

Generatore trova parole

Titolo del progetto	Generatore trova parole
Alunno/a	Davide Branchi
Classe	Info 3 AA
Anno scolastico	2023/2024
Docente responsabile	Michel Palucci

1 Indice

1	Indice.....	2
2	Introduzione.....	3
2.1	Informazioni sul progetto.....	3
2.2	Abstract.....	3
2.3	Scopo.....	3
3	Analisi.....	4
3.1	Analisi del dominio.....	4
3.2	Analisi e specifica dei requisiti.....	4
3.3	Use case.....	8
3.4	Pianificazione.....	9
3.5	Analisi dei mezzi.....	10
3.5.1	Software.....	10
3.5.2	Hardware.....	10
4	Progettazione.....	11
4.1	Design delle interfacce.....	11
4.2	Design procedurale.....	13
5	Implementazione.....	14
5.1	Inserimento delle parole.....	14
5.2	Interfaccia per la modifica del dizionario.....	15
5.3	Generazione della griglia.....	16
5.4	Generazione automatica della griglia.....	21
5.5	Esportazione della pagina.....	22
5.6	Modifica del dizionario (client).....	23
5.7	Modifica del dizionario (server).....	25
5.8	Creazione test generazione griglia.....	26
6	Test.....	27
6.1	Protocollo di test.....	27
6.2	Risultati test.....	34
6.2.1	Tabella riassuntiva test su Google Chrome, Firefox e Microsoft Edge.....	34
6.2.2	Tabella riassuntiva altri test.....	35
6.3	Mancanze/limitazioni conosciute.....	35
7	Consuntivo.....	36
8	Conclusioni.....	37
8.1	Sviluppi futuri.....	37
8.2	Considerazioni personali.....	37
9	Bibliografia.....	38
9.1	Sitografia.....	38
10	Glossario.....	39
11	Indice delle figure.....	39
12	Allegati.....	40

2 Introduzione

2.1 Informazioni sul progetto

- Allievo: Davide Branchi
- Classe: I3AA
- Scuola: Scuola Arti e Mestieri Trevano
- Docente responsabile: Michel Palucci
- Data inizio: 01.09.2023
- Data consegna: 01.12.2023

2.2 Abstract

Before the creation of this project, the employees of the puzzle company created the word puzzles manually. Thanks to the following application (Generatore Trova Parole) employees will be able to generate word puzzles in a much simpler, faster and more automated way. This project will allow the puzzle company to optimize the production process by increasing the simplicity and speed of work

2.3 Scopo

Lo scopo del progetto è quello di facilitare la creazione del gioco del trova parole per un'azienda di enigmistica, tramite un applicativo web.

L'applicativo permetterà di generare una pagina A4 contenente un trova parole e successivamente di esportarla.

I dipendenti dell'azienda generano la griglia del trova parole inserendo manualmente delle parole, oppure, in caso di mancanza di idee sulle parole da inserire, possono utilizzare la funzione di generazione automatica che permette loro di generare la griglia con parole casuali.

La pagina generata verrà poi esportata con un'alta qualità e potrà poi essere elaborata dai grafici dell'azienda e pubblicata sulla rivista di enigmistica. Inoltre l'applicativo faciliterà anche ai dipendenti la scelta della parola finale da utilizzare, grazie alla presenza di un dizionario di parole italiane integrato.

3 Analisi

3.1 Analisi del dominio

L'applicativo verrà utilizzato dai dipendenti dell'azienda di enigmistica per poter generare il gioco del trova parole e per poi esportarlo e pubblicarlo.

Questo applicativo semplifica ai dipendenti la creazione del trova parole, in quanto non bisogna più realizzarlo manualmente ma è l'applicativo che lo genera, i dipendenti devono solamente inserire le parole da utilizzare.

Per l'utilizzo non sono richieste particolari conoscenze tecniche da parte dei dipendenti, in quanto l'applicativo è semplice e intuitivo.

3.2 Analisi e specifica dei requisiti

In base alle interviste effettuate con il cliente sono stati definiti i seguenti requisiti in ordine di priorità

Req-01	
Nome	Inserimento delle parole
Priorità	1
Versione	1.0
Note	L'utente deve poter inserire delle parole per permettere la generazione del trova parole

Req-02	
Nome	Parola finale scelta dall'utente
Priorità	1
Versione	1.0
Note	L'utente può scegliere la parola finale in base al numero di caratteri rimasti

Req-03	
Nome	Parola finale proposta dall'applicativo
Priorità	1
Versione	1.0
Note	L'applicativo propone una lista di parole finali che l'utente può decidere di utilizzare

Req-04

Nome	Generazione della pagina
Priorità	1
Versione	1.0
Note	L'applicativo dovrà generare una pagina in formato A4 contenente il trova parole

Req-05

Nome	Grandezza della griglia proporzionale
Priorità	1
Versione	1.0
Note	La griglia generata dall'applicativo dovrà essere proporzionale alla grandezza dell'intera pagina

Req-06

Nome	Posizione delle parole
Priorità	1
Versione	1.0
Note	Le parole nella griglia dovranno essere posizionate in orizzontale, verticale e diagonale

Req-07

Nome	Generazione della griglia con i dati dell'utente
Priorità	1
Versione	1.0
Note	L'applicativo dovrà permettere di generare la griglia in base alle parole inserite dall'utente

Req-08	
Nome	Generazione automatica della griglia
Priorità	1
Versione	1.0
Note	L'applicativo dovrà permettere all'utente di far generare la griglia del trova parola in maniera completamente automatica, quindi senza il bisogno dell'inserimento delle parole da parte sua

Req-09	
Nome	Dimensione della griglia
Priorità	1
Versione	1.0
Note	La griglia dovrà essere larga 15 e alta 10 caratteri. Si dovrà inserire le parole per un minimo di 140 e un massimo di 147 caratteri. Una parola potrà essere lunga massimo 15 caratteri

Req-10	
Nome	Dizionario delle parole
Priorità	1
Versione	1.0
Note	Dovrà esserci un dizionario di parole che l'applicativo potrà utilizzare

Req-11	
Nome	Esportazione della pagina
Priorità	1
Versione	1.0
Note	La pagina deve poter essere esportata in formato PNG e altri formati

Req-12	
Nome	Immagine di alta qualità
Priorità	1
Versione	1.0
Note	L'immagine generata del trova parole dovrà essere di alta qualità

Req-13	
Nome	Applicativo multiplatforma
Priorità	1
Versione	1.0
Note	L'applicativo dovrà poter essere utilizzato da qualsiasi browser, quindi dovrà anche essere responsive per garantire un'esperienza ottimale su ogni dispositivo

Req-14	
Nome	Modifica del dizionario da parte dell'utente
Priorità	3
Versione	1.0
Note	L'utente dovrà poter accedere ad un'interfaccia che permette di modificare il dizionario di default dell'applicativo, ad esempio aggiungendo, togliendo o inserendo nuove parole

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.

