DOCUMENTAZIONE PAROLE CROCIATE

# Indice

1 Indice 2

2 Introduzione 4

2.1 Informazioni sul progetto 4

2.2 Abstract 4

2.3 Scopo 4

3 Analisi 5

3.1 Analisi del dominio 5

3.2 Analisi e specifica dei requisiti 5

3.3 Use case 8

3.4 Pianificazione 9

3.5 Analisi dei mezzi 11

3.5.1 Software 11

3.5.2 Hardware 11

4 Progettazione 11

4.1 Design dell’architettura del sistema 11

4.1.1 Struttura dell’applicativo web 11

4.2 Design delle interfacce 12

4.3 Design procedurale 13

5 Implementazione 14

5.1 Lettura del file 14

5.1.1 LeggiContenuto() 15

5.1.2 Async leggo() 17

5.1.3 TrovaParoleConSpaziVuoti() 20

5.1.4 PrendiParola() 20

5.2 Popolare la tabella 21

5.2.1 PopolaTabella() 21

5.2.2 ControlloLettere() 22

5.2.3 InserisciLettere() 24

5.2.4 Evitare ciclo infinito 24

5.3 Stampare tabella 26

5.3.1 GeneraTabella() 26

5.3.2 StampaTabella() 27

5.4 Generare la soluzione 27

5.4.1 GeneraSoluzione() 28

5.5 Font 29

5.5.1 SetFont() 29

5.6 Export 30

5.6.1 ExportTxt() 30

5.6.2 ExportHtml() 30

5.6.3 SaveFile() 31

6 Test 32

6.1 Protocollo di test 32

6.2 Risultati test 36

6.3 Mancanze/limitazioni conosciute 40

7 Consuntivo 40

8 Conclusioni 42

8.1 Sviluppi futuri 42

8.2 Considerazioni personali 42

9 Bibliografia 43

9.1 Sitografia 43

10 Glossario 44

11 Indice delle figure 44

12 Allegati 45

# Introduzione

## Informazioni sul progetto

Il progetto consiste nel generare una “tabella” di lettere al cui interno ci sono delle parole da trovare.

Le parole da cercare sono fornite sotto la tabella sotto forma di elenco.

È possibile scegliere il font, il dizionario, la difficolta e la modalità.

È possibile esportare la tabella in formato txt e html.

È possibile generare la soluzione della tabella con marcate le parole nella lista e la possibile parola nascosta nella modalità adulti.

Persone coinvolte nel progetto:

* Riccardo Deiuri 🡪 principale programmatore e realizzatore del progetto
* Geo Petrini 🡪 Project Manager e aiutante

Scuola Arti & Mestieri Trevano, sezione informatica, Laboratorio progetti

Data di inizio progetto: 01.09.2023

Data di consegna: **01.12.2023**

## Abstract

To this day, crosswords can be easily found only in newspapers which nowadays are no longer used much, especially by the young population. Therefore I created this website which is used to generate a crossword easily and quickly. Then you can print or export the table to be able to play with it with a couple of clicks. It is also very useful for those who need many different crosswords in a short time.

All this, so that even young people are more encouraged to play, as it is also a game that can increase their vocabulary.

## Scopo

Lo scopo didattico del progetto è quello di riuscire a gestire al meglio la pianificazione e la realizzazione di un progetto. Per poi prepararci al futuro professionale.

Lo scopo operativo è quello invece di creare un applicativo WEB che generi un parole crociate affinché successivamente si possa stampare e giocarci.

# Analisi

## Analisi del dominio

Questo prodotto verrà utilizzato da tutti coloro che vogliono giocare a parole crociate ma non hanno delle riviste, oppure voglia di cercare in qualche sito dove troverebbe tabelle prese da giornali, potendo, invece in questo prodotto, generare una tabella facilmente e velocemente.

Questo progetto è destinato ad un’utenza universale, infatti l’utente finale potrebbe variare da un bambino di 6 anni a un anziano di 90 anni.

## Analisi e specifica dei requisiti

|  |  |
| --- | --- |
| **Req-001** | |
| **Nome** | Creazione interfaccia |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | Bisogna generare una tabella di lettere con all’interno delle parole messe in qualsiasi direzione, e elencarle sotto la tabella. |
| **Sotto requisiti** | |
| **Req-001\_1** | Si necessita di un dizionario |

|  |  |
| --- | --- |
| **Req-002** | |
| **Nome** | Modalità di gioco |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | Modalità bambini e modalità adulti (parola finale con lettere rimanenti) |

|  |  |
| --- | --- |
| **Req-003** | |
| **Nome** | Difficoltà di gioco |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | Si necessita di poter scegliere tra quattro difficoltà in cui varia la grandezza della tabella e il numero di parole. |
| **Sotto requisiti** | |
| **Req-003\_1** | Si necessita di un input per far scegliere l’utente. |

|  |  |
| --- | --- |
| **Req-004** | |
| **Nome** | Regole di gioco |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | Le parole non si possono sovrapporre a meno che non siano orientate diversamente l’una dall’altra. |
| **Sotto requisiti** | |
| **Req-004\_1** | Si necessita di non avere né accenti né apostrofi |
| **Req-004\_2** | Si necessita che tutte le lettere della tabella siano in maiuscolo |
| **Req-004\_3** | Si necessita che le parole possano essere in tutte le direzioni possibili: orizzontale, orizzontale invertito, verticale, verticale invertito, obliquo verso destra sotto, obliquo verso destra sopra, obliquo verso sinistra sotto, obliquo verso sinistra sopra. |

|  |  |
| --- | --- |
| **Req-005** | |
| **Nome** | Export |
| **Priorità** | 0.8 |
| **Versione** | 1.0 |
| **Note** | Si necessita inserire la possibilità di esportare la tabella in formato .txt |

|  |  |
| --- | --- |
| **Req-006** | |
| **Nome** | Scelta del font |
| **Priorità** | 0.5 |
| **Versione** | 1.0 |
| **Note** | Si necessita di inserire la possibilità di scegliere il font |

|  |  |
| --- | --- |
| **Req-007** | |
| **Nome** | Inserimento dizionario |
| **Priorità** | 0.5 |
| **Versione** | 1.0 |
| **Note** | Si necessita di inserire la possibilità di aggiungere un dizionario scelto dall’utente. |

|  |  |
| --- | --- |
| **Req-008** | |
| **Nome** | Generare soluzione |
| **Priorità** | 0.5 |
| **Versione** | 1.0 |
| **Note** | Si necessita di inserire la possibilità di generare la soluzione della tabella generata. |

