Documentazione Enea Corti

1 Introduzione 4

1.1 Informazioni sul progetto 4

1.2 Abstract 4

1.3 Scopo 4

2 Analisi 5

2.1 Analisi del dominio 5

2.2 Analisi e specifica dei requisiti 5

2.3 Use case 8

2.4 Pianificazione 9

2.5 Analisi dei mezzi 11

2.5.1 Software 11

2.5.2 Hardware 11

3 Progettazione 11

3.1 Design dell’architettura del sistema 11

3.2 Design delle interfacce 12

3.2.1 Design iniziale: 12

3.2.2 Design tabella: 14

3.2.3 Design della soluzione: 15

3.2.4 Design stampa in formato html: 16

3.2.5 Design della stampa in formato TXT: 17

3.3 Design procedurale 18

3.3.1 Genera Array: 18

3.3.2 Inserimento parola: 19

3.3.3 Controllo parola: 20

3.3.4 Inserimento parola nascosta: 21

4 Implementazione 22

4.1 Variabili globali: 22

4.2 Utente: 23

4.2.1 Funzione sceltaDimensione: 23

4.2.2 Funzione sceltaDifficolta: 24

4.2.3 Funzione font: 24

4.2.4 Funzione onChangeFont: 25

4.3 Funzione GeneraArray: 26

4.4 Funzione cordinate: 27

4.5 Lettura file: 28

4.5.1 Funzione fetchWords: 28

4.5.2 Funzione fileDefault: 28

4.5.3 Funzione readFile: 29

4.6 Riempimento Array: 30

4.6.1 Funzione pescaParola: 30

4.6.2 Funzioni di controllo inserimento: 31

4.6.3 Funzioni di inserimento: 32

4.6.4 Funzione riempiVuoti: 33

4.6.5 Funzione contaVuoti: 33

4.6.6 Funzione parolaNascosta: 34

4.6.7 Funzione nascosta: 35

4.7 Stampa: 36

4.7.1 Funzione print: 36

4.7.2 Funzione mostraElementi: 36

4.7.3 Funzione stampa: 37

4.7.4 Funzione stampaSoluzione: 38

4.7.5 Funzione randomRGBColor: 39

4.8 Download: 39

4.8.1 Funzione downloadText: 39

4.8.2 Funzione downloadHTML: 40

4.9 Media print: 41

5 Test 42

5.1 Risultati test 47

6 Consuntivo 48

7 Conclusioni 49

7.1 Sviluppi futuri 49

7.2 Considerazioni personali 49

8 Glossario 49

9 Bibliografia 50

9.1 Sitografia 50

9.2 Indice delle figure: 50

10 Allegati 50

# Introduzione

## Informazioni sul progetto

Progetto sviluppato dall’allievo Enea Corti sotto la supervisione di Geo Petrini nel periodo di tempo che va dal 01.09.2023 al 01.12.2023 ed andrà presentato in data 15.12.2023.

## Abstract

Con l’aumento della tecnologia si tende sempre di più ad abbandonare i vecchi giochi di società e i giornali.

Questo progetto è un ottimo miscuglio tra il moderno e il vecchio, permettendo di creare un gioco che permette di staccare un po’ dalla tecnologia in maniera automatica e rapida, accomuna questi due mondi in maniera ottimale

## Scopo

Lo scopo del progetto è quella di avere una pagina web che genera automaticamente il gioco del cruci puzzle.

Possono essere selezionate tre grandezze diverse per il campo e la difficoltà, che specifica se sarà presente o meno la parola nascosta.

Inoltre si può scegliere il file da dove verranno prese le parole ed il font con cui verrà generata la tabella e le parole ad essa sottostanti.

Si potrà far generare anche la soluzione ed il tutto può essere stampato in txt oppure html.

Viene gestita anche la stampa.

# Analisi

## Analisi del dominio

Il gioco potrà essere utilizzato da tutte le fasce di età, comprende la possibilità di scegliere il livello, se bambino oppure adulto e dà la possibilità di scegliere tra 3 grandezze di tabella.

Non occorre una particolare conoscenza di base per poter utilizzare il prodotto se non sapere come funziona il gioco, quindi quali incroci posso fare, in che direzione, eccetera.

Esistono molti atre pagine che eseguono la stessa operazione, questo progetto non vuole essere un affronto verso questi ultimi.

## Analisi e specifica dei requisiti

|  |  |
| --- | --- |
| **ID: REQ-01** | |
| **Nome** | Interfaccia |
| **Priorità** |  |
| **Versione** | 1.0 |
| **Note** |  |
| **Sotto requisiti** | |
| **001** | Solo creazione tabella e non gioco |
| **002** | Presenza del bordo intorno al rettangolo |
| **003** | Multicolonna per le parole |
| **004** | Nessuna immagine all’interno del rettangolo |

|  |  |
| --- | --- |
| **ID: REQ-02** | |
| **Nome** | Input dell’utente |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** |  |
| **Sotto requisiti** | |
| **001** | Possibilità di immettere un dizionario a scelta al posto del default. |
| **002** | Possibilità di scegliere la grandezza della tabella (piccola, media, grande) |
| **003** | Possibilità di scegliere livello (bambini o adulti) |
| **004** | Deve essere possibile scegliere il font (default Monospace) |

|  |  |
| --- | --- |
| **ID: REQ-03** | |
| **Nome** | Livello |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** |  |
| **Sotto requisiti** | |
| **001** | Per i bambini generare solo griglia con parole da trovare |
| **002** | Per gli adulti generare anche parola finale da trovare |

|  |  |
| --- | --- |
| **ID: REQ-04** | |
| **Nome** | Export |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** |  |
| **Sotto requisiti** | |
| **001** | La stampa è pensata per un foglio A4 |
| **002** | Formato TXT o HTML |

|  |  |
| --- | --- |
| **ID: REQ-05** | |
| **Nome** | Multipiattaforma |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** |  |
| **Sotto requisiti** | |
| **001** | Deve essere utilizzabile in multipiattaforma |

|  |  |
| --- | --- |
| **ID: REQ-06** | |
| **Nome** | Formattazione |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** |  |
| **Sotto requisiti** | |
| **001** | Tutte le lettere devono essere in maiuscolo |
| **002** | Nessun accento o apostrofo |
| **003** | Le parole devono essere in ordine alfabetico |

|  |  |
| --- | --- |
| **ID: REQ-07** | |
| **Nome** | Scrittura |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** |  |
| **Sotto requisiti** | |
| **001** | Le parole saranno scritte in tutte le direzioni |

|  |  |
| --- | --- |
| **ID: REQ-08** | |
| **Nome** | Sovrapposizione |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** |  |
| **Sotto requisiti** | |
| **001** | Le parole possono essere sovrapposte solo se in direzioni diverse |

