Generatore trova parole

# Indice

1 Indice 2

2 Introduzione 3

2.1 Informazioni sul progetto 3

2.2 Abstract 3

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 10

3.4 Pianificazione 11

3.5 Analisi dei mezzi 12

3.5.1 Software 12

3.5.2 Hardware 12

4 Progettazione 13

4.1 Design delle interfacce 13

4.2 Design procedurale 15

5 Implementazione 16

5.1 Inserimento delle parole 16

5.2 Interfaccia per la modifica del dizionario 17

5.3 Generazione della griglia 18

5.4 Generazione automatica della griglia 23

5.5 Esportazione della pagina 24

5.6 Modifica del dizionario (client) 25

5.7 Modifica del dizionario (server) 27

5.8 Creazione test generazione griglia 28

6 Test 29

6.1 Protocollo di test 29

6.2 Risultati test 34

6.3 Mancanze/limitazioni conosciute 34

7 Consuntivo 34

8 Conclusioni 35

8.1 Sviluppi futuri 35

8.2 Considerazioni personali 35

9 Bibliografia 36

9.1 Sitografia 36

10 Glossario 37

11 Indice delle figure 37

12 Allegati 38

# Introduzione

## 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

## 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*

## 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.

# Analisi

## 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.

## 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 prole 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 multipiattaforma |
| **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.

## 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.

## 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:

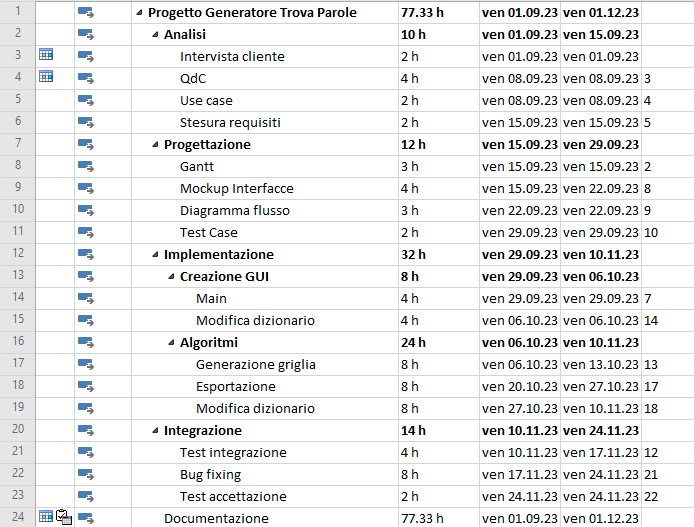


Figura 2 Diagramma di Gantt del progetto

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.