**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

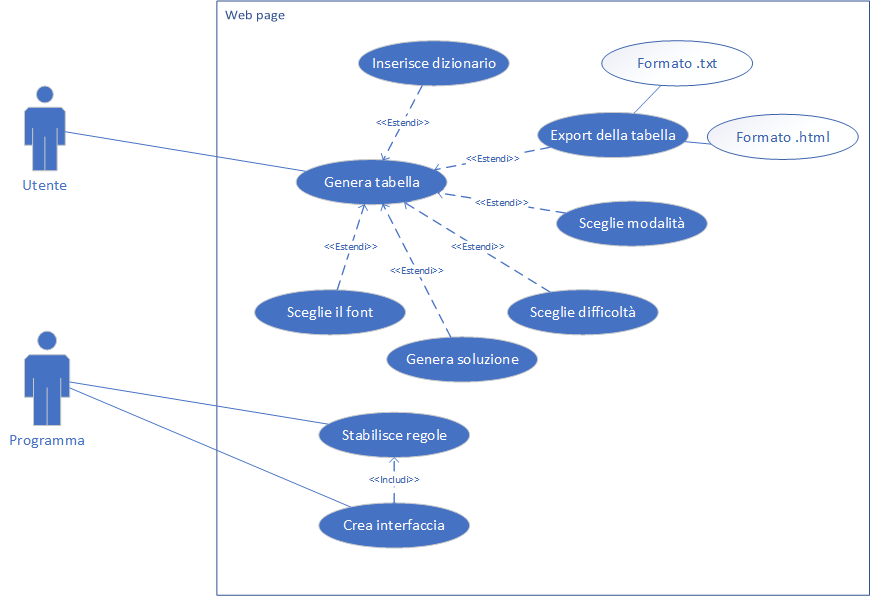


Figura 1 UseCase

## Pianificazione

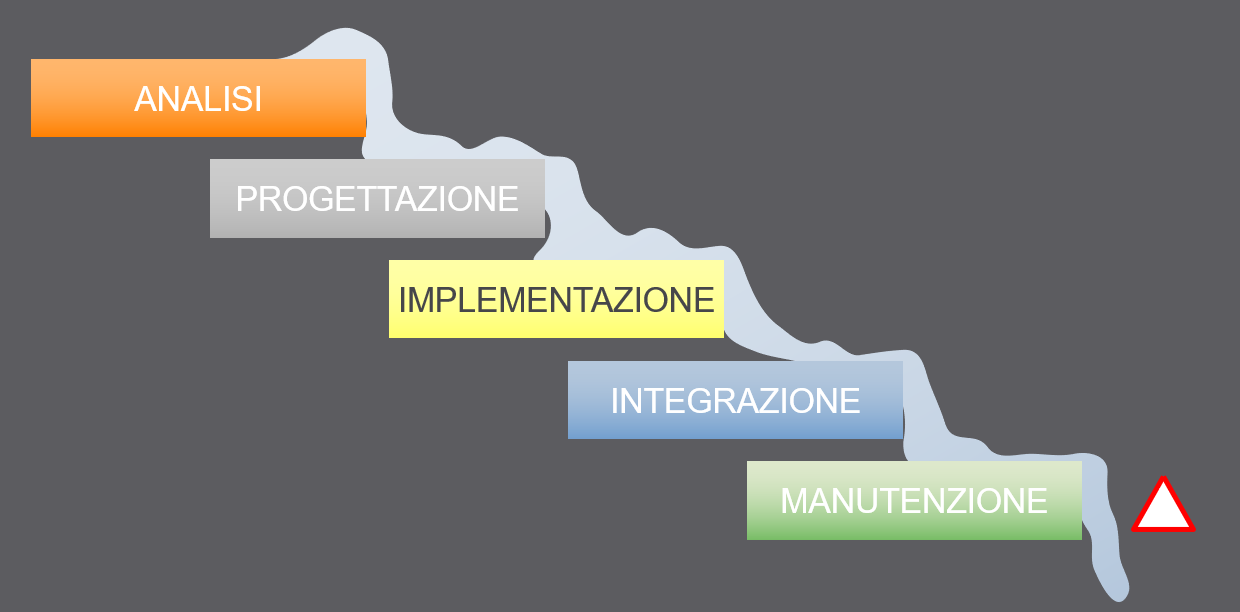
Nella pianificazione di questo progetto ho utilizzato il modello waterfall, cioè una metodologia di gestione sequenziale dei progetti, suddivisi in fasi distinte al fine di agevolare un'ordinata progressione delle lavorazioni

Figura 2 modello waterfall

Come si può vedere nell’immagine sotto, che raffigura il gantt per questo progetto, ci sono le seguenti macro-categorie:

1. **Analisi**: All’interno di questa categoria ho compresso tutte la Attività riguardanti l’analisi del progetto in sé e la sua pianificazione. Teoricamente dovrebbe durare all’incirca 1.5 lezioni che hanno la durata complessiva di 12 ore, ovvero il 17% del progetto.
2. **Progettazione**: Contenuto all’interno di questa categoria ci sono il design delle classi e delle interfacce. Dovrebbe durare teoricamente 0.96 lezioni che corrispondono a 7.68 ore.
3. **Implementazione**: Diviso in altre due sottocategorie contiene tutta la parte pratica del progetto, sino alla scrittura del codice per l’applicativo WEB.