## Use case

Figura 0 – use case

## Pianificazione

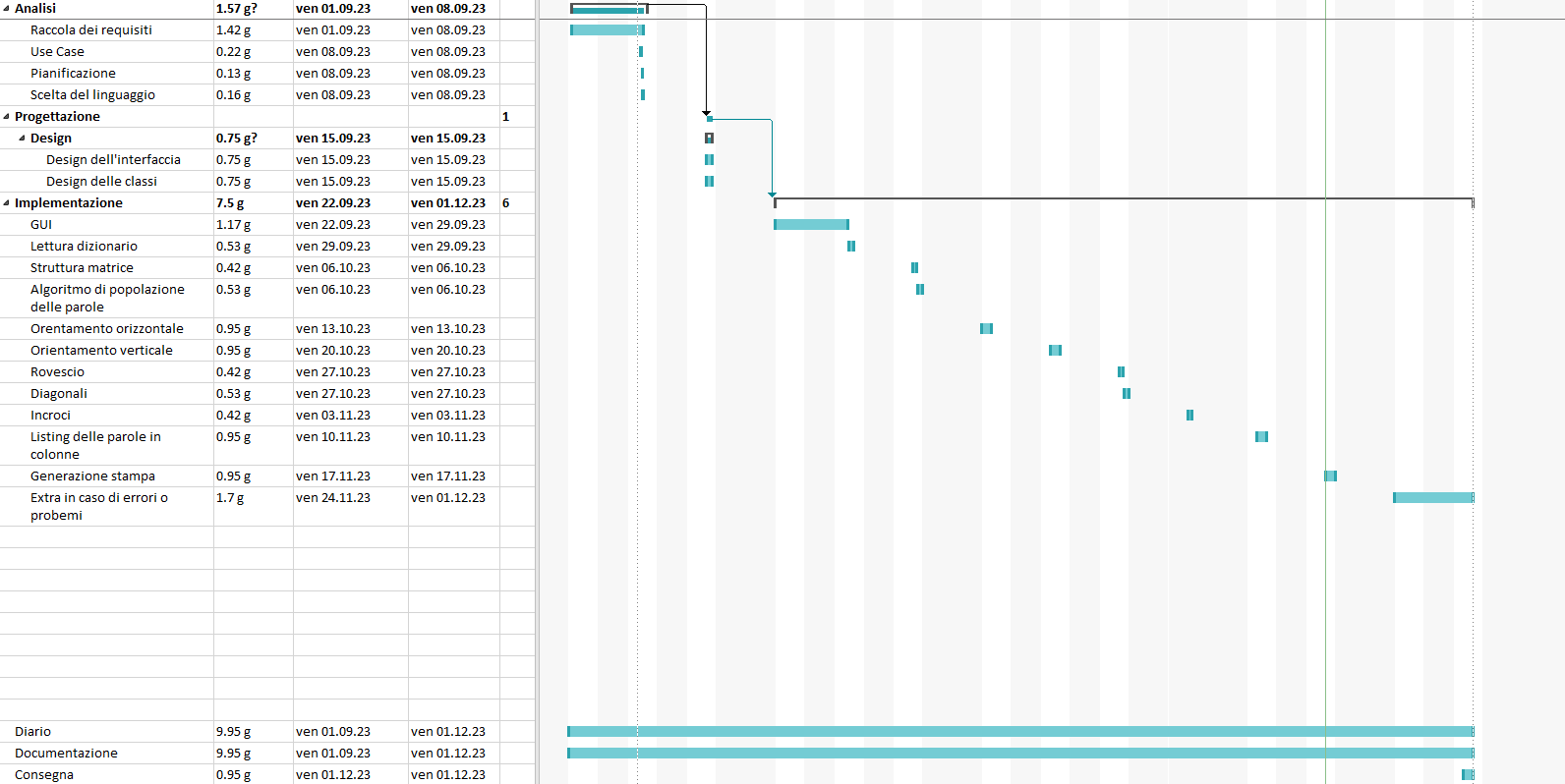


Figura 2 – Gantt della pianificazione

## Analisi dei mezzi

Per la realizzazione di questo progetto ho avuto a disposizione:

### Software

Visual studio Code 1.78.2

HTML 5

Javascript 1.5

### Hardware

1 pc con i seguenti componenti:

* CPU 🡪 Intel i7-9700
* Memoria 🡪 32 GB RAM
* Scheda grafica 🡪 NVIDIA GeForce RTX 2060
* SSD 🡪 512 GB

# Progettazione

## Design dell’architettura del sistema

Struttura del programma:

Immagine che contiene testo, schermata, Carattere, design

Descrizione generata automaticamente

Figura 1 – Struttura dei file

La parte più corposa del progetto è come si può notare la parte dedicata a javascript, in questi file c’è tutta la logica del programma, le varie funzioni che permettono di far funzionare la parte di generazione del campo.

I due file txt sono il file di default ed il file per la parola nascosta.

Il file index.css si occupa dello stile che deve avere la pagina mentre il file index.html si occupa di legare la parte di javascript con la parte di stile e da vita al vero e proprio progetto che verrà visualizzato dall’utente.

## Design delle interfacce

### Design iniziale:

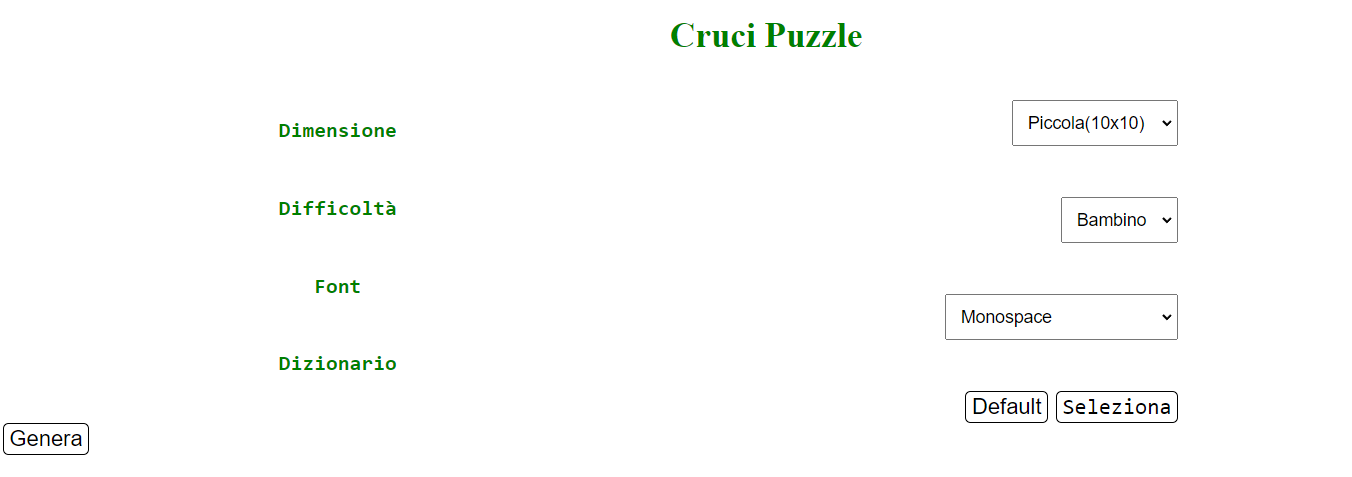


Figura 2 – Design dell’interfaccia iniziale che vede l’utente

La primissima cosa che mostro all’utente quando apre la pagina è questa, dove può scegliere la dimensione del campo di gioco, la difficoltà, il font da usare sia per le lettere della tabella che per le parole sottostanti.

Inoltre può scegliere se usare il dizionario di default oppure selezionarne uno personale con le parole scelte da lui.

### Design tabella:

Figura 3 - tabella generata è di dimensione grande, 20x20

Il design della tabella e delle parole sotto è semplice;

La tabella è posizionata al centro e al di sotto le parole che si devono trovare sono suddivise in colonne.

Il numero di colonne dipende dalla dimensione della tabella(ovvero del numero di parole da trovare).