3.3 Use case

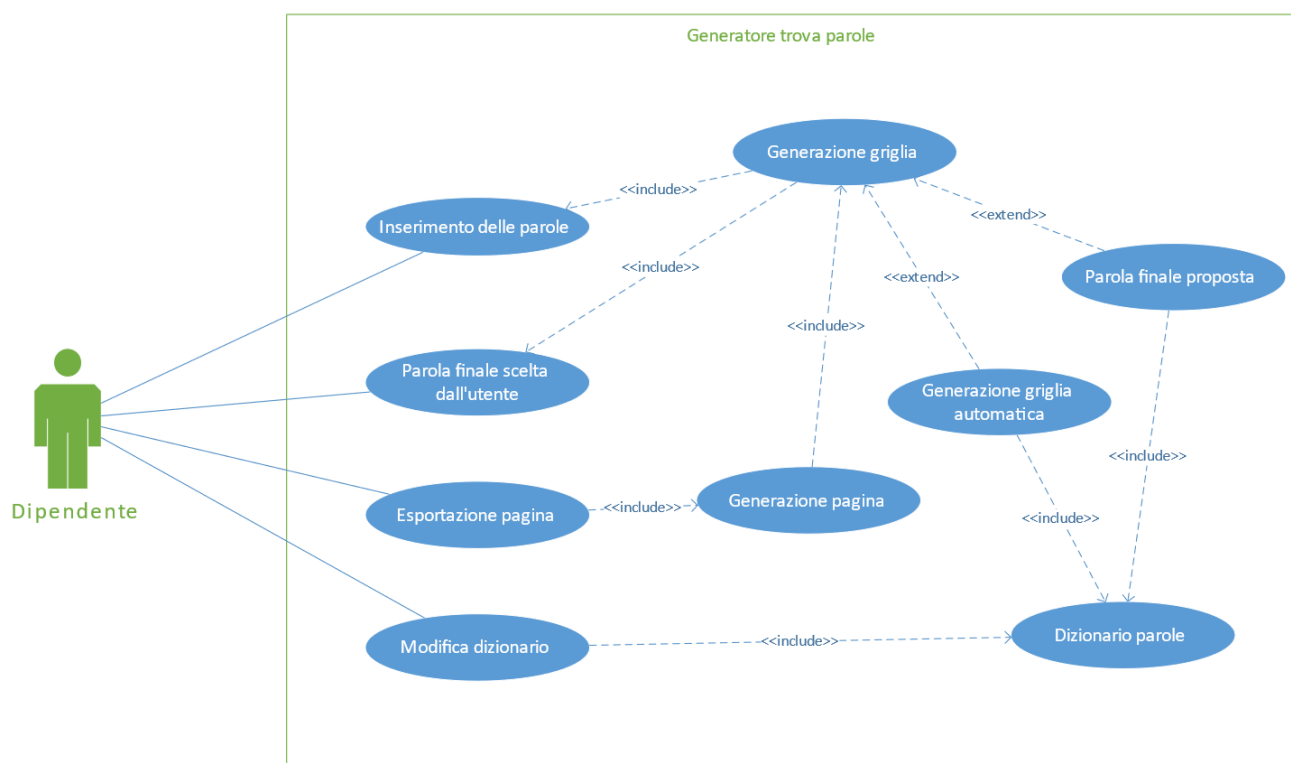


Figura 1 Use case dell'applicativo

L'applicativo avrà un solo attore, che è il dipendente dell'azienda che utilizza il programma.

Le attività che il dipendente potrà direttamente fare sono: l'inserimento delle parole, la scelta della parola finale, l'esportazione della pagina e la modifica del dizionario.

L'inserimento delle parole e la scelta della parola finale includono la generazione della griglia, mentre l'esportazione della pagina include la generazione della pagina e la modifica del dizionario include la presenza del dizionario di parole.

La generazione della griglia automatica include il dizionario di parole ed estende la generazione della griglia.

La parola finale proposta estende la generazione della griglia ed include il dizionario delle parole.

3.4 Pianificazione

Per la pianificazione del progetto ho scelto di utilizzare la metodologia a cascata, tutte le fasi del progetto vengono rappresentate nel diagramma di Gantt sottostante:





























1			▸ Progetto Generatore Trova Parole	77.33 h	ven 01.09.23	ven 01.12.23	
2			▸ Analisi	10 h	ven 01.09.23	ven 15.09.23	
3			Intervista cliente	2 h	ven 01.09.23	ven 01.09.23	
4			QdC	4 h	ven 08.09.23	ven 08.09.23	3
5			Use case	2 h	ven 08.09.23	ven 08.09.23	4
6			Stesura requisiti	2 h	ven 15.09.23	ven 15.09.23	5
7			▸ Progettazione	12 h	ven 15.09.23	ven 29.09.23	
8			Gantt	3 h	ven 15.09.23	ven 15.09.23	2
9			Mockup Interfacce	4 h	ven 15.09.23	ven 22.09.23	8
10			Diagramma flusso	3 h	ven 22.09.23	ven 22.09.23	9
11			Test Case	2 h	ven 29.09.23	ven 29.09.23	10
12			▸ Implementazione	32 h	ven 29.09.23	ven 10.11.23	
13			▸ Creazione GUI	8 h	ven 29.09.23	ven 06.10.23	
14			Main	4 h	ven 29.09.23	ven 29.09.23	7
15			Modifica dizionario	4 h	ven 06.10.23	ven 06.10.23	14
16			▸ Algoritmi	24 h	ven 06.10.23	ven 10.11.23	
17			Generazione griglia	8 h	ven 06.10.23	ven 13.10.23	13
18			Esportazione	8 h	ven 20.10.23	ven 27.10.23	17
19			Modifica dizionario	8 h	ven 27.10.23	ven 10.11.23	18
20			▸ Integrazione	14 h	ven 10.11.23	ven 24.11.23	
21			Test integrazione	4 h	ven 10.11.23	ven 17.11.23	12
22			Bug fixing	8 h	ven 17.11.23	ven 24.11.23	21
23			Test accettazione	2 h	ven 24.11.23	ven 24.11.23	22
24	 		Documentazione	77.33 h	ven 01.09.23	ven 01.12.23	

Figura 2 Diagramma di Gantt del progetto

Nella prima fase del progetto abbiamo l'analisi che comprende l'intervista con il cliente per stabilire i requisiti del progetto e una prima parte di documentazione per la stesura di quest'ultimi.

Dopo l'analisi vi è una fase di progettazione che comprende la creazione del Gantt e i vari diagrammi.

Successivamente nella fase di implementazione verrà realizzato l'applicativo, inizialmente creando le GUI e poi sviluppando i diversi algoritmi.

Nella fase finale abbiamo i test di integrazione che permettono di verificare il buon funzionamento dell'applicativo, il bug fixing per correggere eventuali bug e per finire i test di accettazione del prodotto.