Stile orizzontale

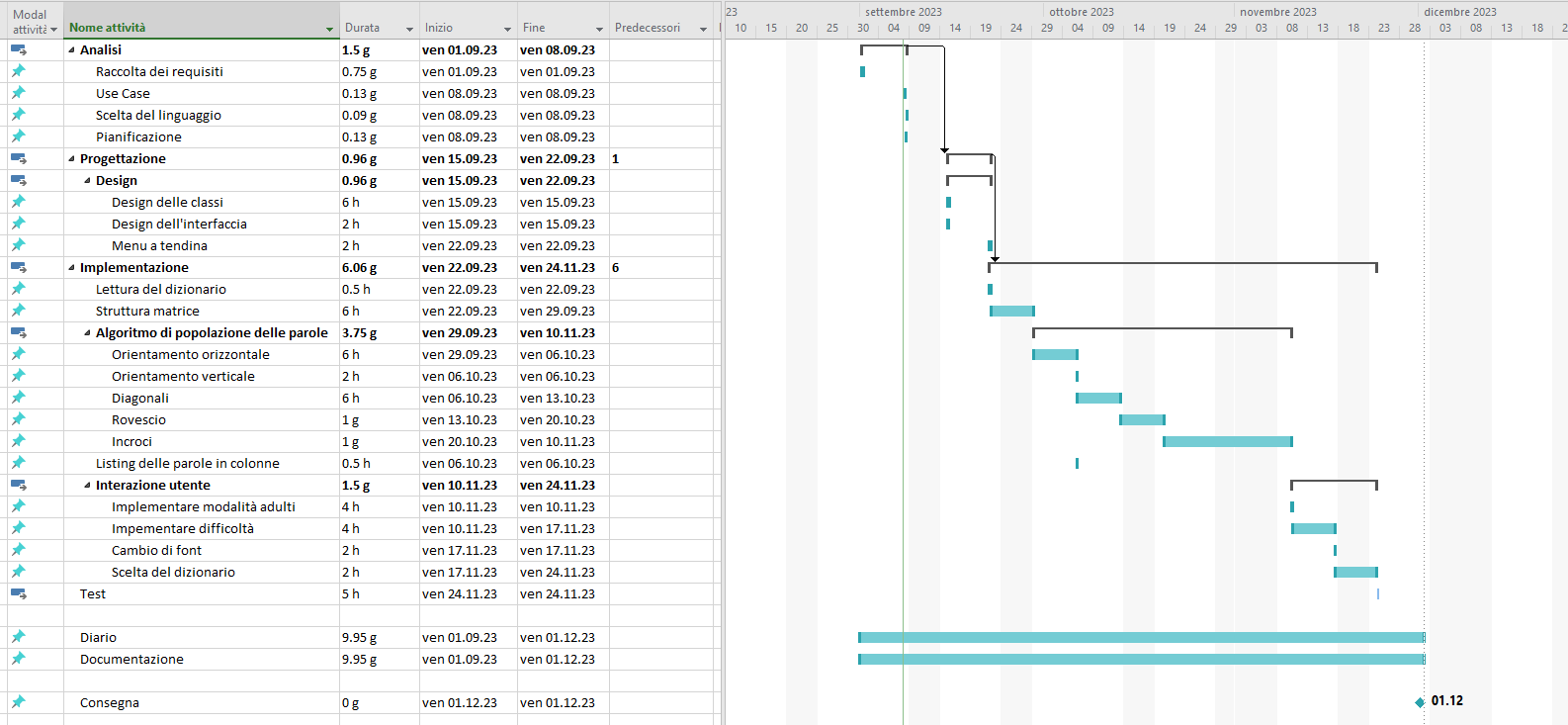


Figura 3 Esempio di diagramma di Gantt

## Analisi dei mezzi

### Software

* Visual Studio Code 1.83
* HTML 5
* CSS 3
* Javascript 1.8.5

### Hardware

Il computer fornito dalla scuola.

# Progettazione

## Design dell’architettura del sistema

### Struttura dell’applicativo web

La struttura delle cartelle è la seguente:

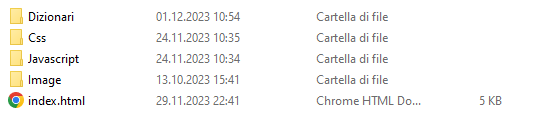


Figura 4 struttura cartelle

* Css: Cartella contenente tutti i file css (file per la veste grafica dell’applicativo web).
* Image: Contiene tutte le immagini che si visualizzeranno nell’applicativo.
* Javascript: Cartella che contiene tutti gli script javascript.
* Dizionari: Cartella contenente i dizionari di base e di default.

## Design delle interfacce

Il design dell’interfaccia è formato da un titolo principale ad esempio “PAROLE CROCIATE”, da un menu a comparsa con le opzioni di creazione della tabella. Mentre la maggior parte della pagina sarà occupata dalla tabella e la lista di parole, tutto ciò centrato al centro.