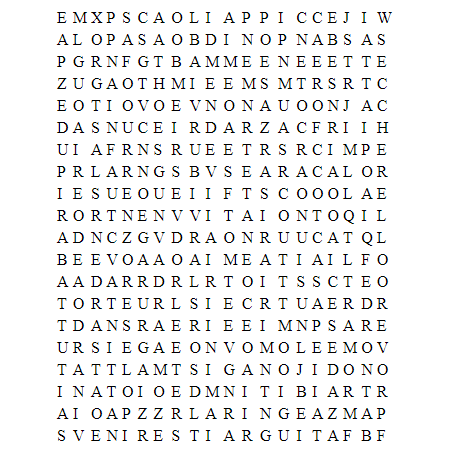
### Design della soluzione:

Figura 4 - soluzione per la tabella generata sopra, nella modalità bambino

Il design della tabella si basa sulle coordinate delle lettere per memorizzare un determinato colore, ad ogni parola viene allora assegnato un colore diverso che permette di distinguerle e quindi di capire dove si trovava una parola.

Nel caso in cui la modalità fosse impostata su adulto, le celle rimaste bianche sarebbero la soluzione alla parola nascosta, mentre in questo caso sono solo lettere inserite per completare il campo

### Design stampa in formato html:



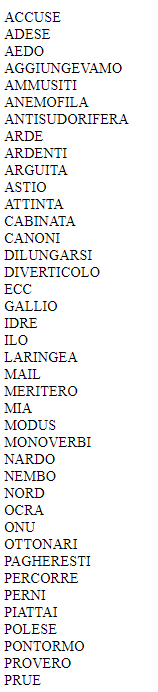


Figura 5 - Lista parole sotto alla tabella Figura 6 - Risultato del download nel formato html

In questo caso la stampa della tabella è priva di stile siccome viene riportato solo l’array contenete le lettere e non anche le informazioni riguardanti lo stile della tabella, come i vari bordi e i padding.

Anche la stampa delle parole sotto non ha stile, non viene divisa in colonne ma viene stampato l’array di parole da trovare in ordine come viene letto e senza nessuna divisione.

### Design della stampa in formato TXT:

Figura 7 – Risultato del download in formato TXT

Anche in questo caso essendo un file txt ovviamente non è presente nessun tipo di stile ne per la tabella ne per le parole stampate sotto di essa.

Per il campo le lettere che formano le parole da trovare sono suddivise in colonne e non sono il massimo da vedere.

Mentre le parole sotto riescono a mantenere la suddivisione di numero di parole per colonna, anche se non si dividono in colonne ma vengono semplicemente stampate una sotto l’altra.

## Design procedurale

### Genera Array:

Figura 8 – Diagramma di flusso funzionamento progetto

### Inserimento parola:

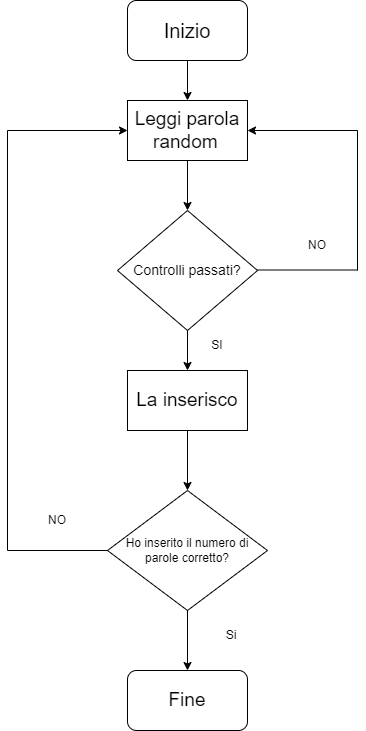


Figura 9 – Diagramma di flusso inserimento parola

### Controllo parola:

Figura 10 – Diagramma di flusso controllo parola

### Inserimento parola nascosta:

Figura 11 – Diagramma di flusso inserimento parola

# Implementazione

## Variabili globali:

Figura 12 – Lista di variabili globali

* listaParole 🡪 verrà riempita con tutte le parole presenti nel file passato come dizionario
* listaParoleNascosta 🡪 verrà riempita con tutte le parole presenti nel file locale delle parole nascoste
* parolaRan 🡪gli verrà assegnato il valore temporaneo della parola che viene pescata
* x 🡪 coordinata per le righe dell’array
* y 🡪 coordinata per le colonne dell’array
* dimensione 🡪 dimensione scelta per la tabella
* array 🡪 array che verrà riempito con le lettere che formeranno le parole da trovare
* arrayParole 🡪 verrà riempita con tutte le parole che sono state inserite nell’array
* pos 🡪 verrà usata come condizione per la durata del ciclo, stabilisce la fine della pesca delle parole
* pNascosta 🡪 verrà assegnata la parola nascosta pescata
* nParole 🡪 numero di parole che devono essere inserite nella tabella, il numero è il 15% di dimensione x dimensione
* direzione 🡪 usata per determinare che direzione avrà la parola con un random
* vuoti 🡪 verrà assegnato il numero di spazi vuoti che sono rimasti una volta inserite le parole
* indiceNascosta 🡪 usata per inserire la parola nascosta nell’array
* uguale 🡪 usata per determinare se sono state prese parole uguali
* arrayControllo 🡪 Usato per contare vuoti presenti una volta finito l’inserimento di parole
* dif 🡪 verrà assegnata la difficoltà scelta dall’utente
* coloriCelle 🡪 usata per stampare la soluzione con ogni parola colorata in modo diverso.

## Utente:

### Immagine che contiene testo, schermata, Carattere Descrizione generata automaticamenteFunzione sceltaDimensione:

Figura 13 – funzione per la scelta della dimensione

Nella funzione scelta dimensione prendo l’input del’utente per quanto riguarda la dimensione della tabella, io do le tre scelte al’utente, le tre dimensioni sono:

* Piccola 🡪 10 x 10
* Media 🡪 15 x 15
* Grande 🡪 20 x 20

A nParole, che è la variabile che uso per determinare il numero di parole che ci saranno da trovare, assegno il valore con un calcolo che si basa sulla dimensione, ovvero prendo il 15% della dimensione.

Inoltre inizializzo i due array, quello chiamato array e quello chiamato arrayControllo, e li creo della dimensione passata dall’utente.

### Funzione sceltaDifficolta:

Immagine che contiene testo, Carattere, schermata, linea

Descrizione generata automaticamente

Figura 14 – Funzione per la scelta della difficoltà

La funzione sceltaDifficolta serve per prendere la scelta dell’utente sulla difficolta, ovvero se bambino o adulto:

* Bambino 🡪 in questa modalità viene generata solo la tabella e viene richiamata la funzione di riempimento vuoti.
* Adulto 🡪 Viene generata la tabella e viene inserita una parola nascosta

### Funzione font:

Immagine che contiene testo, schermata, Carattere

Descrizione generata automaticamente

Figura 15 – Funzione che mi serve per selezionare il font

Questa funzione serve per mostrare la lista di possibili font che l’utente può selezionare, per farlo utilizzo arrayFonts che contiene una lista di font da poter scegliere, successivamente assegno ad ogni opzione assegno il valore corrispondente all’array.

Inoltre ho anche l’elento select che richiamala funzione onChangeFont che spiego qui sotto.

### Funzione onChangeFont:

Con questa funzione assegno alla tabella e alla lista di parole il font che l’utente ha scelto.

La funzione viene richiamata ogni volta che l’utente cambia il font selezionato.

Figura 16 – Diagramma di flusso inserimento parola

## Funzione GeneraArray:

Figura 17 – Funzione main

La funzione generaArray è relativamente semplice, è il main di questo progetto.

L’array dir mi servirà poco dopo per scegliere una direzione casuale da assegnare alla parola.

