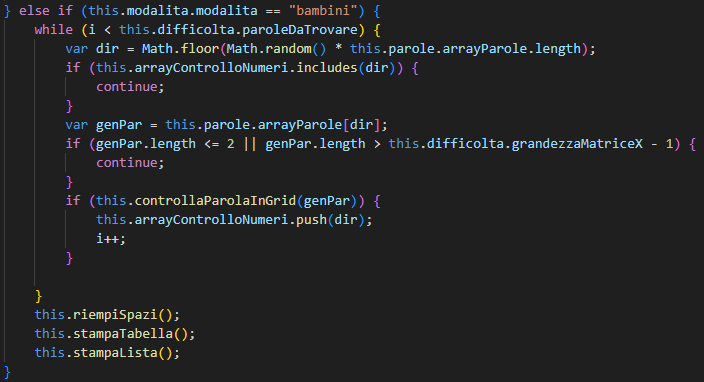


Figura 8 - Funzione InData - Modalità Bambini

Mentre nell’immagine sovrastante si vede la modalità bambini, con lettere casuali al posto delle lettere dalla parola nascosta, infatti vediamo che nel ciclo viene richiesto il numero di parole da inserire e dopo che sarà stato effettuato questo procederà con riempire gli spazi.

# 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-012 | **Nome:** | Import a card with KIC, KID and KIK keys, but not shown with the GUI |
| **Descrizione:** | Import a card with KIC, KID and KIK keys with no obfuscation, but not shown with the GUI | | |
| **Prerequisiti:** | Store on local PC: Profile\_1.2.001.xml (appendix n\_n) and Cards\_1.2.001.txt (appendix n\_n).  PIN (OTA\_VIEW\_PIN\_PUK\_KEY) and ADM (OTA\_VIEW\_ADM\_KEY) user right not set. | | |
| **Procedura:** | 1. Go to “Cards manager” menu,  in main page click “Import Profiles” link, Select the “1.2.001.xml” file, Import the Profile 2. Go to “Cards manager” menu,  in main page click “Import Cards” link, Select the “1.2.001.txt” file, Delete the cards,  Select the “1.2.001.txt” file, Import the cards 3. Research the “41795924770” Card, Click the imsi card link Check the card details 4. Execute the SQL: SELECT imsi, dir, keyset, cntr, rawtohex(kickey), rawtohex(kidkey), rawtohex(kikkey), rawtohex(chv), rawtohex(dap)FROM otacardkey a where imsi='340041795924770' ORDER BY keyset; | | |
| **Risultati attesi:** | Keys visible in the DB (OtaCardKey) but not visible in the GUI (Card details) | | |

## 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

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

Migliorie o estensioni che possono essere sviluppate sul prodotto.

## Considerazioni personali

Cosa ho imparato in questo progetto? ecc.

# Glossario

Inserite una semplice tabella con due colonne che spieghi i termini specifici del progetto (lista dei termini in ordine alfabetico A-Z)

Esempio:

|  |  |
| --- | --- |
| **Termine** | **Descrizione** |
| AJAX | **Asynchronous JavaScript And XML**: una tecnica che permette di eseguire richieste ed ottenere dati da una pagina web in modo asincrono. |
| CSS | **Cascading Style Sheets**: linguaggio che permette di definire il layout e la grafica di una pagina web. |

# Bibliografia

## Bibliografia per articoli di riviste:

## Bibliografia per libri

## Sitografia

* https://www.w3schools.com/, Learn to Code (JS, AJAX, CSS), 08-09-23
* https://stackoverflow.com/questions/14964035/how-to-export-javascript-array-info-to-csv-on-client-side, How to export JavaScript array info to csv (on client side), 20-10-2023
* https://www.html.it/pag/18966/bordi-arrotondati-con-i-css/, Bordi arrotondati con i CSS, 10-11-2023

# 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
* Gantt (Progettazione)
* Mockup
* Use Case
* Diagrammi di flusso