## Analisi dei mezzi

### Software

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

### 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

# Progettazione

## 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.

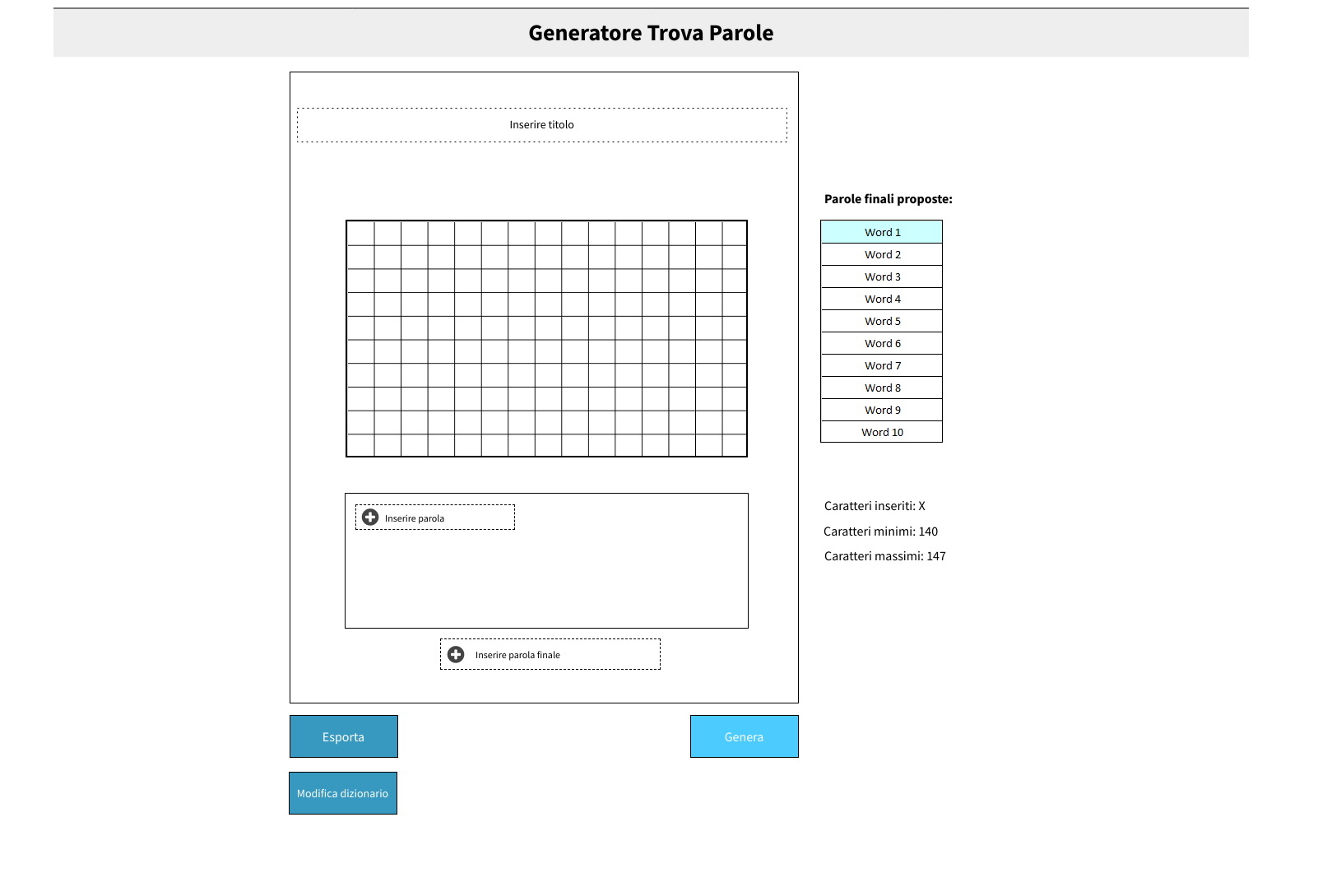


Figura 3 Design dell'interfaccia principale

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.