Nel primo ciclo for inizializzo i due array che diventano bidimensionali, mi serviranno da matrice per il gioco.

Nel secondo ciclo invece, che finisce quando ho inserito nella matrice il numero di parole che mi servono.

Successivamente giro la metà delle parole, così che quando le inserisco ho anche quel tipo di direzione.

Pesco una direzione casuale e richiamo la funzione cordinate che mi serve per prendere la x e la y, se le cordinate sono entrambe positive o sono 0 le utilizzo.

## Funzione cordinate:

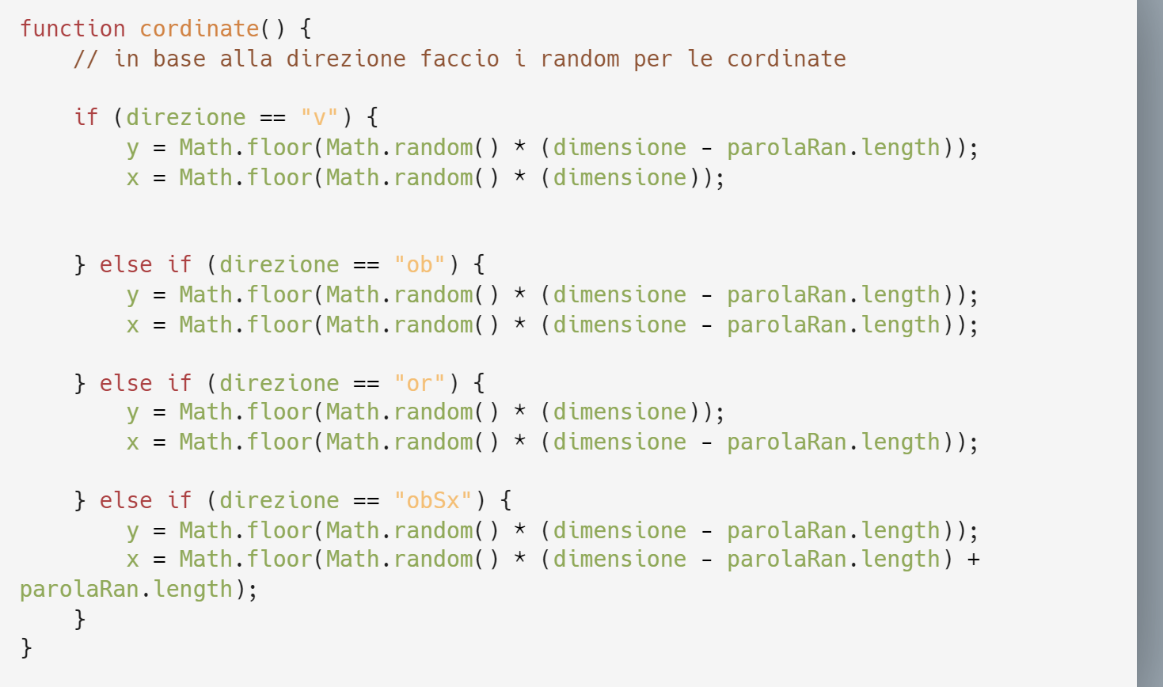


Figura 18 – Funzione per la scelta delle coordinate

Questa funzione si occupa della scelta casuale di una di una direzione che avrà la parola, in base alla coordinata che è uscita(v = verticale, ob = obliquo, or = orizzontale, obSx = obliquo a sinistra) genero le x e le y che non vadano fuori dalla dimensione della tabella una volta incrementate.

**Direzione = V:**

In questo caso la y dove posiziono la prima lettera non deve essere più grande della (dimensione della griglia – lunghezza della parola), così da no fuoriuscire dalla tabella

Mentre la x ovviamente non deve essere un numero più grande della larghezza dell’array.

**Direzione = OR:**

In questo caso faccio il contrario che facevo per il verticale, la x non deve essere più grande della (dimensione della griglia – lunghezza della parola).  
Mentre la y non deve essere un numero maggiore della altezza dell’array.

**Direzione = OB oppure direzione = OBSX:**

In questo caso sia la x che la y non dovranno essere maggiori della(dimensione – lunghezza della parola), così mi garantisco di non fuoriuscire dalla tabella.

## Lettura file:

### Funzione fetchWords:

Immagine che contiene testo, schermata, Carattere

Descrizione generata automaticamente

Figura 19 – Funzione per la lettura dei file

La funzione FetchWords mi serve per verificare la validità del file che ho passato(url), nel caso sia un file valido, lo trasformo in un array, altrimenti ritorno un errore.

Nel caso in cui il file non sia di tipo txt, la pagina manderà un alert contente un messaggio di errore dicendo che il file passato non è di tipo testuale, di conseguenza verrà utilizzato il file di default.

### Funzione fileDefault:



Figura 20 – Funzione per impostare il file di default

La funzione fileDefault è molto semplice, assegna alla costante words tutte le parole che sono contenute nel file locale dizionario.txt, dopodiché alla variabile listaParole assegna il valore di words

Infine richiama la funzione generaArray.

### Funzione readFile:



Figura 21 – Funzione per la lettura del file dell’utente

La funzione readFile, come dice il nome, si occupa della lettura del file contenete le parole che saranno poi quelle da trovare nel gioco.

Per prima cosa il valore dell’input dizionario viene assegnato alla costante fileInput, successivamente assegno alla costante chiamata file il primo valore di fileInput, nel caso fosse ci fosse più di un input.

Controllo che file non sia nullo, in caso che abbia un valore, ovvero che sia stato inserito un file, alla costante words, grazie al richiamo della funzione fetchWords, assegno tutte le parole contenute nel file.

Nel caso in cui il file passato abbia effettivamente al suo interno delle parole, assegno alla variabile listaParole il contenuto di words, successivamente chiamo la funzione generaArray vista prima.

In caso che no venga passato nessun file, oppure il file passato sia vuoto, richiamo la funzione fileDefault.

## Riempimento Array:

### Funzione pescaParola:



Figura 22 – Funzione per la pesca delle parole

Nella funzione pescaParola vado a leggere il file (il dizionario usato) e prendo una parola, per farlo utilizzo una variabile chiamata continua a cui assegno il valore true.

Successivamente creo un loop, alla variabile parolaRan prima assegno un numero casuale da 0 fino ad un massimo uguale alla grandezza della lista di parole, ovvero il numero di quante parole possibili ci sono.

Dopo uso sempre parolaRan come variabile, ma in questo caso le assegno una parola alla posizione del numero random generato poco prima.

Il ciclo for invece mi serve per passare tutto l’array di parole già inserite fino a questo punto, per controllare che non ce ne siano di uguali, in questo caso assegno alla variabile uguale il valore di false.

Per evitarmi di dover eseguire dopo controlli che mi rallentano il programma, prima id fare il ritorno controllo che la parola pescata abbia almeno tre caratteri, che la parola pescata abbia una lunghezza minore della dimensione e che non sia già stata pescata in precedenza.

Se tutti i requisiti sono come necessario, assegno a continua il valore di false, così che il ciclo finisce e la parola è stata presa.

### Funzioni di controllo inserimento:

Le mie funzioni di controllo per l’inserimento sono quattro, una per ogni possibile direzione che possono avere le parole quando le inserisco nell’array.