3.5 Analisi dei mezzi

3.5.1 Software

- Visual Studio Code v. 1.78.2
- HTML 5
- CSS 3
- Javascript V8 11.6.189.20
- NodeJS 18.18.0 (LTS)

3.5.2 Hardware

- PC: per sviluppare l'applicativo verrà utilizzato un PC con le seguenti specifiche:
 - CPU: Intel i7-7700
 - RAM: 16 GB
 - SSD: 512 GB
 - GPU: Intel HD Graphics 630

4 Progettazione

4.1 Design delle interfacce

L'applicativo avrà due interfacce principali, la prima dove è presente la generazione della griglia, mentre la seconda che permette di modificare il contenuto del dizionario.

Generatore Trova Parole

Inserire titolo

Griglia 10x10

Inserire parola

Inserire parola finale

Parole finali proposte:

Word 1
Word 2
Word 3
Word 4
Word 5
Word 6
Word 7
Word 8
Word 9
Word 10

Caratteri inseriti: X
Caratteri minimi: 140
Caratteri massimi: 147

Esporta

Modifica dizionario

Genera

Figura 3 Design dell'interfaccia principale

In questa interfaccia inizialmente il bottone “Esporta” sarà disattivato e non saranno presenti la lista di parole finali proposte e il campo per inserire la parola finale. Dopo che verrà generata la griglia tramite il bottone “Genera” sarà possibile selezionare la parola finale o inserirla tramite il campo apposito e esportare la pagina.

Generatore Trova Parole

< Esci

Modifica dizionario

1	Word 1		
2	Word 2		
3	Word 3		
4	Word 4		
5	Word 5		
6	Word 6		
7	Word 7		
8	Word 8		
9	Word 9		
10	Word 10		
11	Word 11		
12	Word 12		
13	Word 13		
14	Word 14		
15	Word 15		
16	Word 16		
17	Word 17		
18	Word 18		
19	Word 19		
20	Word 20		
...	...		

+ Aggiungi parola

Salva modifiche

Figura 4 Design dell'interfaccia "Modifica dizionario"

Nella seconda interfaccia è possibile modificare il contenuto del dizionario che viene visualizzato in una tabella.

Tramite l'icona della penna è possibile modificare la parola, mentre con l'icona "X" è possibile eliminarla.

In fondo alla griglia è presente un pulsante che permette di aggiungere nuove parole al dizionario. Per terminare è necessario salvare le modifiche e successivamente si potrà uscire dalla pagina.

4.2 Design procedurale

Il diagramma di flusso è disponibile nella sezione degli allegati (Activity Diagram - generazione).

In questo diagramma di flusso è rappresentato l'algoritmo che viene avviato alla pressione del pulsante "Genera".

Quando viene premuto il pulsante vengono lette le parole inserite dall'utente, e viene controllato se è stato rispettato il limite dei caratteri, in caso contrario viene visualizzato un errore e l'esecuzione termina.

Se il limite è stato rispettato per ogni parola viene generata una posizione casualmente, successivamente la parola viene posizionata nella matrice.

Quando tutte le parole sono state inserite all'interno della matrice viene stampata la griglia, poi viene visualizzata la lista di parole finali proposte, per proseguire bisogna aspettare l'input dell'utente.

Dopo che l'utente sceglie la parola finale, quest'ultima viene controllata, se non è valida viene visualizzato un errore e l'algoritmo rimane in attesa di un altro input, altrimenti la parola viene inserita nella matrice che poi viene ristampata, a questo punto l'algoritmo raggiunge la sua fine.

5 Implementazione

5.1 Inserimento delle parole

Il primo algoritmo che ho realizzato è quello per l’inserimento delle parole da parte dell’utente, quando l’utente inserisce una parola viene visualizzato un altro input fino al massimo di 30 parole:


parola 1	parola 2	parola 3	parola 4	parola 5
parola 6	parola 7	parola 8	parola 9	parola 10
parola 11	parola 12	parola 13	parola 14	parola 15
parola 16	parola 17	parola 18	parola 19	parola 20
parola 21	parola 22	parola 23	parola 24	parola 25
parola 26				

Figura 5 Esempio interfaccia inserimento parole

Quando l’utente cancella una parola (in questo esempio verrà cancellata la “parola 4”), tutte le altre parole tornano indietro di un posto:


parola 1	parola 2	parola 3	parola 5	parola 6
parola 7	parola 8	parola 9	parola 10	parola 11
parola 12	parola 13	parola 14	parola 15	parola 16
parola 17	parola 18	parola 19	parola 20	parola 21
parola 22	parola 23	parola 24	parola 25	parola 26
				

Figura 6 Esempio rimozione parola dall'interfaccia di inserimento

Questo algoritmo si trova nel file “input-manager.js” alle righe 97-134

5.2 Interfaccia per la modifica del dizionario

Il dizionario delle parole integrato nell'applicativo è in formato xml (dictionary.xml), contiene circa 400 parole italiane. Ogni parola è racchiusa in un tag contenente la lunghezza della stessa, in questo modo è facilitata la selezione delle parole per una certa lunghezza.

```
<DICTIONARY>

  <LENGTH8>aquilone</LENGTH8>
  <LENGTH7>giraffa</LENGTH7>
  <LENGTH5>asino</LENGTH5>
  <LENGTH5>caldo</LENGTH5>
  <LENGTH5>pesca</LENGTH5>
  <LENGTH6>grazie</LENGTH6>
  <LENGTH4>topo</LENGTH4>
  ...
</DICTIONARY>
```

Figura 7 Struttura del dizionario

Al caricamento della pagina viene richiesto il file xml al server, successivamente viene memorizzato il contenuto nella variabile *dictionary* e poi viene caricata la tabella e stampata tramite le funzioni *loadTable()* e *displayTable()*.

Il codice seguente si trova nel file “load-dictionary.js” e viene utilizzato per eseguire quanto descritto nel precedente paragrafo.

```
window.onload = function(){
  let xhttp = new XMLHttpRequest();
  xhttp.open("GET", "get-dictionary"); //Richiesta del dizionario
  xhttp.send();
  xhttp.onreadystatechange = function() {
    if (this.readyState == 4) {
      if(this.status == 200){
        dictionary =
        this.responseXML.querySelectorAll('LENGTH2,LENGTH3,LENGTH4,LENGTH5,LENGTH6,LENGTH7,LENGTH8,LENGTH9,LENGTH10,LENGTH11,LENGTH12,LENGTH13,LENGTH14,LENGTH15');
        loadTable(100); //Carica le prime 100 righe della tabella
        displayTable(); //Mostra la tabella
      }else{
        //Errore nel caso la richiesta del dizionario fallisca
        alert("Errore nel caricamento del dizionario, codice: " +
        this.status);
      }
    }
  }
}
```

Figura 8 Codice caricamento dizionario

5.3 Generazione della griglia

L'algoritmo di generazione della griglia si trova nel file "grid-generator.js".

Inizialmente questo algoritmo rileva l'evento della pressione del pulsante genera,

successivamente viene controllato se la griglia è già stata generata.

Se la griglia è già stata generata viene avviato l'algoritmo per la scelta e l'inserimento della parola finale (descritto successivamente), altrimenti viene avviato l'algoritmo per generare la griglia.