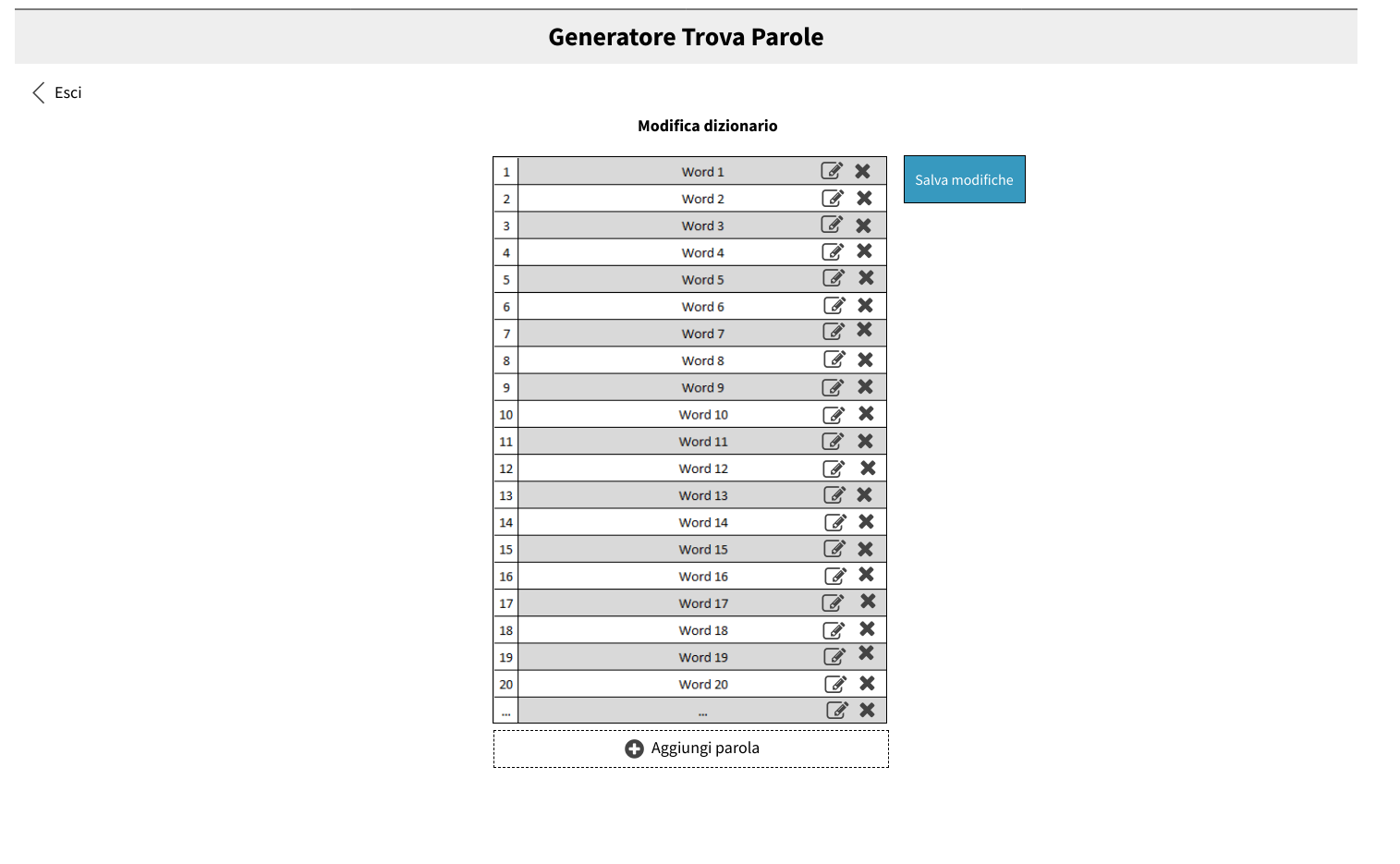


Figura 4 Design dell'interfaccia "Modifica dizionario"

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.

## 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.

# Implementazione

## 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:

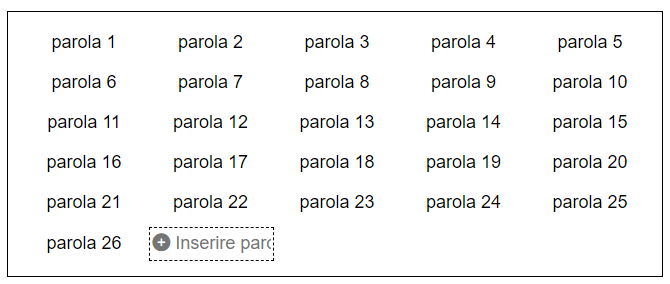


Figura 5 Esempio interfaccia inserimento parole

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

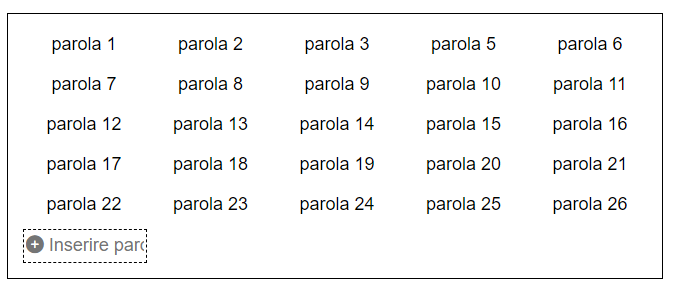


Figura 6 Esempio rimozione parola dall'interfaccia di inserimento

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

## 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

## 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.

Immagine che contiene linea, quadrato, Policromia, Rettangolo

Descrizione generata automaticamente

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

Immagine che contiene testo, schermata, Carattere, numero

Descrizione generata automaticamenteImmagine che contiene testo, schermata, Parallelo, numero

Descrizione generata automaticamente

Figura 12 Esempio lista parole finali proposte

Figura 13 Anteprima provvisoria griglia generata

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 fi 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.