Un esempio di funzione di controllo che ho utilizzato:

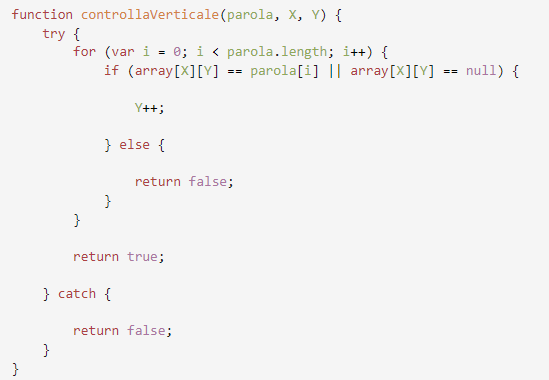


Figura 23 – Funzione per i controlli

In questo esempio mostro la funzione che mi serve per controllare la possibilità di inserimento per una parola in verticale.

Vengono passati tre valori; parola, X e Y.

Il programma prova a fare un ciclo di tutto l’array contenente le parole da trovare, e fa due controlli:  
Se alle coordinate dell’array si trova già la lettera della parola che si vorrebbe inserire, oppure a quelle coordinate la cella è vuota, procede con l’incremento delle Y in questo caso, siccome si tratta della funzione per il controllo verticale. Se alla fine del ciclo le condizioni sono state sempre rispettate ritorna true, mentre nel momento in cui una entrambe le due condizioni non sono rispettate si interrompe e ritorna false.

In questo caso viene incrementato solo Y per il controllo, ma come detto prima le funzioni di controllo sono quattro, nello specifico:

**ControllaOrizzontale**, con la stessa logica della funzione controllaVerticale spiegata in precedenza, l’unica differenza è che al posto di incrementare la Y per la verifica della possibilità di inserimento si incrementa X.

**ControllaObliquo,** anche questa funzione con la stessa logica delle precedenti, ma in questo caso entrambe le variabili X ed Y vengono incrementate per la verifica.

**ControllaObliquoSx,** stesso ragionamento e logica delle funzioni descritte sopra, anche in questo caso vengono toccate sia X che Y, ma con la differenza rispetto a prima che Y viene incrementata ad ogni ciclo, mentre X ad ogni ciclo viene decrementata.

### Funzioni di inserimento:

Anche in questo caso le funzioni sono quattro, come prima una per ogni direzione che posso avere, anche in

questo caso come esempio ho usato la funzione verticale:



Figura 24 – Funzione per l’inserimento

Le funzioni di inserimento le ho sviluppate con la stessa logica di quelle di controllo:

Per prima cosa, quando viene chiamata la funzione, richiamo la funzione di controllo mostrata prima, gli passo parolRan, che sarebbe la parola che ho pescato come mostrato nella funzione pescaParola, mentre x ed y, che sono le coordinate generate prima con la funzione coordinate.

Richiamo la funzione randomRGBColor, che spiegherò dopo quando parlo della stampa della soluzione, per assegnare il colore.

Ciclo tutta la parola appena pescata, alle coordinate x ed y assegno la lettera corrispondente della parola da inserire.

In questo caso incremento la y, mentre negli altri casi:

**Orizzontale 🡪** Incremento la x

**Obliquo 🡪** Incremento sia x che y

**ObliquoSx 🡪** Incremento y e decremento x

### Immagine che contiene testo, schermata, Carattere, linea Descrizione generata automaticamenteFunzione riempiVuoti:

Figura 25 – Funzione per il riempimento dei vuoti

Questa funzione serve nella modalità bambino, dove non c’è la parola nascosta e di conseguenza quando le parole da inserire sono finite rimarrebbero spazi senza niente al loro interno.

Questo problema lo risolvo con questa funzione, ho tutte le lettere dell’alfabeto nell’array, dopo aver inserito tutte le parole ciclo l’array contenete le parole da trovare, quando incontro una cella che ha valore undefined gli assegno una lettera randomica dell’alfabeto.

### Funzione contaVuoti:

Immagine che contiene testo, schermata, Carattere

Descrizione generata automaticamente

Figura 26 – Funzione per contare i vuoti nel campo

La funzione conta vuoti mi serve, come spiega il nome, per contare i vuoti che ci sono una volta inserito il numero di parole specificato.

Passo tutto il mio array contenente le parole, quando incontro una cella con valore undefined, incremento la variabile vuoti.

### Funzione parolaNascosta:

Immagine che contiene testo, schermata, Carattere, numero

Descrizione generata automaticamente

Figura 27 – Funzione per la pesva della parola nascosta

La funzione parola nascosta mi serve per prendere una parola nascosta da usare nel gioco.

Per prima cosa richiamo la funzione usata in precedenza per leggere da un file, a cui assegno il contenuto alla variabile listaParoleNascosta.

Richiamo la funzione per contare i vuoti presenti nel mi array, che mi serve per determinare di quanti caratteri dovrà essere composta la parola nascosta che devo pescare.

Successivamente creo un array chiamato nascostePossibili, che userò dopo per inserirci le parole che possono essere utilizzate come parola nascosta.

Nel primo ciclo controllo che la parola che si sta controllando ora sia della stessa lunghezza del numero dei vuoti presenti, in questo caso, aggiungo all’array nascostePossibili la parola in questione.

Successivamente controllo che l’array non sia vuoto e quindi che almeno una parola sia della lunghezza corretta.

In caso fosse vuoto faccio uscire un alert con un input per far inserire all’utente una parola nascosta a sua scelta della lunghezza adeguata.

Nel caso invece che l’array non sia vuoto, genero un numero casuale da 0 fino alla lunghezza dell’array e prendo una delle parole possibili casualmente.

### Immagine che contiene testo, schermata, Carattere Descrizione generata automaticamenteFunzione nascosta:

Figura 28 – Funzione per l’inserimento della parola nascosta

Nella funzione chiamata nascosta mi occupo di riempire con le lettere della parola nascosta gli spazi rimasti vuoti nel mio array.

Per prima cosa assegno a 0 la variabile indiceNascosta, che mi serve per determinare a quale lettera della parola nascosta si sta facendo riferimento.

Successivamente richiamo la funzione parolaNascosta vista in precedenza.

Ciclo tutto il mio array, quando incontro una cella con valore undefined, gli assegno la lettera della parola nascosta che mi serve.

Una volta assegnato il valore alla cella incremento la variabile dell’indice per prendere la lettera successiva la volta dopo.

Successivamente vado a stampare anche il numero di trattini corrispondente al numero di caratteri della parola, dove si può scrivere la soluzione una volta completato il gioco.

## Stampa:

### Funzione print:

Immagine che contiene testo, schermata, Carattere

Descrizione generata automaticamente

Figura 29 – Funzione per la stampa del campo

La funzione print mi serve per stampare la tabella contenente le parole che si devono trovare.

Per farlo utilizzo il metodo con i due cicli for, che mi permettono di passare tutto l’array.

La variabile table mi server per essere stampata successivamente, a quest’ultima, assegno ad ogni cella la lettera dell’array che corrisponde a quelle coordinate.

### Immagine che contiene testo, Carattere, schermata, linea Descrizione generata automaticamenteFunzione mostraElementi:

Figura 30 – Funzione per mostrare tasti di download e soluzione

La funzione serve per mostrare i tasti di download e di generazione della soluzione una volta che la tabella è stata stampata.

### Funzione stampa:

Immagine che contiene testo, schermata, Carattere, documento

Descrizione generata automaticamente

Figura 31 – Funzione stampa parole sotto alla tabella

Questa funzione mi serve per stampare sotto alla tabella le parole che vanno trovate nella tabella.