Per prima cosa viene fatto un controllo sui caratteri minimi e massimi, successivamente vengono prese tutte le parole inserite negli input e memorizzate in un array, poi viene chiamato il metodo *generateTable()* che genererà la tabella.

```
if(charCounter >= 140 && charCounter <= 147){
    initGrid();
    words = [];
    for(let input of inputs){
        let value = input.value.trim();
        if(input.style.visibility = "visible" && value != ""){
            if(value.length > 15){ //Se la parola è più lunga di 15 caratteri
                value = value.substring(0, 15); //Tronca la parola a 15 caratteri
            }
            words.push(value);
        }
    }
    //Ordinare l'array dalla parola più lunga (indice 0) a quella più corta
    words = words.sort((a, b) => b.length - a.length);
    generateTable();
    alreadyGenerated = true;
}else{
    Swal.fire({
        icon: 'error',
        title: 'Impossibile generare la griglia!',
        text: 'Numero di caratteri massimi/minimi non rispettato',
    });
}
```

Figura 9 Controlli per la generazione della griglia

Nel metodo *generateTable()* vengono inizialmente fatte scorrere tutte le celle della griglia e per ogni cella viene controllato se c'è abbastanza spazio per posizionare la parola in tutte le 8 direzioni possibili.

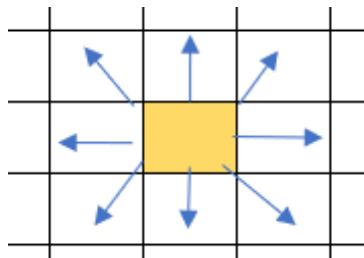


Figura 10 Direzioni in cui è possibile piazzare una parola

Per controllare se c'è abbastanza spazio per posizionare la parola viene utilizzato il metodo *isValidDirection()* (riga 165-184 grid-generator.js)

```
/**
 * Funzione per verificare se a partire da una determinata cella della griglia
 * è possibile piazzare una parola in una determinata posizione
 * la funzione controlla se c'è abbastanza spazio per posizionare la parola
 * nella direzione passata come parametro
 * @param i coordinata della riga
 * @param j coordinata della colonna
 * @param len lunghezza della parola da piazzare
 * @param direction direzione in cui piazzare la parola (costante Direction)
 * @returns true se c'è abbastanza spazio per la parola, false altrimenti
 */
function isValidDirection(i, j, len, direction){
    switch(direction){
        case Direction.UP:
            return i - len >= 0;
        case Direction.UP_RIGHT:
            return i - len >= 0 && j + len <= 14;
        case Direction.RIGHT:
            return j + len <= 14;
        case Direction.DOWN_RIGHT:
            return i + len <= 9 && j + len <= 14;
        case Direction.DOWN:
            return i + len <= 9;
        case Direction.DOWN_LEFT:
            return i + len <= 9 && j - len >= 0;
        case Direction.LEFT:
            return j - len >= 0;
        case Direction.UP_LEFT:
            return i - len >= 0 && j - len >= 0;
    }
}
```

Figura 11 Funzione isValidDirection()

Successivamente per ogni direzione in cui c'è spazio per posizionare la parola viene controllato se è effettivamente possibile inserire la parola, guardando se in quella direzione ci sono degli spazi vuoti o delle lettere uguali che quindi possono essere sovrascritte, questo controllo viene fatto dal metodo *isWordPlaceable()* (riga 199-298 grid-generator.js)

Infine viene passato alla funzione *placeWord()* (riga 305-388 grid-generator.js) un array contenente tutte le celle con le direzioni in cui è possibile piazzare la parola, in questa funzione viene presa casualmente una delle celle valide e viene piazzata la parola nella griglia

Dopo che tutte le parole sono state piazzate nella griglia viene visualizzata un'anteprima provvisoria, in quanto manca ancora l'inserimento della parola finale e delle lettere casuali per coprire gli spazi vuoti.

Insieme all'anteprima provvisoria viene anche visualizzata la lista delle parole finali proposte in automatico dall'applicativo

F			H	A	M	B	U	R	G	E	R		M	
I	F	A	N	T	A	S	M	A		A	L	A	C	S
O	O	L	L	E	R	R	A	C	R		C	C	P	
R	A		A		R		A	E	G	C		I	S	L
E	D	A	N			I	M	N	H	U	A	O	O	A
V	A	L	I	G	I	A	M	E	C	G	F	T	F	T
	P	O	C		C		R	R	G	I	T	O	I	T
	M	U	U			O		I	O	E	A	L	T	I
	A	C	C		N		A		L	D		A	T	N
	L	S		I		C	A	S	T	E	L	L	O	A

hamburger	fiore	carrello	scala	scuola
arancia	dormire	cucina	fantasma	spiaggia
gufo	letto	valigia	lattina	maccheroni
lampada	castello	soffitto	ciotola	camera

Figura 13 Anteprima provvisoria griglia generata

Parole finali proposte:

competano
covavi
vuoteremmo
valutava
ricolmi
rassodo
istituisco
torniranno
alzo
pedinata

Figura 12 Esempio lista parole finali proposte

L'algoritmo per mostrare le parole finali prese casualmente dal dizionario è inserito nel file "final-word-chooser.js", in questo algoritmo viene fatta la richiesta del dizionario al server e successivamente tramite la funzione *searchWords()* vengono selezionate 10 parole casuali che verranno poi mostrate all'utente tramite la funzione *showFinalWords()*

```
/**
 * Funzione che cerca 10 parole casuali da proporre
 * @returns array contenente le 10 parole trovate casualmente dal dizionario
 */
function searchWords(){
    let fWords = [];
    while(fWords.length != 10){
        let w = dictionary[rand(0,dictionary.length-1)].childNodes[0].nodeValue;
        //Controllo per verificare che la parola non superi il limite
        //massimo di caratteri (spazi vuoti rimasti nella griglia)
        if(w.length <= countEmptySpaces()){
            fWords.push(w);
        }
    }
    return fWords;
}
```

Figura 14 Funzione che cerca le parole finali da proporre

Una volta che l'utente ha inserito una parola finale oppure l'ha selezionata da quelle proposte può premere nuovamente sul pulsante genera, partirà nuovamente l'algoritmo di generazione della griglia che però invece che generare una nuova griglia si occuperà di inserire la parola finale e di riempire gli spazi vuoti con lettere casuali.

```
/**
 * Funzione che piazza la parola finale passata come parametro casualmente
 * @param fw final word (parola finale) da piazzare
 */
function placeFinalWord(fw){
    countEmptySpaces();
    fw = fw.toUpperCase();
    for(let i = 0; i < fw.length; i++){
        let coords = emptySpacesCoords[rand(0,emptySpacesCoords.length-1)];
        grid[coords[0]][coords[1]] = fw[i];
        emptySpacesCoords = removeItemOnce(emptySpacesCoords, coords);
    }
}
```

Figura 15 Funzione per inserire la parola finale nella griglia

Dopo che la parola finale è stata inserita verrà visualizzata la griglia completamente generata.