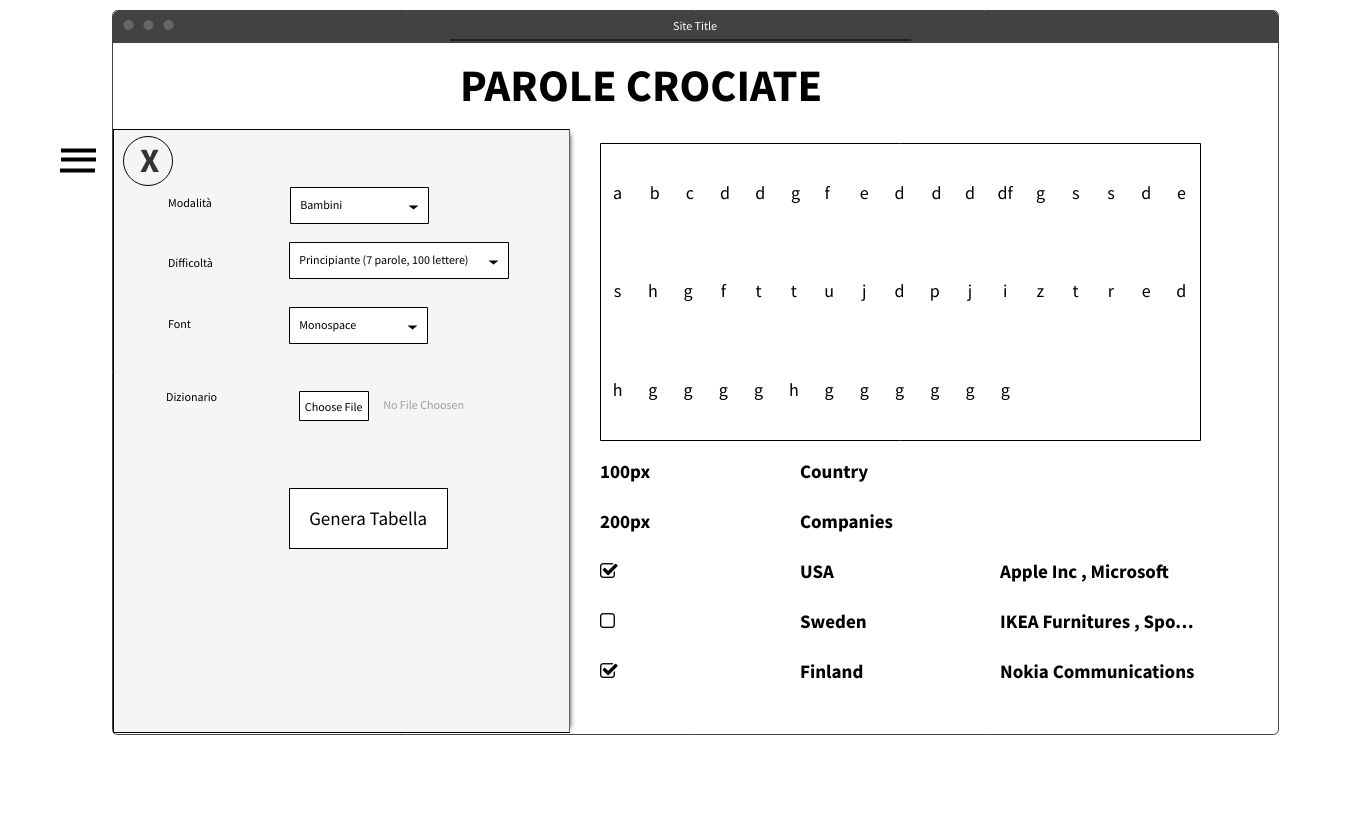


Figura 5 design delle interfaccie

## Design procedurale

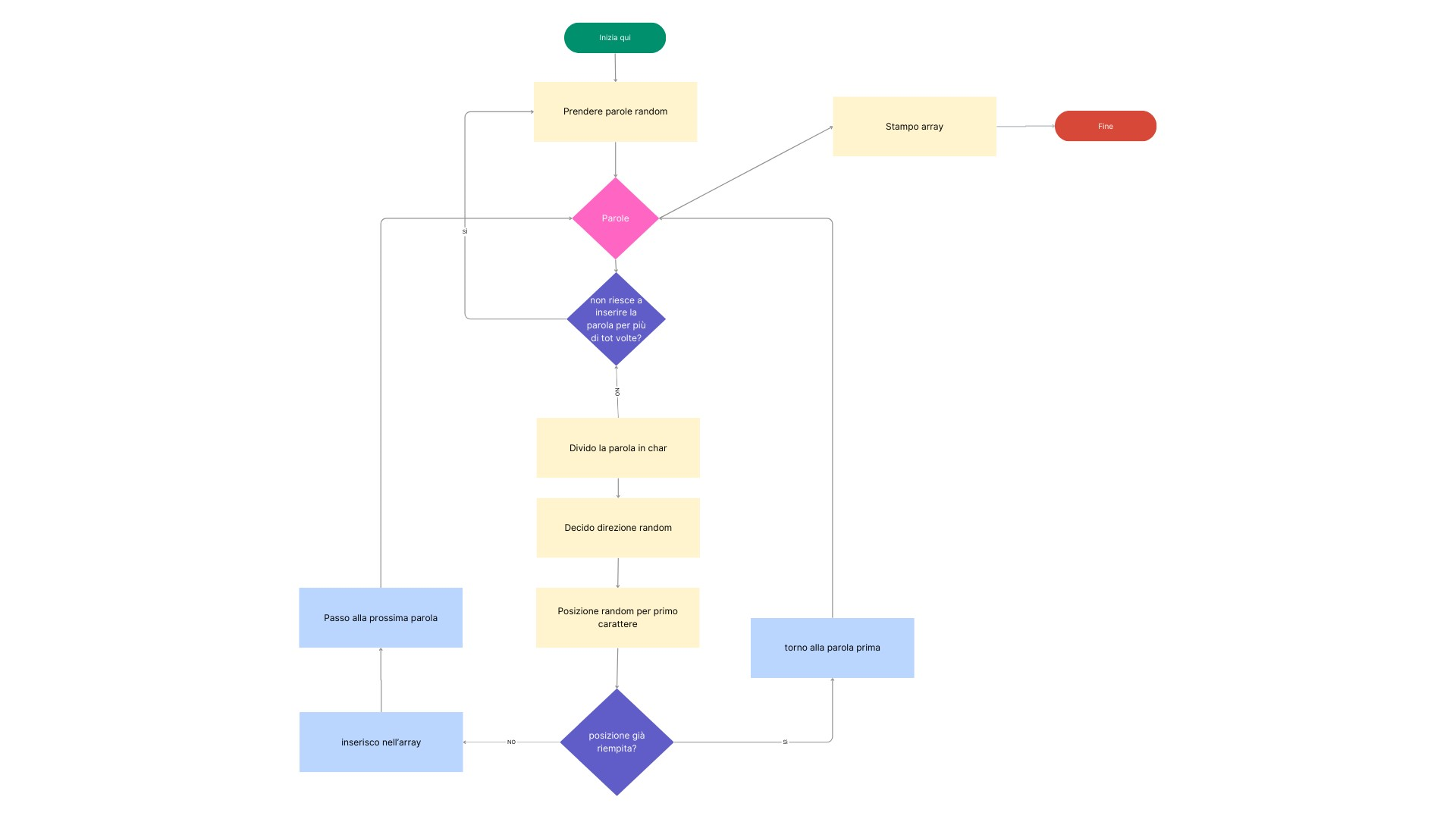


Figura 6 Diagramma di flusso - inserimento parole nella tabella

# Implementazione

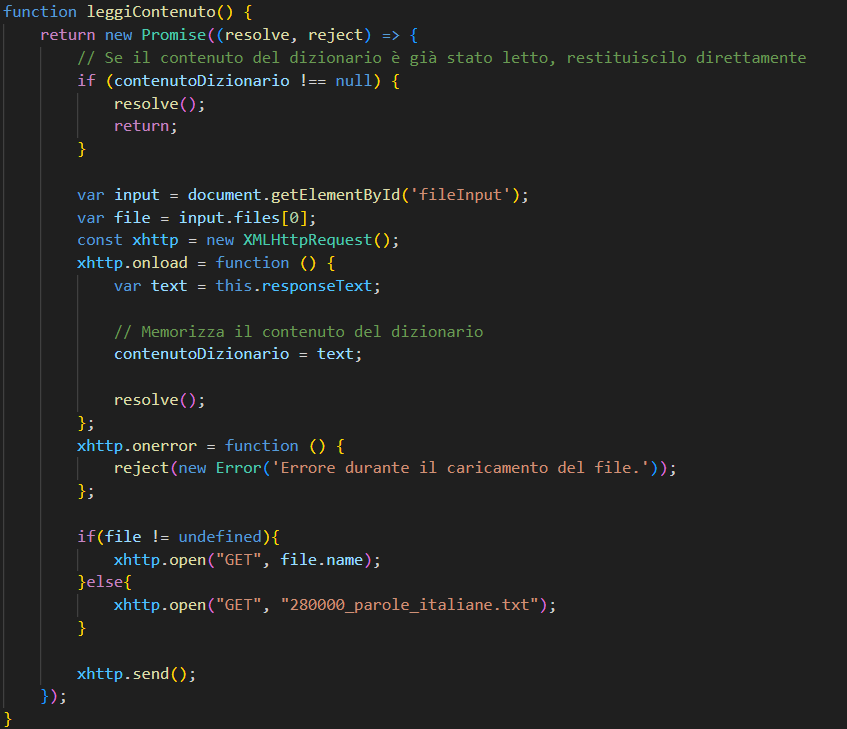
## Lettura del file

Per la lettura del dizionario ho avuto parecchi problemi non sapendo cos’era e come funzionava la promise.

Tuttavia, dopo ricerche e approfondimenti sono riuscito a utilizzarlo per il mio scopo.