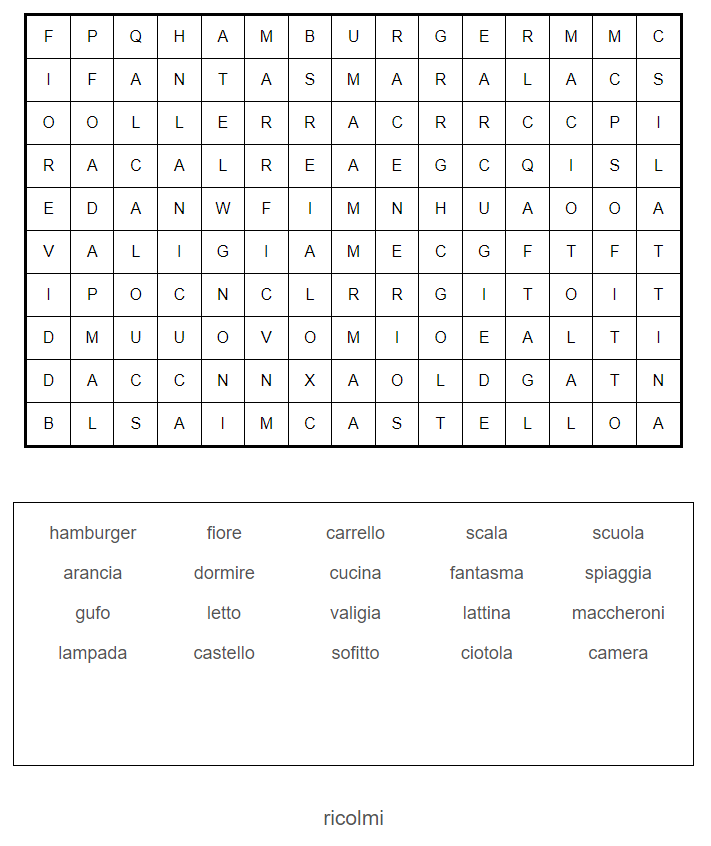


Figura 16 Esempio griglia completamente generata

## 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.

Figura 17 Inserimento parole casuali negli input

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 18 Funzione getRandomWord()

/\*\*

 \* 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;

}

## 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()*

Figura 19 Esportazione della pagina come immagine

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.getHours()}\_${date.getMinutes()}\_${date.getSeconds()}.${type}`;

    link.click();

});

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.

Figura 20 Funzione exportAsPdf()

/\*\*

 \* 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();

}

## 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 parole 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.

Immagine che contiene testo, schermata, numero, Carattere

Descrizione generata automaticamente

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’ultime 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()

## 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*, successivamentecontrolla 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”).

Figura 24 Funzione executeAllActions()

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();

}

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

Figura 25 Funzione sanitizeInput()

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;