F	P	Q	H	A	M	B	U	R	G	E	R	M	M	C
I	F	A	N	T	A	S	M	A	R	A	L	A	C	S
O	O	L	L	E	R	R	A	C	R	R	C	C	P	I
R	A	C	A	L	R	E	A	E	G	C	Q	I	S	L
E	D	A	N	W	F	I	M	N	H	U	A	O	O	A
V	A	L	I	G	I	A	M	E	C	G	F	T	F	T
I	P	O	C	N	C	L	R	R	G	I	T	O	I	T
D	M	U	U	O	V	O	M	I	O	E	A	L	T	I
D	A	C	C	N	N	X	A	O	L	D	G	A	T	N
B	L	S	A	I	M	C	A	S	T	E	L	L	O	A

hamburger	fiore	carrello	scala	scuola
arancia	dormire	cucina	fantasma	spiaggia
gufo	letto	valigia	lattina	maccheroni
lampada	castello	soffitto	ciotola	camera

ricolmi

Figura 16 Esempio griglia completamente generata

5.4 Generazione automatica della griglia

L'autogenerazione della griglia è contenuta nel file “auto-gen.js”, in questo script c'è la funzione *writeWords()* che si occupa di inserire delle parole prese casualmente dal dizionario negli input delle parole.

Questa funzione è chiamata nel file principale di generazione della griglia (“grid-generator.js”), dopo che la funzione ha inserito tutte le parole negli input, viene chiamata la generazione standard.

```
let words = [];
let charCount = 0;
while(charCount < 140){
    let w = getRandomWord();
    if(words.includes(w)){ //Saltare la parola se esiste già nell'array
        continue;
    }
    if(words.length >= 30){
        words.pop();
    }
    //Controllo per non superare il massimo di caratteri
    if(charCount + w.length <= 147){
        words.push(w);
        charCount += w.length;
    }
}
for(let i = 0; i < words.length; i++){
    wordsInput[i].style.visibility = "visible";
    wordsInput[i].value = words[i];
    changeBorderInput(wordsInput[i]);
}
updateCharsCount();
```

Figura 17 Inserimento parole casuali negli input

```
/**
 * Funzione che ritorna una parola casuale presa dal dizionario
 * @returns parola casuale del dizionario
 */
function getRandomWord(){
    return dictionary[rand(0,dictionary.length-1)].childNodes[0].nodeValue;
}
```

Figura 18 Funzione getRandomWord()

5.5 Esportazione della pagina

Al termine della generazione, la griglia può essere esportata nei seguenti formati: JPEG, PNG, PDF. Per l'esportazione come immagine viene utilizzata la libreria *html2canvas* che prende come parametro un elemento del DOM e lo esporta come immagine.

Nello script “page-export.js”, al click del bottone per esportare la pagina viene visualizzato un alert dove è possibile selezionare il formato, successivamente viene chiamata la funzione *exportAsImage()* o *exportAsPdf()*

```
html2canvas(document.getElementsByClassName("page-box")[0]).then(canvas => {
    let date = new Date();
    const img = canvas.toDataURL(`image/${type}`);
    const link = document.createElement('a');
    link.href = img;
    link.download =
`TrovaParole_${date.getFullYear()}_${date.getMonth()+1}_${date.getDate()}_${date.get
Hours()}_${date.getMinutes()}_${date.getSeconds()}.${type}`;
    link.click();
});
```

Figura 19 Esportazione della pagina come immagine

Nella funzione per l'esportazione della pagina come PDF viene semplicemente chiamata la funzione *window.print()* che apre la finestra di dialogo della stampa del browser. Da questa finestra l'utente può decidere se salvare il documento come PDF (tramite la stampante PDF virtuale) oppure se procedere a stamparlo su carta.

```
/**
 * Funzione per esportare la pagina come pdf, la funzione apre
 * solamente la pagina di stampa del browser
 */
function exportAsPdf(){
    truncateTitle(); //Tronca il titolo a max. 60 caratteri
    window.print();
}
```

Figura 20 Funzione exportAsPdf()

5.6 Modifica del dizionario (client)

La modifica del dizionario avviene attraverso l'interfaccia di modifica, e supporta l'aggiunta, l'eliminazione e la modifica di una parola.

Nell'interfaccia di modifica è presente una tabella contenente tutte le parole del dizionario, di fianco a ogni parola c'è un'icona per modificarla oppure eliminarla. Per l'aggiunta di una nuova parola è necessario andare alla fine della parte caricata del dizionario.







3	asino	 
4	<input type="text" value="caldo"/>	 
5	pesca	 

Figura 21 Esempio modifica di una parola nel dizionario

Lo script che contiene l'algoritmo per gestire queste azioni è "edit-dictionary.js".

Tutte le modifiche vengono salvate nell'array *dictionaryChanges*, che successivamente al salvataggio di quest'ultima viene inviato al server.

Per ogni azione è presente una funzione che si occupa dell'aggiunta di quest'ultima nell'array.

```
/**
 * Funzione per aggiungere all'array dictionaryChanges l'evento
 * di un aggiornamento di una parola
 * @param id id della parola nella tabella
 * @param value nuovo valore aggiornato della parola
 */
function addUpdateAction(id, value){
    dictionaryChanges.push(["update", id, value]);
}
```

Figura 22 Funzione addUpdateAction()

Quando viene premuto il pulsante per salvare le modifiche, l'array *dictionaryChanges* viene convertito in un'oggetto JSON e viene inviato al server tramite la funzione *sendData()*. La funzione invia i dati al server tramite il metodo POST, all'endpoint “/change-dictionary”

```
function sendData(){
    if(dictionaryChanges.length != 0){
        let xhttp = new XMLHttpRequest();
        xhttp.open("POST", "/change-dictionary", true);
        xhttp.setRequestHeader('Content-Type', 'application/json');
        xhttp.onreadystatechange = function() {
            if (this.readyState == 4) {
                if(this.status == 200){
                    Swal.fire({
                        icon: 'success',
                        title: 'Modifiche applicate con successo!',
                    }).then((result) => {
                        window.location.reload();
                    });
                    dictionaryChanges = [];
                }else{
                    Swal.fire({
                        icon: 'error',
                        title: 'Impossibile modificare il dizionario!',
                        text: `Codice errore: ${this.status}`,
                    });
                }
            }
        }
        xhttp.send(JSON.stringify({ dictionaryChanges: dictionaryChanges }));
    }else{
        Swal.fire({
            icon: 'info',
            title: 'Non ci sono modifiche da salvare!',
        });
    }
}
```

Figura 23 Funzione sendData()

5.7 Modifica del dizionario (server)

Quando il server riceve i dati per la modifica del dizionario chiama la funzione *executeAllActions(dictionaryChanges)* (presente nel file “dictionary-manager.js”), a questa funzione passa come parametro l’array che ha ricevuto contenente tutte le modifiche da apportare al dizionario.