### LeggiContenuto()

In questo metodo utilizzo una promise, cioè una classe che mi permette di eseguire le operazioni in modo asicrono(la risoluzione del codice non in sequenza), questo mi serve per prendere il file dal file system.

Figura 7 leggiConenuto()

All’interno del promise apro una richiesta XML per poter accedere al file.

Figura 8 richiesta xml

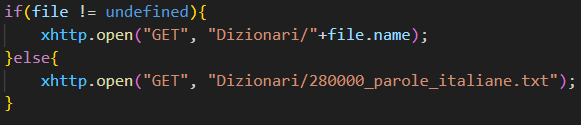
Apro il file che mi passa l’utente oppure quello di default. 

Figura 9 gestisciModalità()

Dopodiché, invio la richiesta.



Al caricamento della richiesta prendo il contenuto del file e lo salvo in una variabile globale

### Async leggo()

In questa funzione **async** utilizzo un **await** che serve per prendere il risultato della promise, infatti await aspetta la sua risposta senza far andare avanti il codice. Utilizzando await devo per forza mettere la funzione async, che significa che il metodo ritornerà sempre una promise.

Figura 10 leggo()

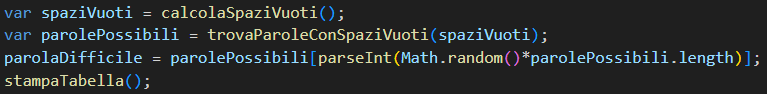
Successivamente se ho 1 sola parola da prendere, vuole dire che ho bisogno della parola difficile della modalità adulti, perciò ciclo il dizionario e raccolgo in un array le parole del numero di caratteri uguale agli spazi vuoti che necessito, dopodiché prendo una di queste casualmente. 

Figura 11 presa della parola difficile

Se invece, devo prendere le parole da mettere nella tabella, se la modalità è “adulti” prendo una parola alla volta finché gli spazi vuoti non sono troppi.

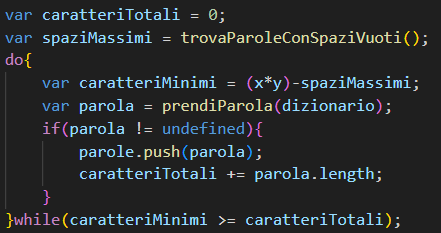


Figura 12 presa delle parole – adulti

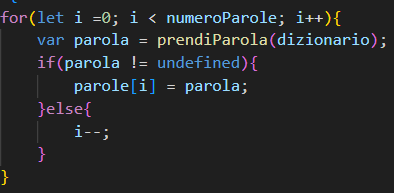
Altrimenti, prendo il numero di parole predefinito. 

Figura 13 presa delle parole - bambini

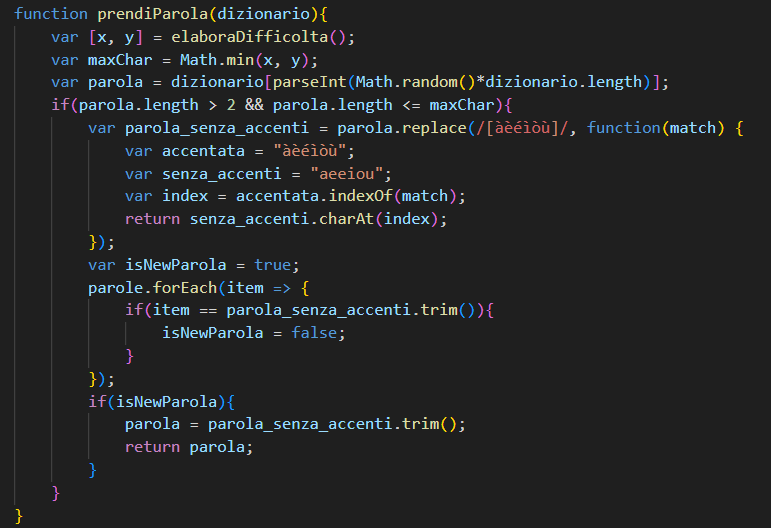
### TrovaParoleConSpaziVuoti()

Questa funzione prende dal dizionario una parola casuale, la quale abbia il numero di lettere uguale agli spazi vuoti. Inoltre se si chiama senza gli spazi vuoti come parametro ritorna la lunghezza massima che equivale alla parola più lunga del dizionario.

Figura 14 trovaParoleConSpaziVuoti()

### PrendiParola()

Questo metodo serve a prendere la parola dal dizionario e facendo i controlli che vada bene.

Figura 15 prendiParola()

Inserisco infine le parole in un div invisibile per poi riprenderle comodamente.

## Popolare la tabella

### PopolaTabella()

In questo metodo prendo le parole selezionate e le inserisco nell’array bidimensionale “contenuto”. Queste parole ovviamente devo inserirle in modo casuale: con una posizione casuale e una direzione casuale. Esse non possono sovrapporsi, ma possono incrociarsi.

Per fare ciò, per ogni parola selezionavo una direzione casuale con un random, dopodiché definisco una posizione x e una posizione y da cui iniziare a inserire la parola e infine controllo se si può inserire, se sì la inserisce, altrimenti ricalcola posizione e direzione.

Immagine che contiene testo, schermata, software, Carattere

Descrizione generata automaticamente

Figura 16 inserimento lettere in contenuto

Infine faccio ancora un controllo sugli spazi vuoti, dato che ora sono effettivi al contrario di quelli al momento della scelta delle parole (vedere 5.1.2). Immagine che contiene testo, schermata, Carattere, linea

Descrizione generata automaticamente

Figura 17 doppio controllo spazi vuoti

### ControlloLettere()

Il metodo controlloLettere() verifica che tutte le lettere della parola si possano inserire nell’array nella posizione e direzione scelta in precedenza.

Più precisamente, se c’è una già una lettera diversa dove dovrebbe essere inserita ritorna false, altrimenti true.

Immagine che contiene testo, schermata, schermo, software

Descrizione generata automaticamenteFigura 18 controlloLettere()

### InserisciLettere()

Questa funzione inserisce le lettere della parola nell’array. Immagine che contiene testo, schermata, schermo, software

Descrizione generata automaticamente

Figura 19 inserisciLettere()

### Evitare ciclo infinito

In questa porzione di codice ho risolto un grosso problema, che consisteva nel generare ogni tanto un ciclo infinito, dovuto al non riuscire a inserire una parola nella tabella.

Con questo metodo tengo in memoria una variabile dei tentativi che fa una parola per entrare nella tabella, dopo 2000 rigenero le parole e riprovo.

Immagine che contiene testo, schermata, Carattere, software

Descrizione generata automaticamenteFigura 20 evitare ciclo infinito

## Stampare tabella

Per “stampare” la tabella ho semplicemente dovuto inserire in ogni td il valore corrispondente dell’array.