}

## 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 legge 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.

Immagine che contiene testo, schermata, Policromia, quadrato

Descrizione generata automaticamente

Figura 26 Esempio test generazione griglia con colori

# Test

## Protocollo di test

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case**  **Riferimento** | TC-001  REQ-01 | **Nome** | Inserimento delle parole |
| **Descrizione** | L’utente dovrà inserire delle parole per permettere la generazione della griglia trova parole | | |
| **Prerequisiti** | Pagina di creazione del trova parole | | |
| **Procedura** | 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**  **Riferimento** | TC-002  REQ-02 | **Nome** | Scelta parola finale |
| **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** | 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**  **Riferimento** | TC-003  REQ-03 | **Nome** | Parola finale proposta |
| **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** | 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**  **Riferimento** | TC-004  REQ-04 | **Nome** | Generazione della pagina in A4 |
| **Descrizione** | L’applicativo deve generare una pagina in formato A4 | | |
| **Prerequisiti** | Griglia del trova parole generata compresa di parola finale | | |
| **Procedura** | 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**  **Riferimento** | TC-005  REQ-05 | **Nome** | Grandezza della griglia proporzionale |
| **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** | 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**  **Riferimento** | TC-006  REQ-06 | **Nome** | Corretta posizione delle parole |
| **Descrizione** | La griglia generata dovrà contenere le parole posizionate orizzontalmente, verticalmente e diagonalmente | | |
| **Prerequisiti** | Pagina caricata con le parole già inserite | | |
| **Procedura** | 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**  **Riferimento** | TC-007  REQ-07 | **Nome** | Generazione della griglia con dati dell’utente |
| **Descrizione** | La griglia dovrà essere generata correttamente con le parole inserite dall’utente | | |
| **Prerequisiti** | Pagina caricata con le parole già inserite | | |
| **Procedura** | 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**  **Riferimento** | TC-008  REQ-08 | **Nome** | Griglia generata automaticamente |
| **Descrizione** | La griglia deve poter essere generata automaticamente | | |
| **Prerequisiti** | Pagina caricata senza nessuna parola inserita | | |
| **Procedura** | 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**  **Riferimento** | TC-009  REQ-09 | **Nome** | Griglia di dimensioni corrette |
| **Descrizione** | La griglia dovrà essere delle corrette dimensioni | | |
| **Prerequisiti** | Pagina caricata senza nessuna parola inserita | | |
| **Procedura** | 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**  **Riferimento** | TC-010  REQ-10 | **Nome** | Dizionario delle parole |
| **Descrizione** | L’applicativo dovrà avere un dizionario di parole da utilizzare ad esempio per la generazione automatica | | |
| **Prerequisiti** | Pagina caricata | | |
| **Procedura** | 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**  **Riferimento** | TC-011  REQ-11 | **Nome** | Esportazione della pagina |
| **Descrizione** | La pagina dovrà poter essere esportata in formato PNG e altri formati | | |
| **Prerequisiti** | Pagina con griglia già completamente generata | | |
| **Procedura** | 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**  **Riferimento** | TC-012  REQ-12 | **Nome** | Qualità delle immagini esportate |
| **Descrizione** | Le immagini che vengono esportate dovranno essere di alta qualità | | |
| **Prerequisiti** | Pagina con griglia già completamente generata | | |
| **Procedura** | 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**  **Riferimento** | TC-013  REQ-13 | **Nome** | Test applicativo su più browser |
| **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**  **Riferimento** | TC-014  REQ-13 | **Nome** | Test applicativo mobile |
| **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**  **Riferimento** | TC-015  REQ-14 | **Nome** | Modifica del dizionario |
| **Descrizione** | L’utente deve poter aggiungere/rimuovere/modificare le parole presenti nel dizionario | | |
| **Prerequisiti** | Pagina caricata | | |
| **Procedura** | 1. Cliccare sul pulsante “Modifica dizionario” 2. Dall’interfaccia di modifica provare a: - 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 | | |

## 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 | Testo o immagine a conferma del test | Passato |
| TC-002 | Testo o immagine a conferma del test | Fallito |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

## Mancanze/limitazioni conosciute

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

# Consuntivo

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

Stile orizzontale

# Conclusioni

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

## Sviluppi futuri

Migliorie o estensioni che possono essere sviluppate sul prodotto.

## Considerazioni personali

Cosa ho imparato in questo progetto? ecc.

# Bibliografia

## 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_htmldom_navigation.asp>, 17-11-2023
* <https://www.npmjs.com/package/sanitize-html>, 24-11-2023
* <https://stackoverflow.com/questions/39519246/make-xmlhttprequest-post-using-json>, 24-11-2023

# 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. |

# Indice delle figure

[Figura 1 Use case dell'applicativo 10](file:///D:\M306\ProgettoM306\GeneratoreTrovaParole\3_Documentazione\Documentazione_GeneratoreTrovaParole.docx#_Toc152275841)

[Figura 2 Diagramma di Gantt del progetto 11](#_Toc152275842)

[Figura 3 Design dell'interfaccia principale 13](#_Toc152275843)

[Figura 4 Design dell'interfaccia "Modifica dizionario" 14](#_Toc152275844)