Questa funzione inizialmente chiama la funzione *loadWholeDictionary()* per caricare le parole del dizionario all’interno dell’array *words*, successivamente controlla ogni azione e in base al tipo (update/delete/insert) chiama altre funzioni che si occupano di modificare l’array *words*. Infine effettua una chiamata anche alla funzione *updateWholeDictionary()* che si occupa scrivere tutte le parole contenute nell’array *words* all’interno del dizionario (“dictionary.xml”).

```
function executeAllActions(actions){
    errors = [];
    loadWholeDictionary();
    for(let i = 0; i < actions.length; i++){
        if(actions[i][0] == "update"){
            updateAction(sanitizeInput(actions[i][1]), sanitizeInput(actions[i][2]));
        }else if(actions[i][0] == "delete"){
            deleteAction(sanitizeInput(actions[i][1]));
        }else if(actions[i][0] == "insert"){
            insertAction(sanitizeInput(actions[i][1]), sanitizeInput(actions[i][2]));
        }
    }
    updateWholeDictionary();
}
```

Figura 24 Funzione executeAllActions()

Nello script viene anche utilizzata la funzione *sanitizeInput()* che si occupa di rimuovere eventuali caratteri che potrebbero essere pericolosi ad esempio per injection di script.

```
function sanitizeInput(value){
    if(typeof value === "string"){
        value = value.trim();
        value = value.replace(/[\\\/\]/g, "");
        value = sanitizeHtml(value, {allowedTags: [], allowedAttributes: {}});
        if(value.length > 15){
            value = value.substring(0, 15);
        }
    }
    return value;
}
```

Figura 25 Funzione sanitizeInput()

5.8 Creazione test generazione griglia

Per testare la generazione della griglia ho creato uno script di test (“gen-TEST.js”). Questo script va incluso nell’head della pagina quando si vogliono effettuare dei test, e successivamente bisogna richiamare le funzioni dalla console del browser. Questo script contiene la funzione *genWithColorsTest()* che si occupa di leggere l’array *placedWordsInfo* che contiene tutte le posizioni in cui sono state posizionate le parole. Questa funzione successivamente scorre tutta la griglia e colora le parole con dei colori generati casualmente.

U	E	B	A	A	T	R	O	T	B	A	G	N	O	C
O	M	C	D	I	T	O	N	I	C	L	U	P	I	E
V	A	I	L	A	I	H	C	C	O	K	M	A	R	E
A	N	E	E	I	L	O	T	T	A	C	O	I	G	S
N	G	L	T	W	X	D	O	R	M	I	R	E	L	E
I	E	O	I	I	A	L	M	A	E	S	T	R	O	V
L	L	C	V	K	O	T	A	R	T	A	R	U	G	A
L	A	L	A	T	T	U	G	A	A	L	O	I	V	I
A	F	K	P	A	N	T	A	L	O	N	I	O	R	H
G	M	O	T	O	O	T	T	O	C	S	I	B	T	C

dormire	viola	vite	biscotto	mare
chiave	cielo	bagno	kiwi	torta
pulcino	maestro	pantaloni	falegname	moto
dito	uova	gallina	tartaruga	giocattoli
occhiali	ciao	lattuga		

sorella

Figura 26 Esempio test generazione griglia con colori

6 Test

6.1 Protocollo di test

Test Case	TC-001	Nome	Inserimento delle parole
Riferimento	REQ-01		
Descrizione	L'utente dovrà inserire delle parole per permettere la generazione della griglia trova parole		
Prerequisiti	Pagina di creazione del trova parole		
Procedura	<ol style="list-style-type: none"> 1. Aprire la pagina di generazione del trova parole 2. Cliccare sul pulsante "Inserire parola" 3. Inserire una parola nel campo di testo che appare 		
Risultati attesi	L'utente riesce a inserire correttamente le parole		

Test Case	TC-002	Nome	Scelta parola finale
Riferimento	REQ-02		
Descrizione	L'utente deve poter scegliere la parola finale in base al numero di caratteri rimasti liberi nel trova parole		
Prerequisiti	Griglia del trova parole generata		
Procedura	<ol style="list-style-type: none"> 1. Cliccare sul pulsante "Inserire parola finale" 2. Inserire la parola finale nel campo di testo che appare 3. Cliccare sul pulsante "Genera" 		
Risultati attesi	La griglia viene generata nuovamente con all'interno la parola finale scelta dall'utente		

Test Case	TC-003	Nome	Parola finale proposta
Riferimento	REQ-03		
Descrizione	L'applicativo propone una lista di parole finali che l'utente può decidere di utilizzare come parola finale		
Prerequisiti	Pagina di creazione del trova parole		
Procedura	<ol style="list-style-type: none"> 1. Inserire delle parole 2. Cliccare sul pulsante “Genera” 3. Verificare che appare una lista con delle parole finali proposte dall'applicativo 		
Risultati attesi	Viene visualizzata una tabella contenente delle parole finali che è possibile utilizzare		

Test Case	TC-004	Nome	Generazione della pagina in A4
Riferimento	REQ-04		
Descrizione	L'applicativo deve generare una pagina in formato A4		
Prerequisiti	Griglia del trova parole generata compresa di parola finale		
Procedura	<ol style="list-style-type: none"> 1. Cliccare sul pulsante “Esporta” 2. Selezionare uno dei formati proposti 3. Aprire il documento scaricato e verificare che sia in formato A4 		
Risultati attesi	Il documento che l'applicativo esporta è in formato A4		

Test Case	TC-005	Nome	Grandezza della griglia proporzionale
Riferimento	REQ-05		
Descrizione	L'applicativo deve generare la griglia con una grandezza proporzionale a resto della pagina		
Prerequisiti	Griglia del trova parole generata compresa di parola finale		
Procedura	<ol style="list-style-type: none"> 1. Cliccare sul pulsante “Esporta” 2. Selezionare uno dei formati proposti 3. Aprire il documento scaricato e verificare che la griglia sia proporzionale al resto della pagina 		
Risultati attesi	All'interno del documento che l'applicativo esporta la grandezza della griglia è proporzionale al resto della pagina		

Test Case	TC-006	Nome	Corretta posizione delle parole
Riferimento	REQ-06		
Descrizione	La griglia generata dovrà contenere le parole posizionate orizzontalmente, verticalmente e diagonalmente		
Prerequisiti	Pagina caricata con le parole già inserite		
Procedura	<ol style="list-style-type: none"> 1. Cliccare sul pulsante “Genera” 2. Verificare che le parole nella griglia siano nella posizione corretta 		
Risultati attesi	Le parole all'interno della griglia sono posizionate orizzontalmente, verticalmente o diagonalmente		

Test Case	TC-007	Nome	Generazione della griglia con dati dell'utente
Riferimento	REQ-07		
Descrizione	La griglia dovrà essere generata correttamente con le parole inserite dall'utente		
Prerequisiti	Pagina caricata con le parole già inserite		
Procedura	<ol style="list-style-type: none"> 1. Cliccare sul pulsante “Genera” 2. Verificare che la griglia contenga tutte le parole inserite dall'utente 		
Risultati attesi	La griglia viene generata correttamente con all'interno tutte le parole inserite dall'utente		