### GeneraTabella()

In questo metodo genero la tabella html, setto[[1]](#footnote-1) lo style della tabella.. 

Figura 21 creazione tabella html

### StampaTabella()

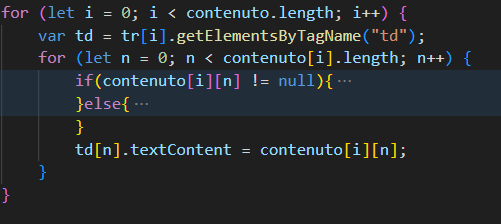
In questa funzione riempio la tabella html con l’array contenuto (vedi 5.2.1). 

Figura 22 inserimento contenuto in tabella

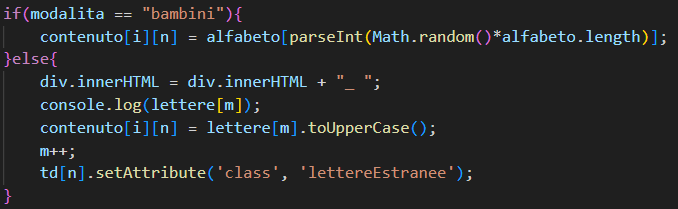
Inoltre, negli spazi vuoti inserisco lettere a caso se la modalità è bambini oppure la parola difficile se la modalità è adulti. 

Figura 23 inserimento lettere negli spazi vuoti

## Generare la soluzione

Per generare la soluzione ho assegnato una classe diversa ad ogni lettera di diverse parole. Questo l’ho fatto nel metodo InserisciLettere().

Figura 24 assegno classe ad array

Dopodiché nello StampaTabella() ho assegnato la classe al td.

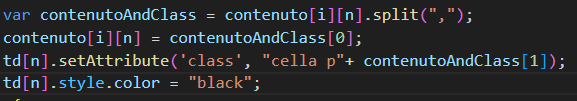


Figura 25 assegno classe alla tabella

### GeneraSoluzione()

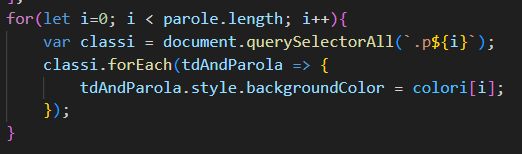
Infine in questo metodo assegno un colore diverso per ogni classe (parola) differente. 

Figura 26 soluzione - colorazione parole

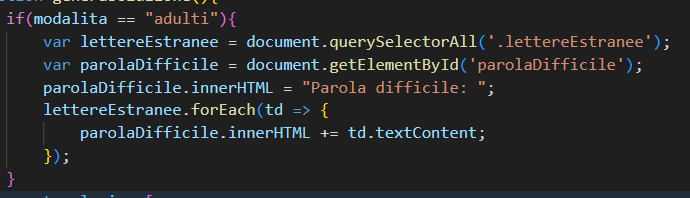
Se la modalità è “adulti” mostro anche la parola difficile. 

Figura 27 soluzione - modalità adulti

## Font

Per far scegliere il font all’utente ho creato un array con tutti i font di Windows[[2]](#footnote-2) e dopodiché gli ho inseriti in un select. 

Figura 28 font in select

### SetFont()

In questo metodo setto il font selezionato dall’utente e lo assegno alla tabella e alla lista. 

Figura 29 setFont()

## Export

Per fare l’export in formato txt e html ho usato lo stesso metodo, cioè quello di creare una stringa con il contenuto, dopodiché trasformare la stringa in byte. E infine grazie al metodo SaveFile() fa partire il download del file.

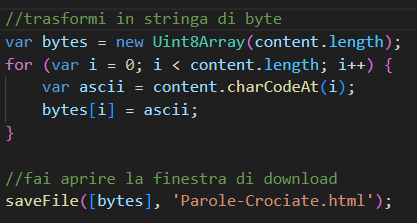


Figura 30 export - string to byte

### ExportTxt()

In questo metodo esporto un file txt con la tabella generata e la lista di parole. Prendo il contenuto della tabella e lo divido in lettere, dopodiché stampo le lettere in modo di formattarle al meglio.

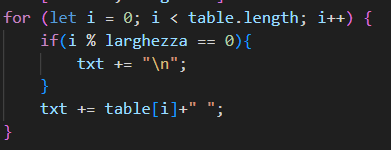


Figura 31 exportTxt() - contenuto tabella

### ExportHtml()

In questo metodo esporto un file html con la tabella generata e la lista di parole.

Creo una stringa con il codice html necessario a creare una pagina, dopodiché aggiungo il css e il contenuto della tabella.

### SaveFile()

Questa funzione crea un link nascosto in html, il metodo restituisce un'altra funzione che accetta due parametri: data (i dati del file) e name (il nome del file).

All'interno della funzione restituita, viene creato un oggetto Blob che rappresenta la sequenza di dati (data) del tipo specificato come parametro nel costruttore di Blob . Nel mio caso, viene impostato il tipo “octet/stream”, che generalmente è utilizzato per contenuti non specificati.

Successivamente, viene creato un URL per l'oggetto Blob utilizzando window.URL.createObjectURL(blob). Questo URL rappresenta un riferimento ai dati del file creato nel browser.

Dopodiché, si aggiunge al link creato inizialmente l’URL.

Infine, viene simulato un clic sull'elemento <a> con il metodo click().

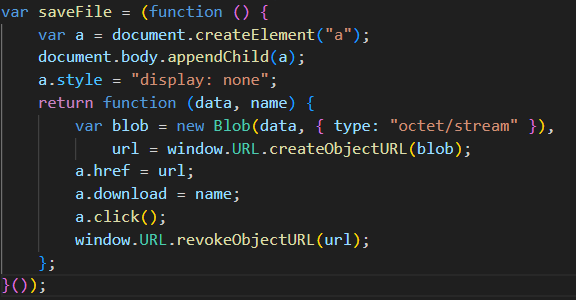


Figura 32 saveFile()