Prima di tutto però mi passo tutto l’array che contiene le parole che bisogna trovare all’interno della tabella, la metà di queste le giro, così da ottenere tutte le possibili combinazioni di stampa.

Successivamente stampo le parole, dividendole in colonne da otto parole.

Inoltre le parole sotto alla tabella le metto in ordine alfabetico facendo arrayParole.sort().

### Funzione stampaSoluzione:

Immagine che contiene testo, schermata, Carattere, documento

Descrizione generata automaticamente

Figura 32 – Funzione per stampare la soluzione

La funzione di stampa della soluzione si occupa appunto di stampare la soluzione del gioco del crucipuzzle.

Per farlo utilizzo due cicli for ed una variabile chiamata soluzione.

Il metodo di stampa è uguale a quello usato nella funzione print, la differenza sta nello stile della tabella.

Assegno un colore di sfondo diverso ad ogni parola (in precedenza ho salvato le coordinate della parola e ho assegnato a quest’ultimo un colore), per far in modo che le parole si riconoscano una volta che le stampo.

Per farlo, oltre a ciclare l’array con le lettere ciclo anche l’array contente le informazioni dei colori, così che quando sto ciclando una determinata lettera, assegno allo stile uno sfondo con il colore corrispondente nell’array chiamato coloriCelle.

Se la difficoltà è impostata su adulto, stampo anche quale era la parola nascosta che si doveva trovare.

### Funzione randomRGBColor:

Figura 33 – Funzione per generare colore casuale

La funzione mi serve per prendere un colore casuale, per farlo uso 3 random da 0 fino a 255, uno per ogni colore RGB(Red, green, blu).

Successivamente uso anche uno 0.5 in fondo alla stringa che mi serve per fare in modo che il colore abbia una leggera trasparenza per non essere troppo forte e di conseguenza coprire la visibilità della lettera.

## Download:

### Immagine che contiene testo, schermata, Carattere Descrizione generata automaticamenteFunzione downloadText:

Figura 34 – Funzione per il download in formato TXT

La funzione qui sopra mi serve in caso l’utente voglia scaricare la tabella in formato testuale.

Per farlo uso la variabile element a cui assegno gli elementi che hanno la classe download, successivamente creo un oggetto blob con al suo interno il contenuto della variabile element.

Creo un url per il blob che ho creato e successivamente creo anche un elemento a, ovvero un link a cui assegno come valore la variabile url che ho creato in precedenza.

Una volta che ho creato il link simulo un click per far in modo che il download avvenga.

### Funzione downloadHTML:

Immagine che contiene testo, schermata, Carattere

Descrizione generata automaticamente

Figura 35 – Funzione per il download in formato HTML

La funzione di download per l’html funziona della funzione downloadText, il procedimento passo passo è lo stesso cambia solo il modo in cui viene generato l’oggetto blob, che in questo caso è appunto un html.

## Media print:



Figura 36 – Stile per quando si stampa

Il media print mi serve per gestire la stampa, ovvero a nascondere gli elementi che non mi servono come i bottoni di generazione e di stampa.

Per farlo ho assegnato a tutti questi elementi che non voglio che siano mostrati la classe nascondi-print, a cui assegno il valore di display a none, così che quando stampo non siano visibili.