Test Case	TC-008	Nome	Griglia generata automaticamente
Riferimento	REQ-08		
Descrizione	La griglia deve poter essere generata automaticamente		
Prerequisiti	Pagina caricata senza nessuna parola inserita		
Procedura	<ol style="list-style-type: none"> 1. Cliccare sul pulsante “Genera automaticamente” 2. Verificare che la griglia venga generata senza errori e sia compresa di parola finale 		
Risultati attesi	La griglia viene generata completamente in automatico, ed è compresa di parola finale		

Test Case	TC-009	Nome	Griglia di dimensioni corrette
Riferimento	REQ-09		
Descrizione	La griglia dovrà essere delle corrette dimensioni		
Prerequisiti	Pagina caricata senza nessuna parola inserita		
Procedura	<ol style="list-style-type: none"> 1. Inserire delle parole fino ad arrivare al minimo di caratteri richiesti 2. Cliccare sul pulsante “Genera” 		
Risultati attesi	La griglia generata dovrà essere larga 15 e alta 10 caratteri		

Test Case	TC-010	Nome	Dizionario delle parole
Riferimento	REQ-10		
Descrizione	L'applicativo dovrà avere un dizionario di parole da utilizzare ad esempio per la generazione automatica		
Prerequisiti	Pagina caricata		
Procedura	<ol style="list-style-type: none"> 1. Cliccare sul pulsante “Modifica dizionario” 2. Verificare che nell’interfaccia di modifica sia presente il dizionario 		
Risultati attesi	Nell’interfaccia dovrà essere presente il dizionario rappresentato in una tabella		

Test Case	TC-011	Nome	Esportazione della pagina
Riferimento	REQ-11		
Descrizione	La pagina dovrà poter essere esportata in formato PNG e altri formati		
Prerequisiti	Pagina con griglia già completamente generata		
Procedura	<ol style="list-style-type: none"> 1. Cliccare sul pulsante “Esporta” 2. Verificare che venga proposto di esportare in formato PNG e in altri formati 		
Risultati attesi	Quando si esporta la pagina viene proposto il formato PNG e anche altri formati		

Test Case	TC-012	Nome	Qualità delle immagini esportate
Riferimento	REQ-12		
Descrizione	Le immagini che vengono esportate dovranno essere di alta qualità		
Prerequisiti	Pagina con griglia già completamente generata		
Procedura	<ol style="list-style-type: none"> 1. Cliccare sul pulsante “Esporta” 2. Esportare la pagina in formato PNG 3. Aprire l’immagine esportata e verificare che sia di alta qualità 		
Risultati attesi	L’immagine esportata è di alta qualità		

Test Case	TC-013	Nome	Test applicativo su più browser
Riferimento	REQ-13		
Descrizione	L'applicativo dovrà funzionare su più browser		
Prerequisiti	Pagina caricata		
Procedura	1. Testare tutte le funzionalità dell'applicativo sui seguenti browser: - Google Chrome - Microsoft Edge - Firefox - Safari		
Risultati attesi	L'applicativo deve funzionare correttamente su tutti i browser		

Test Case	TC-014	Nome	Test applicativo mobile
Riferimento	REQ-13		
Descrizione	L'applicativo dovrà essere responsive e funzionare sui dispositivi mobili		
Prerequisiti	Pagina caricata		
Procedura	1. Testare tutte le funzionalità dell'applicativo su smartphone e tablet		
Risultati attesi	L'applicativo deve funzionare correttamente sugli smartphone e sui tablet		

Test Case	TC-015	Nome	Modifica del dizionario
Riferimento	REQ-14		
Descrizione	L'utente deve poter aggiungere/rimuovere/modificare le parole presenti nel dizionario		
Prerequisiti	Pagina caricata		
Procedura	<ol style="list-style-type: none"> 1. Cliccare sul pulsante “Modifica dizionario” 2. Dall'interfaccia di modifica provare a: <ul style="list-style-type: none"> - Inserire una nuova parola - Modificare una parola già esistente - Eliminare una parola 3. Cliccare sul pulsante “Salva modifiche” 4. Verificare che il dizionario si sia aggiornato 		
Risultati attesi	Tutte le operazioni di modifica del dizionario devono andare a buon fine		

6.2 Risultati test

Tabella riassuntiva dei test case. Le compprove con gli screenshot dei test sono disponibili nel documento allegato: “Documento comprova dei test”.

Tutti i test sono stati eseguiti sui seguenti browser:

- Google Chrome
- Firefox
- Microsoft Edge

Il browser “Safari” non è stato testato per mancanza di infrastruttura.

6.2.1 Tabella riassuntiva test su Google Chrome, Firefox e Microsoft Edge

Test Case	Risultato ottenuto	Stato
TC-001	L'utente riesce ad inserire le parole nella pagina.	Passato
TC-002	La griglia viene generata e contiene la parola finale scelta dall'utente.	Passato
TC-003	L'applicativo visualizza la tabella con le parole finali proposte prese dal dizionario.	Passato
TC-004	Il documento che l'applicativo esporta è in formato A4.	Passato
TC-005	Nel documento esportato la grandezza della griglia è proporzionale al resto della pagina.	Passato
TC-006	Le parole all'interno della griglia vengono posizionate correttamente, in direzione orizzontale, verticale e diagonale.	Passato
TC-007	La griglia viene generata correttamente ma non necessariamente tutte le parole dell'utente vengono inserite. Alcune volte l'algoritmo non riesce a trovare spazio per piazzare tutte le parole.	Parzialmente passato
TC-008	La griglia viene generata in modo completamente automatico e la generazione avviene senza errori.	Passato
TC-009	La griglia generata è delle dimensioni 10x15	Passato
TC-010	Nell'interfaccia di modifica viene caricato correttamente il dizionario sottoforma di tabella.	Passato
TC-011	Quando si clicca sul pulsante per l'esportazione viene proposto il formato JPEG, PNG e PDF	Passato
TC-012	Le immagini che vengono esportate sono di alta qualità	Passato
TC-015	Tutte le operazioni di modifica sono andate a buon fine	Passato

6.2.2 Tabella riassuntiva altri test

Test Case	Risultato ottenuto	Stato
TC-013	L'applicativo è stato testato ed è funzionante su Google Chrome, Firefox e Microsoft Edge. Non è stato possibile testare l'applicativo su Safari per mancanza di infrastruttura.	Parzialmente passato
TC-014	L'applicativo non è stato testato sui dispositivi mobili per mancanza di tempo.	Non testato

6.3 Mancanze/limitazioni conosciute

L'applicativo è un po' limitato per quanto riguarda l'accessibilità. Alcuni testi potrebbero risultare troppo piccoli da leggere, inoltre alcuni colori (come ad esempio il giallo chiaro utilizzato per segnalare una parola duplicata) potrebbero risultare poco visibili.

L'applicativo funziona con un solo dizionario, quindi se viene utilizzato da più utenti e ognuno apporta delle modifiche al dizionario, quest'ultime verranno utilizzate anche per il resto degli utenti senza che necessariamente lo desiderino.

L'algoritmo di generazione della griglia può non riuscire a piazzare determinate parole per una mancanza di spazio, questo comporta ad un riempimento degli spazi bianchi con lettere casuali che potrebbero rischiare di formare qualche altra parola non desiderata.