# Test

## Protocollo di test

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case**  **Riferimento** | TC-001  REQ-004 | **Nome** | Generazione tabella default |
| **Descrizione** | Si prova a generare la tabella con gli input di default. | | |
| **Prerequisiti** | Tabella completata (aver fatto tutte le direzioni, con incroci). | | |
| **Procedura** | 1. Cliccare il bottone “Genera tabella” | | |
| **Risultati attesi** | Dovrebbe generarsi la tabella con le parole in base difficoltà, in tutte le direzioni, senza sovrapposizione e con possibili incroci. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case**  **Riferimento** | TC-002  REQ-002 | **Nome** | Modalità adulto |
| **Descrizione** | Si verifica che la modalità adulto funzioni e che quindi crei una parola con le lettere rimanenti. | | |
| **Prerequisiti** | Tabella completata | | |
| **Procedura** | 1. Selezionare la modalità adulti tramite input 2. Cliccare “genera tabella” 3. Trovare tutte le parole e unire le lettere rimanenti | | |
| **Risultati attesi** | Con le lettere rimanenti dovrebbe crearsi una parola. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case**  **Riferimento** | TC-003  REQ-003 | **Nome** | Difficoltà |
| **Descrizione** | Si verifica che le tutte le difficoltà funzionino correttamente, parole e grandezza corrette. | | |
| **Prerequisiti** | Tabella completata | | |
| **Procedura** | 1. Selezionare la difficoltà tramite input 2. Cliccare “genera tabella” 3. Rifare i passaggi per tutte le difficoltà | | |
| **Risultati attesi** | Per ogni difficoltà devono esserci il numero di parole e la grandezza corretta (definita nell’input). | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case**  **Riferimento** | TC-004  REQ-005 | **Nome** | Export tabella |
| **Descrizione** | Si verifica che l’export in formato txt e html funzioni correttamente | | |
| **Prerequisiti** | Tabella completata e export completato | | |
| **Procedura** | 1. Cliccare export txt 2. Aprire file (da verificare) 3. Verificare che contenuto sia corretto 4. Cliccare export html 5. Aprire file (da verificare) 6. Verificare che contenuto sia corretto | | |
| **Risultati attesi** | Nei file ci dovrà essere la tabella riempita con la lista delle parole. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case**  **Riferimento** | TC-005  REQ-006 | **Nome** | Scelta del font |
| **Descrizione** | Si verifica che si possa scegliere il font della tabella | | |
| **Prerequisiti** | Tabella completata e font completato | | |
| **Procedura** | 1. Cliccare font: 2. Scegliere un font 3. Generare tabella | | |
| **Risultati attesi** | Il font della tabella e della lista dovrà essere quello selezionato | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case**  **Riferimento** | TC-006  REQ-007 | **Nome** | Scelta del dizionario |
| **Descrizione** | Si verifica che si possa scegliere il dizionario della tabella | | |
| **Prerequisiti** | Tabella completata e dizionario completato | | |
| **Procedura** | 1. Ricaricare la pagina 2. Cliccare dizionario: 3. Scegliere un file txt con delle parole 4. Generare tabella | | |
| **Risultati attesi** | Le parole nella tabella devono essere quelle nel dizionario | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case**  **Riferimento** | TC-007  REQ-008 | **Nome** | Mostrare la soluzione |
| **Descrizione** | Si verifica che si possa visualizzare la soluzione della tabella, con le parole evidenziate e la possibile parola nascosta | | |
| **Prerequisiti** | Tabella completata e soluzione completata | | |
| **Procedura** | 1. Cliccare “genera tabella” 2. Dopodiché cliccare “genera soluzione” | | |
| **Risultati attesi** | Le parole nella tabella devono essere evidenziate in modo che si intravvedono una dall’altra e la parola nascosta se c’è deve essere mostrata. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case**  **Riferimento** | TC-008  REQ-004\_1 | **Nome** | Controllo accenti |
| **Descrizione** | Si verifica che non ci possano essere degli accenti nelle parole della tabella. | | |
| **Prerequisiti** | Tabella completata e dizionario completato. | | |
| **Procedura** | 1. Selezionare un dizionario con delle parole con accenti. 2. Cliccare “genera tabella” | | |
| **Risultati attesi** | Le parole nella tabella non devono avere accenti. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case**  **Riferimento** | TC-009  REQ-007 | **Nome** | Controllo dizionario corretto |
| **Descrizione** | Si verifica che il dizionario abbia una lunghezza adatta | | |
| **Prerequisiti** | Tabella completata e dizionario completato. | | |
| **Procedura** | 1. Selezionare un dizionario con poche parole. 2. Cliccare “genera tabella” | | |
| **Risultati attesi** | Deve esserci un avviso che il dizionario inserito non è abbastanza lungo | | |

## 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.

|  |  |  |
| --- | --- | --- |
| **Test Case** | **Risultato ottenuto** | **Stato** |
| TC-001 |  | Passato |
| TC-002 |  | Passato |
| TC-003 | Modalità adulto e difficoltà leggenda non funziona | Fallito |
| TC-004 |  | Passato |
| TC-005 |  | Passato |
| TC-006 | La tabella conteneva le parole del dizionario selezionato | Passato |
| TC-007 |  | Passato |
| TC-008 |  | Passato |
| TC-009 |  | Passato |

## Mancanze/limitazioni conosciute

Il loader non funziona correttamente, per colpa della richiesta async della promise. L’elaborazione della tabella e la visualizzazione del loader sono eseguite contemporaneamente, non facendo vedere il loader.

Per mancanza di priorità e tempo non sono riuscito a risolvere il problema.

# Consuntivo

Rispetto a quello che mi aspettavo ci ho messo molto meno a implementare la generazione della tabella con le parole inserite in ogni direzione e invece, ci ho messo molto più tempo a elaborare il dizionario e prendere le parole dal file.