[Figura 5 Esempio interfaccia inserimento parole 16](#_Toc152275845)

[Figura 6 Esempio rimozione parola dall'interfaccia di inserimento 16](#_Toc152275846)

[Figura 7 Struttura del dizionario 17](file:///D:\M306\ProgettoM306\GeneratoreTrovaParole\3_Documentazione\Documentazione_GeneratoreTrovaParole.docx#_Toc152275847)

[Figura 8 Codice caricamento dizionario 17](file:///D:\M306\ProgettoM306\GeneratoreTrovaParole\3_Documentazione\Documentazione_GeneratoreTrovaParole.docx#_Toc152275848)

[Figura 9 Controlli per la generazione della griglia 18](file:///D:\M306\ProgettoM306\GeneratoreTrovaParole\3_Documentazione\Documentazione_GeneratoreTrovaParole.docx#_Toc152275849)

[Figura 10 Direzioni in cui è possibile piazzare una parola 19](#_Toc152275850)

[Figura 11 Funzione isValidDirection() 19](file:///D:\M306\ProgettoM306\GeneratoreTrovaParole\3_Documentazione\Documentazione_GeneratoreTrovaParole.docx#_Toc152275851)

[Figura 12 Esempio lista parole finali proposte 20](file:///D:\M306\ProgettoM306\GeneratoreTrovaParole\3_Documentazione\Documentazione_GeneratoreTrovaParole.docx#_Toc152275852)

[Figura 13 Anteprima provvisoria griglia generata 20](#_Toc152275853)

[Figura 14 Funzione che cerca le parole finali da proporre 21](file:///D:\M306\ProgettoM306\GeneratoreTrovaParole\3_Documentazione\Documentazione_GeneratoreTrovaParole.docx#_Toc152275854)

[Figura 15 Funzione per inserire la parola finale nella griglia 21](file:///D:\M306\ProgettoM306\GeneratoreTrovaParole\3_Documentazione\Documentazione_GeneratoreTrovaParole.docx#_Toc152275855)

[Figura 16 Esempio griglia completamente generata 22](#_Toc152275856)

[Figura 17 Inserimento parole casuali negli input 23](file:///D:\M306\ProgettoM306\GeneratoreTrovaParole\3_Documentazione\Documentazione_GeneratoreTrovaParole.docx#_Toc152275857)

[Figura 18 Funzione getRandomWord() 23](file:///D:\M306\ProgettoM306\GeneratoreTrovaParole\3_Documentazione\Documentazione_GeneratoreTrovaParole.docx#_Toc152275858)

[Figura 19 Esportazione della pagina come immagine 24](file:///D:\M306\ProgettoM306\GeneratoreTrovaParole\3_Documentazione\Documentazione_GeneratoreTrovaParole.docx#_Toc152275859)

[Figura 20 Funzione exportAsPdf() 24](file:///D:\M306\ProgettoM306\GeneratoreTrovaParole\3_Documentazione\Documentazione_GeneratoreTrovaParole.docx#_Toc152275860)

[Figura 21 Esempio modifica di una parola nel dizionario 25](#_Toc152275861)

[Figura 22 Funzione addUpdateAction() 25](file:///D:\M306\ProgettoM306\GeneratoreTrovaParole\3_Documentazione\Documentazione_GeneratoreTrovaParole.docx#_Toc152275862)

[Figura 23 Funzione sendData() 26](file:///D:\M306\ProgettoM306\GeneratoreTrovaParole\3_Documentazione\Documentazione_GeneratoreTrovaParole.docx#_Toc152275863)

[Figura 24 Funzione executeAllActions() 27](file:///D:\M306\ProgettoM306\GeneratoreTrovaParole\3_Documentazione\Documentazione_GeneratoreTrovaParole.docx#_Toc152275864)

[Figura 25 Funzione sanitizeInput() 27](file:///D:\M306\ProgettoM306\GeneratoreTrovaParole\3_Documentazione\Documentazione_GeneratoreTrovaParole.docx#_Toc152275865)

[Figura 26 Esempio test generazione griglia con colori 28](#_Toc152275866)

# Allegati

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