# Test

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-001  REQ-01 | **Nome:** | Controllo interfaccia |
| **Descrizione:** | Questo test serve per verificare che l’applicativo soddisfi le caratteristiche del primo requisito, nello specifico controllo che non ci siano immagini all’interno della griglia, che al suo esterno ci sia un bordo e che le parole sottostanti siano allineate ed in colonna. | | |
| **Prerequisiti:** | Browser con una connessione ad internet funzionante oppure codice sorgente con possibilità di avviare il progetto tramite server. | | |
| **Procedura:** | 1. Aprire il link oppure il progetto, ad esempio, tramite l’estensione di live server di visual studio code. 2. Lasciare le impostazioni di default, quelle saranno verificate in un altro test 3. Cliccare sul tasto Genera 4. Controllare che la tabella non abbia degli spazi vuoti al suo interno, che i bordi siano ben definiti. Inoltre, controllare anche che le parole sotto alla tabella siano suddivise in colonne. | | |
| **Risultati attesi:** | Alla generazione ci si aspetta di vedere la tabella al centro della pagina, con al di sotto le parole da trovare suddivise in colonne.  Inoltre la tabella dovrebbe essere contornata da un bordo nero | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-002  REQ-02-1 | **Nome:** | Dizionario |
| **Descrizione:** | Questo test serve per verificare che l’utente abbia a disposizione la possibilità di scegliere un suo dizionario di parole. | | |
| **Prerequisiti:** | Browser con una connessione ad internet funzionante oppure codice sorgente con possibilità di avviare il progetto tramite server.  Dizionario con le parole che si vogliono inserire | | |
| **Procedura:** | 1. Aprire il link oppure il progetto, ad esempio, tramite l’estensione di live server di visual studio code.  2. Cliccare sul bottone seleziona  3. Scegliere il file che si vuole usare  4. Cliccare sul bottone genera  5. Controllare che le parole generate siano quelle presenti nel file | | |
| **Risultati attesi:** | Ci si aspetta che il programma controlli il file passato, nel caso non fosse un txt, verrà mostrato un messaggio di errore verrà usato il file di default.  Nel caso il file passato fosse valido, ci si aspetta che le parole da trovare siano quelle presenti nel file. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-003  REQ-02 | **Nome:** | Grandezza tabella |
| **Descrizione:** | Questo test serve per verificare che la grandezza della tabella in base alla scelta dell’utente sia fatta in modo corretto. | | |
| **Prerequisiti:** | Browser con una connessione ad Internet funzionante oppure codice sorgente con possibilità di avviare il progetto tramite server. | | |
| **Procedura:** | 1. Aprire il link oppure il progetto, ad esempio, tramite l’estensione di live server di visual studio code.  2. Generare la tabella con la dimensione di default  3. Cliccare sul menu a tendina relativo alla dimensione  4. Selezionare la grandezza media  5. Cambiare nuovamente dimensione e scegliere quella grande  6. Generare nuovamente la tabella | | |
| **Risultati attesi:** | Il risultato aspettato è quello di vedere ad ogni generazione una grandezza diversa, nel primo caso che sia una tabella piccola, nel secondo media e nel terzo grande.  Ci si aspetta anche che lasciando le impostazioni base così come sono venga generata la tabella piccola. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-004  REQ-02 / REQ-03 | **Nome:** | Difficoltà |
| **Descrizione:** | Questo test serve per verificare che in base alla scelta dell’utente ci sia la difficoltà corretta | | |
| **Prerequisiti:** | Browser con una connessione ad Internet funzionante oppure codice sorgente con possibilità di avviare il progetto tramite server. | | |
| **Procedura:** | 1. Aprire il link oppure il progetto, ad esempio, tramite l’estensione di live server di visual studio code.  2. Generare la tabella con la difficoltà preimpostata  3. Controllare che non ci siano voci per la parola nascosta sotto alla tabella  4. Cliccare sul menu a tendina relativo alla difficoltà  5. Cambiare la difficoltà e mettere adulto  6. Generare nuovamente la tabella  7. Controllare che ci sia lo spazio per la parola nascosta | | |
| **Risultati attesi:** | Da questo test ci si aspetta che in base alla difficoltà che l’utente ha scelto ci sia una determinata funzione di gioco.  Nel caso della modalità bambino deve essere generata solo la griglia con le parole da trovare, nessuna parola nascosta, di conseguenza nessuno spazio al di sotto la tabella per poter inserire la parola nascosta.  Nel caso della modalità adulto ci si aspetta che oltre alle parole da trovare, le lettere rimanenti formino una parola che l’utente potrà poi scrivere nell’apposito spazio che apparirà al di sotto della tabella. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-005  REQ-02 | **Nome:** | Font |
| **Descrizione:** | Questo test serve per verificare che il font della tabella e delle parole sottostanti sia scelto dall’utente | | |
| **Prerequisiti:** | Browser con una connessione ad Internet funzionante oppure codice sorgente con possibilità di avviare il progetto tramite server. | | |
| **Procedura:** | 1. Aprire il link oppure il progetto, ad esempio, tramite l’estensione di live server di visual studio code.  2. Generare la tabella con il font di default  3. Controllare che sia monospace  4. Scegliere un font diverso dal menu a tendina con la voce font  5. Controllare che il font sia cambiato | | |
| **Risultati attesi:** | Ci si aspetta che in un qualunque momento, non per forza dopo ogni generazione, il font cambi in base a quello che sceglie l’utente.  Se non si toccano le impostazioni dovrebbe sempre essere monospace, in caso contrario ci si aspetta che il font cambia in tempo reale. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-006  REQ-04 | **Nome:** | Export |
| **Descrizione:** | Questo test serve per verificare che sia possibile esportare il progetto in formato txt ed in formato html | | |
| **Prerequisiti:** | Browser con una connessione ad Internet funzionante oppure codice sorgente con possibilità di avviare il progetto tramite server. | | |
| **Procedura:** | 1. Aprire il link oppure il progetto, ad esempio, tramite l’estensione di live server di visual studio code.  2. Generare la tabella con le impostazioni che meglio si preferisce  3. Cliccare sul tasto scarica html  4. Cliccare sul tasto scarica TXT  5. Controllare che i due file si siano scaricati e abbiano il contenuto corretto | | |
| **Risultati attesi:** | Quando si cliccano i due tasti ci si aspetta che vengano scaricati due file, uno in formato txt e l’altro in html.  Aprendo i due file si deve trovare solo il campo di gioco con le parole da trovare al di sotto di essa. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-007  REQ-05 | **Nome:** | Multipiattaforma |
| **Descrizione:** | Questo test serve per verificare che sia possibile esportare il progetto in formato txt ed in formato html | | |
| **Prerequisiti:** | Browser con una connessione ad Internet funzionante  VM con linux installato o un qualsiasi sistema operativo | | |
| **Procedura:** | 1. Aprire il link oppure il progetto, ad esempio, tramite l’estensione di live server di visual studio code.  2. Generare la tabella  3. Generare la soluzione  4. Scaricare in entrambi i formati | | |
| **Risultati attesi:** | Ci si aspetta che il sito sia funzionante su qualsiasi piattaforma e che anche tutte le funzionalità funzionino correttamente. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-008  REQ-06 | **Nome:** | Formattazione |
| **Descrizione:** | Questo test controlla che le parole siano scritte nel modo corretto secondo i requisiti. | | |
| **Prerequisiti:** | Browser con una connessione ad Internet funzionante oppure codice sorgente con possibilità di avviare il progetto tramite server. | | |
| **Procedura:** | 1. Aprire il link oppure il progetto, ad esempio, tramite l’estensione di live server di visual studio code.  2. Generare la tabella | | |
| **Risultati attesi:** | Da questo test ci si aspetta che la formattazione del documento sia fatta come richiesto.  Non ci devono essere parole doppie, non devono essere presenti apostrofi o accenti.  Le parole e le lettere devono essere scritte tutte in maiuscolo.  Le parole sotto alla tabella devono essere stampate in ordine alfabetico. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-009  REQ-07 | **Nome:** | Scrittura |
| **Descrizione:** | Questo test controlla che le parole all’interno della tabella siano state scritte in tutte le direzioni possibili. | | |
| **Prerequisiti:** | Browser con una connessione ad Internet funzionante oppure codice sorgente con possibilità di avviare il progetto tramite server. | | |
| **Procedura:** | 1. Aprire il link oppure il progetto, ad esempio, tramite l’estensione di live server di visual studio code.  2. Generare la tabella  3. Genera la soluzione  4. Controllare che le parole siano scritte come desiderato | | |
| **Risultati attesi:** | Il test serve per verificare che le parole all’interno della tabella siano scritte nelle seguenti direzioni:   * Verticale * Orizzontale * Obliquo * Obliquo verso sinistra   In tutti i casi le parole possono anche essere girate, per esempio nell’orizzontale al posto che trovare la parola che si legge da sinistra verso destra, la si può trovare da destra verso sinistra. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-010  REQ-08 | **Nome:** | Sovrapposizione |
| **Descrizione:** | Questo test controlla che le parole all’interno della tabella abbiano delle sovrapposizioni appropriate. | | |
| **Prerequisiti:** | Browser con una connessione ad Internet funzionante oppure codice sorgente con possibilità di avviare il progetto tramite server. | | |
| **Procedura:** | 1. Aprire il link oppure il progetto, ad esempio, tramite l’estensione di live server di visual studio code.  2. Generare la tabella  3. Genera la soluzione  4. Controllare che le parole siano sovrapposte come desiderato | | |
| **Risultati attesi:** | Con questo test si controlla che non vengano generate delle parole sovrapposte in modo inadeguato, ovvero che due parole che hanno la stessa direzione si sovrappongono.  Ci si aspetta che le parole all’interno della tabella siano sovrapposte solo nel caso in cui siano di direzione diversa. | | |

## Risultati test

|  |  |  |
| --- | --- | --- |
| Numero test: | Esito: | Commenti: |
| TC-001 | Passato |  |
| TC-002 | Passato |  |
| TC-003 | Passato |  |
| TC-004 | Passato |  |
| TC-005 | Passato |  |
| TC-006 | Passato |  |
| TC-007 | Passato |  |
| TC-008 | Passato |  |
| TC-009 | Passato |  |
| TC-010 | Passato |  |