Stile orizzontale

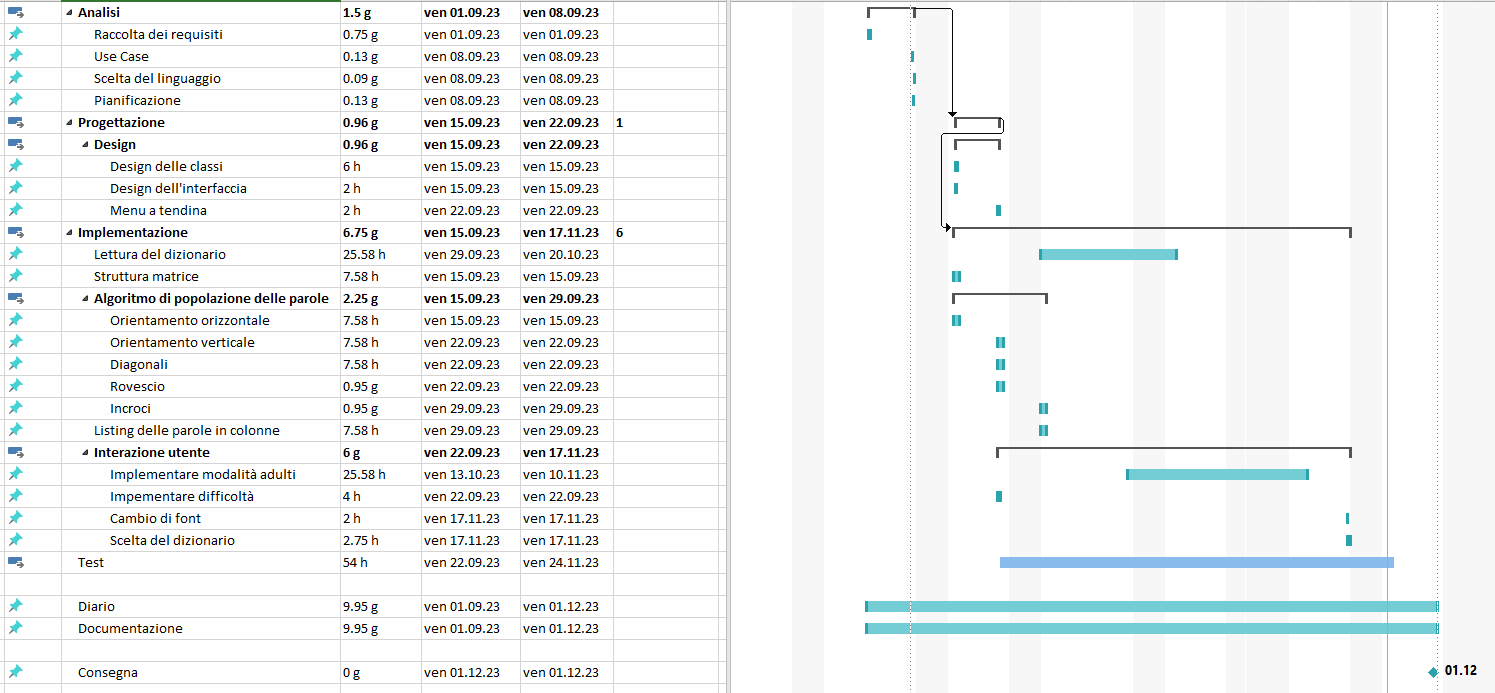


Figura 33 Esempio di diagramma di Gantt consuntivo

# Conclusioni

## Sviluppi futuri

Si potrebbe implementare un algoritmo più veloce ed efficiente nella creazione della tabella.

Si potrebbe ampliare l’opzione per poter giocare direttamente dal sito.

## Considerazioni personali

In conclusione posso dire che questo progetto mi ha permesso di migliorare nella programmazione, scoprendo e applicando aspetti nuovi che non avevo mai visto nella mia formazione.

Inoltre, mi ha permesso di ampliare la mia esperienza nella gestione di un progetto di queste dimensioni, essendo il mio primo progetto effettivo.

Sono contento delle tempistiche, dal momento che non sono mai stato indietro e ho avuto tempo per problemi e inconvenienze/dimenticanze.

# Bibliografia

## Sitografia

* <https://www.w3schools.com> , 25.10.2023.
* <https://stackoverflow.com>, 17.11.2023
* <https://aulab.it/>, promise javascript, 24.11.2023
* <https://forum.html.it/> , 10.11.2023
* <https://www.canva.com/> , 20.10.2023

# Glossario

|  |  |
| --- | --- |
| **Termine** | **Significato** |
| Applicativo WEB | Si intende il sito web che verrà visualizzato a progetto completato. |
| Gantt | Diagramma per pianificare un intero progetto e dividerlo in varie attività. Grazie ad esso si può avere un lavoro più organizzato. |
| Hardware | Parte fisica di un sistema. |
| Software | l'insieme dei programmi che possono essere impiegati su un sistema di elaborazione |
| HTML | Hyper Text Markup Language è un linguaggio che permette di creare siti web |
| CSS | Cascading Style Sheets è un linguaggio usato per programmare la resa grafica di un applicativo web |
| Javascript | linguaggio di programmazione che gli sviluppatori utilizzano per realizzare pagine Web interattive. |
| Visual Studio Code | Software che permette di scrivere codice in vari linguaggi (IDE) |
| Blob | Binary Large Object è una tipologia di dati che funziona solo con informazioni binarie. |

# Indice delle figure

[Figura 1 UseCase 8](#_Toc152336000)

[Figura 2 modello waterfall 9](#_Toc152336001)

[Figura 3 Esempio di diagramma di Gantt 10](#_Toc152336002)

[Figura 4 struttura cartelle 11](#_Toc152336003)

[Figura 5 design delle interfaccie 12](#_Toc152336004)

[Figura 6 Diagramma di flusso - inserimento parole nella tabella 13](#_Toc152336005)

[Figura 7 leggiConenuto() 15](#_Toc152336006)

[Figura 8 richiesta xml 15](#_Toc152336007)

[Figura 9 gestisciModalità() 16](#_Toc152336008)

[Figura 10 leggo() 17](#_Toc152336009)

[Figura 11 presa della parola difficile 18](#_Toc152336010)

[Figura 12 presa delle parole – adulti 18](#_Toc152336011)

[Figura 13 presa delle parole - bambini 19](#_Toc152336012)

[Figura 14 trovaParoleConSpaziVuoti() 20](#_Toc152336013)

[Figura 15 prendiParola() 21](#_Toc152336014)

[Figura 16 inserimento lettere in contenuto 22](#_Toc152336015)

[Figura 17 doppio controllo spazi vuoti 22](#_Toc152336016)

[Figura 18 controlloLettere() 23](#_Toc152336017)

[Figura 19 inserisciLettere() 24](#_Toc152336018)

[Figura 20 evitare ciclo infinito 25](#_Toc152336019)

[Figura 21 creazione tabella html 26](#_Toc152336020)

[Figura 22 inserimento contenuto in tabella 27](#_Toc152336021)

[Figura 23 inserimento lettere negli spazi vuoti 27](#_Toc152336022)

[Figura 24 assegno classe ad array 27](#_Toc152336023)

[Figura 25 assegno classe alla tabella 28](#_Toc152336024)

[Figura 26 soluzione - colorazione parole 28](#_Toc152336025)

[Figura 27 soluzione - modalità adulti 28](#_Toc152336026)

[Figura 28 font in select 29](#_Toc152336027)

[Figura 29 setFont() 29](#_Toc152336028)

[Figura 30 export - string to byte 30](#_Toc152336029)

[Figura 31 exportTxt() - contenuto tabella 30](file:///E:\2023-24\Progetti\Parole-crociate\3_Documentazione\Documentazione%20parole%20crociate.docx#_Toc152336030)

[Figura 32 saveFile() 31](#_Toc152336031)

[Figura 33 Esempio di diagramma di Gantt consuntivo 41](#_Toc152336032)

# Allegati

* QdC
* Abstract

1. **Settare:** usato nel gergo, significa assegnare un valore, è preso dal verbo inglese ‘set’. [↑](#footnote-ref-1)
2. **Windows:** in informatica,è unsistema operativo. [↑](#footnote-ref-2)