L'applicativo non è stato testato sul browser Safari e nemmeno sui dispositivi mobili, quindi non è possibile assicurare il suo corretto funzionamento su quest'ultimi.

7 Consuntivo

Diagramma di Gantt consuntivo del progetto:

1			▸ Progetto Generatore Trova Parole	77.33 h	ven 01.09.23	ven 01.12.23	
2			▸ Analisi	8 h	ven 01.09.23	ven 08.09.23	
3			Intervista cliente	2 h	ven 01.09.23	ven 01.09.23	
4			QdC	2 h	ven 08.09.23	ven 08.09.23	3
5			Use case	2 h	ven 08.09.23	ven 08.09.23	4
6			Stesura requisiti	2 h	ven 08.09.23	ven 08.09.23	5
7			▸ Progettazione	12 h	ven 15.09.23	ven 22.09.23	
8			Gantt	3 h	ven 15.09.23	ven 15.09.23	2
9			Mockup interfacce	4 h	ven 15.09.23	ven 22.09.23	8
10			Diagramma flusso	3 h	ven 22.09.23	ven 22.09.23	9
11			Test case	2 h	ven 22.09.23	ven 22.09.23	10
12			▸ Implementazione	46 h	ven 29.09.23	ven 24.11.23	
13			▸ Creazione GUI	14 h	ven 29.09.23	ven 13.10.23	
14			Main	6 h	ven 29.09.23	ven 29.09.23	7
15			Modifica dizionario	8 h	ven 06.10.23	ven 13.10.23	14
16			▸ Algoritmi	32 h	ven 13.10.23	ven 24.11.23	
17			Generazione griglia	16 h	ven 13.10.23	ven 27.10.23	13
18			Esportazione	8 h	ven 10.11.23	ven 17.11.23	17
19			Modifica dizionario	8 h	ven 17.11.23	ven 24.11.23	18
20			▸ Integrazione	8 h	ven 24.11.23	ven 01.12.23	
21			Test integrazione	8 h	ven 24.11.23	ven 01.12.23	12
22			Documentazione	77.33 h	ven 01.09.23	ven 01.12.23	

Figura 27 Diagramma di Gantt consuntivo

Il progetto ha seguito la linea della pianificazione fino alla fase di implementazione. In questa fase non ho previsto accuratamente il tempo da dedicare ad ogni task, inizialmente ho accumulato un ritardo con la fase di creazione delle GUI che era prevista per durare 8h invece ho impiegato 14h.

Successivamente anche la creazione dell'algoritmo di generazione della griglia ha richiesto il doppio del tempo di lavoro previsto.

A causa di questo ritardo ho dovuto eliminare la parte del bug fixing e dei test di accettazione dell'applicativo.

8 Conclusioni

8.1 Sviluppi futuri

In futuro si potrebbe aggiungere all'applicativo un sistema di login per permettere ad ogni dipendente di avere il proprio account, con il proprio dizionario da utilizzare. Inoltre si potrebbe anche sviluppare una funzionalità per salvare i “trova parole” generati sul server, in modo che ogni dipendente con il proprio account possa salvare i giochi che genera e magari anche modificarli.

Un'altra funzionalità interessante da implementare sarebbe un bottone che se premuto visualizza/nasconde la soluzione del trova parole, in questo modo sarà anche possibile esportare le soluzioni del gioco.

8.2 Considerazioni personali

Per concludere posso affermare che grazie a questo progetto ho imparato che in futuro dovrò dare più importanza e anche più attenzione alla pianificazione delle ore, visto che in questo progetto mi sono ritrovato ad aver pianificato alcune task con una durata troppo corta, e di conseguenza ho avuto meno tempo a disposizione ad esempio per effettuare i test dell'applicativo.

9 Bibliografia

9.1 Sitografia

- <https://nodejs.org/en>, 22-09-2023
- <https://expressjs.com/it/>, 22-09-2023
- <https://color.adobe.com/it/create/color-wheel>, 22-09-2023
- https://www.w3schools.com/xml/xml_http.asp, 13-10-2023
- <https://fontawesome.com/>, 13-10-2023
- <https://sweetalert2.github.io/>, 10-11-2023
- <https://www.npmjs.com/package/morgan>, 10-11-2023
- <https://html2canvas.hertzen.com/>, 10-11-2023
- https://www.w3schools.com/js/js_html_dom_navigation.asp, 17-11-2023
- <https://www.npmjs.com/package/sanitize-html>, 24-11-2023
- <https://regex101.com/>, 24-11-2023
- https://www.w3schools.com/nodejs/nodejs_filesystem.asp, 24-11-2023
- <https://stackoverflow.com/questions/39519246/make-xmlhttprequest-post-using-json>, 24-11-2023

10 Glossario

Termine	Significato
PDF	Portable Document Format. È un formato di file versatile creato da Adobe che permette di presentare e scambiare documenti in modo semplice e affidabile
JSON	Javascript Object Notation. È un formato per lo scambio dati basato sul linguaggio di programmazione JavaScript
endpoint	È l'URL specifico utilizzato per accedere a una risorsa fornita da un'applicazione web
DOM	Document Object Model. È un modello ad oggetti che compongono la struttura e il contenuto di un documento sul web.

11 Indice delle figure

Figura 1 Use case dell'applicativo	8
Figura 2 Diagramma di Gantt del progetto.....	9
Figura 3 Design dell'interfaccia principale.....	11
Figura 4 Design dell'interfaccia "Modifica dizionario"	12
Figura 5 Esempio interfaccia inserimento parole	14
Figura 6 Esempio rimozione parola dall'interfaccia di inserimento.....	14
Figura 7 Struttura del dizionario	15
Figura 8 Codice caricamento dizionario	15
Figura 9 Controlli per la generazione della griglia	16
Figura 10 Direzioni in cui è possibile piazzare una parola	17
Figura 11 Funzione isValidDirection()	17
Figura 12 Esempio lista parole finali proposte	18
Figura 13 Anteprima provvisoria griglia generata	18
Figura 14 Funzione che cerca le parole finali da proporre.....	19
Figura 15 Funzione per inserire la parola finale nella griglia.....	19
Figura 16 Esempio griglia completamente generata	20
Figura 17 Inserimento parole casuali negli input.....	21
Figura 18 Funzione getRandomWord().....	21
Figura 19 Esportazione della pagina come immagine	22
Figura 20 Funzione exportAsPdf().....	22
Figura 21 Esempio modifica di una parola nel dizionario	23
Figura 22 Funzione addUpdateAction()	23
Figura 23 Funzione sendData()	24
Figura 24 Funzione executeAllActions()	25
Figura 25 Funzione sanitizeInput()	25
Figura 26 Esempio test generazione griglia con colori.....	26
Figura 27 Diagramma di Gantt consuntivo	36

12 Allegati

- Diari di lavoro
- QdC
- Applicativo web
- Design delle interfacce
- Documento comprova dei test
- Activity Diagram – generazione
- Use case
- Diagramma di Gantt preventivo
- Diagramma di Gantt consuntivo