# Consuntivo

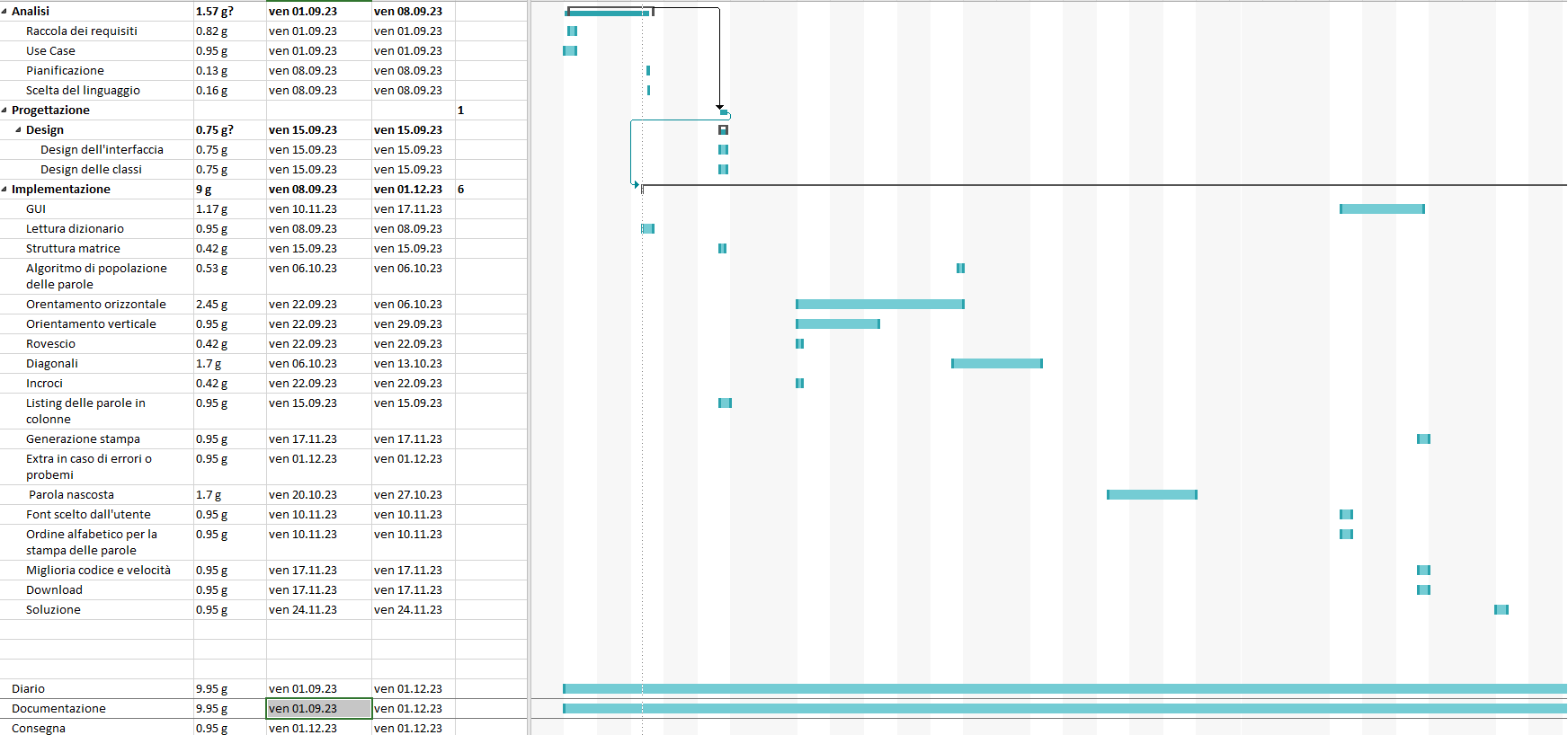


Figura 37 – Gantt di come è andata realmente

# Conclusioni

Il mio progetto non stravolge di certo il mondo, esistono molte altre alternative simili al mio progetto, penso però che nel suo modo di essere sia unico, grazie anche ai requisiti dati che ognuno decide di soddisfare in modi diversi.

## Sviluppi futuri

La miglioria che mi sarebbe piaciuta portare, ma che non ho implementato per la mancanza di tempo è la generazione della tabella migliorata, ovvero:  
Quando si genera la soluzione della tabella al momento se due parole si sovrappongono viene preso lo sfondo dell’ultima lettera che viene inserita in quella cella, in altre parole, se due parole si incrociano lo stile che viene preso è quello della parola che viene inserita per ultima in quella cella.

La miglioria che ho pensato di apportare è quella di creare un gradiente per gli incroci, ovvero un misto dei colori delle parole che si incrociano ai lati e al centro della cella lasciare il bianco con all’interno la lettera ben visibile.

## Considerazioni personali

Con questo progetto ho migliorato le mie competenze con js, nello specifico ho imparato ad usare i blob, a migliorare la mia capacità di ottimizzazione del codice.

Inoltre ho anche imparato a suddividere in maniera più efficiente le mie funzioni ed il mio file così da rendere più pulito da leggere ed è più semplice che funzione si occupa di cosa.

Credo anche di aver migliorato le mie capacità di organizzazione e di gestione del tempo durante un progetto.

# 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** |
| CSS | **Cascading Style Sheets**: linguaggio che permette di definire il layout e la grafica di una pagina web. |
| JS | **Javascript:** linguaggio di programmazione che permette di realizzare pagine Web interattive |

# Bibliografia

## Sitografia

1. <https://www.w3schools.com/>
2. <https://stackoverflow.com/>
3. <https://xkcd.com/color/rgb/>

## Indice delle figure:

[Figura 0 – use case 8](#_Toc152336746)

[Figura 2 – Gantt della pianificazione 9](#_Toc152336747)

[Figura 1 – Struttura dei file 10](#_Toc152336748)

[Figura 2 – Design dell’interfaccia iniziale che vede l’utente 11](#_Toc152336749)

[Figura 3 - tabella generata è di dimensione grande, 20x20 12](#_Toc152336750)

[Figura 4 - soluzione per la tabella generata sopra, nella modalità bambino 13](#_Toc152336751)

[Figura 5 - Lista parole sotto alla tabella Figura 6 - Risultato del download nel formato html 14](#_Toc152336752)

[Figura 7 – Risultato del download in formato TXT 15](#_Toc152336753)

[Figura 8 – Diagramma di flusso funzionamento progetto 16](#_Toc152336754)

[Figura 9 – Diagramma di flusso inserimento parola 17](#_Toc152336755)

[Figura 10 – Diagramma di flusso controllo parola 18](#_Toc152336756)

[Figura 11 – Diagramma di flusso inserimento parola 19](#_Toc152336757)

[Figura 12 – Lista di variabili globali 20](#_Toc152336758)

[Figura 13 – funzione per la scelta della dimensione 21](#_Toc152336759)

[Figura 14 – Funzione per la scelta della difficoltà 22](#_Toc152336760)

[Figura 15 – Funzione che mi serve per selezionare il font 22](#_Toc152336761)

[Figura 16 – Diagramma di flusso inserimento parola 23](#_Toc152336762)

[Figura 17 – Funzione main 24](#_Toc152336763)

[Figura 18 – Funzione per la scelta delle coordinate 25](#_Toc152336764)

[26](#_Toc152336765)

[Figura 19 – Funzione per la lettura dei file 26](#_Toc152336766)

[Figura 20 – Funzione per impostare il file di default 26](#_Toc152336767)

[Figura 21 – Funzione per la lettura del file dell’utente 27](#_Toc152336768)

[Figura 22 – Funzione per la pesca delle parole 28](#_Toc152336769)

[Figura 23 – Funzione per i controlli 29](#_Toc152336770)

[Figura 24 – Funzione per l’inserimento 30](#_Toc152336771)

[Figura 25 – Funzione per il riempimento dei vuoti 31](#_Toc152336772)

[Figura 26 – Funzione per contare i vuoti nel campo 31](#_Toc152336773)

[Figura 27 – Funzione per la pesva della parola nascosta 32](#_Toc152336774)

[Figura 28 – Funzione per l’inserimento della parola nascosta 33](#_Toc152336775)

[Figura 29 – Funzione per la stampa del campo 34](#_Toc152336776)

[Figura 30 – Funzione per mostrare tasti di download e soluzione 34](#_Toc152336777)

[Figura 31 – Funzione stampa parole sotto alla tabella 35](#_Toc152336778)

[Figura 32 – Funzione per stampare la soluzione 36](#_Toc152336779)

[Figura 33 – Funzione per generare colore casuale 37](#_Toc152336780)

[Figura 34 – Funzione per il download in formato TXT 37](#_Toc152336781)

[Figura 35 – Funzione per il download in formato HTML 38](#_Toc152336782)

[Figura 36 – Stile per quando si stampa 39](#_Toc152336783)

[Figura 37 – Gantt di come è andata realmente 46](#_Toc152336785)

# Allegati

* Diari di lavoro
* Codici sorgente
* Mandato e/o QdC
* Prodotto