

## **Documentazione HangZone**

2	Introduzione.....	3
	Informazioni sul progetto.....	3
2.1	Abstract .....	<b>Errore. Il segnalibro non è definito.</b>
2.1.1	Situazione iniziale .....	3
2.1.2	Approccio .....	3
2.1.3	Risultati .....	3
2.2	Scopo .....	4
	Analisi.....	4
2.3	Analisi del dominio .....	4
2.4	Analisi e specifica dei requisiti .....	4
2.5	Use case .....	7
2.6	Pianificazione .....	1
2.6.1	Analisi.....	1
2.6.2	Progettazione .....	1
2.6.3	Implementazione.....	1
	Analisi dei mezzi.....	2
2.6.4	Software .....	3
2.6.5	Hardware.....	3
3	Progettazione .....	4
3.1	Design dei dati e database.....	4
3.1.1	Diagramma ER .....	4
3.2	Design delle interfacce .....	5
3.2.1	Pagina Home .....	5
3.2.2	Pagina Login .....	6
3.2.3	Pagina Registrazione.....	7
3.2.4	Pagina gioco .....	8
3.2.5	Pagina Gestione parole .....	9
3.3	Design procedurale .....	<b>Errore. Il segnalibro non è definito.</b>
4	Implementazione .....	11
5	Test.....	28
5.1	Protocollo di test.....	28
5.2	Risultati test.....	<b>Errore. Il segnalibro non è definito.</b>
5.3	Mancanze/limitazioni conosciute.....	33
6	Consuntivo.....	1
7	Conclusioni .....	1
7.1	Sviluppi futuri.....	1
7.2	Considerazioni personali.....	2
8	Glossario .....	2
9	Bibliografia .....	4
9.1	Bibliografia per articoli di riviste:.....	<b>Errore. Il segnalibro non è definito.</b>
9.2	Bibliografia per libri.....	<b>Errore. Il segnalibro non è definito.</b>
9.3	Sitografia .....	4
10	Allegati .....	<b>Errore. Il segnalibro non è definito.</b>

## 1 Introduzione

---

### Informazioni sul progetto

- Allievo: Tessa Caminada
- Classe: I3AC
- Docente: Ingrid Cereda
- SAMT Progetti
- 5.09.2025 – 19.12.2025

#### 1.1.1 Situazione iniziale

Sul web cercando attentamente non esiste un sito completamente dedicato al gioco dell’impiccato, soprattutto non permettono un login, la possibilità di poter modificare le parole e tenere conto dei propri successi e progressi.

Proprio dopo aver provato gli altri siti di gaming, ho capito quanto non fossero davvero avanzati come si pensava, proprio per questo ho tenuto necessario creare uno. Per poter liberamente giocare ad un gioco nato originariamente per essere fatto su un foglio, sui dispositivi elettronici, che ormai di recente stanno diventando una parte quasi fondamentale della nostra quotidianità.

#### 1.1.2 Approccio

Per questo progetto sono necessarie conoscenze di HTML, JAVASCRIPT,CSS e PHP. Inoltre è necessario anche conoscere MySQL e il suo funzionamento base.

L'utilizzo di HTML permette di realizzare un'applicazione web che sarà poi compatibile con tutti i dispositivi, entrando semplicemente da un qualsiasi motore di ricerca.

Per la realizzazione di questo progetto è stato voluto prima focalizzarsi sulla parte di JavaScript e PHP, ovvero quella più logica e lunga, poi una volta conclusa e accertata quella, sono passata a quella grafica, rendendo il sito più bello e intuitivo all'occhio umano e facendo sì che fosse disponibile su tutti i dispositivi con un layout adatto.

#### 1.1.3 Risultati

Gli obiettivi di questo progetto sono stati raggiunti, l'applicativo web è funzionante, presenti anche tutte le funzionalità che si erano pensate nella progettazione si sono rivelate utili e aggiuntive al gioco standard. L'unica pecca che ho avuto l'impossibilità di aggiungere è la possibilità di inserire le lettere tramite voce, la problematica purtroppo stava nelle prestazioni dell'interpretatore vocale di Google o Microsoft Edge, che si non si sono rivelati abbastanza moderni da comprendere e interpretare le singole lettere dette, anche se dispensi di un microfono.

L'applicativo è stato comunque concluso con successo e molto apprezzato dai clienti di prova che lo hanno utilizzato.

## 1.2 Scopo

Lo scopo di questo progetto è sviluppare una versione avanzata del gioco dell'impiccato in ambiente web, dotata di sistema di login e gestione personalizzata dei dati dell'utente.

L'applicazione permette agli utenti registrati di accedere con il proprio account, salvare e visualizzare le proprie statistiche di gioco e giocare partite con un numero limitato di vite.

Inoltre, il progetto include un'area dedicata agli amministratori che consente di aggiungere, modificare o eliminare le parole presenti nel gioco.

Attraverso questo progetto metto in pratica competenze di sviluppo web, gestione del DOM, manipolazione dei dati, controllo degli accessi e progettazione di un'interfaccia interattiva

## 2 Analisi

---

### 2.1 Analisi del dominio

Sin da bambini giochiamo o abbiamo giocato al gioco dell'impiccato, essendo un gioco molto facile e intuitivo.

Oggi esistono diverse versioni online del gioco, ma sono standard, prive di funzionalità innovative. L'applicativo HangZone nasce con l'obiettivo di rendere più interessante e interattivo il gioco con l'integrazione di poterci giocare in diverse lingue e della possibilità di poter modificare le parole da indovinare.

Questo sito non è indirizzato a nessuna categoria di persone, infatti vuole essere il più user friendly e inclusivo possibile.

Oltre a tutte queste funzionalità sarà possibile decidere quali parole immettere nel gioco, questo potrebbe evitare che certi bambini imparino parole poco eleganti e che certa gente rimanga toccata da certi argomenti legate alle parole, esse potranno essere controllate da una terza persona, il quale avrà il ruolo di amministratore.

### 2.2 Analisi e specifica dei requisiti

Requisito	Requisito 1	Priorità	1	Versione	1.0
Nome	Schermata home				
Note	L'utente deve poter visualizzare la schermata iniziale con i pulsante LOGIN				

Requisito	Requisito 2	Priorità	1	Versione	1.0
Nome	Avvio gioco				
Note	Il sistema sceglie una parola dal file JSON e la visualizza con trattini al posto delle lettere				
Sotto requisiti					
001	Si necessita del corretto funzionamento del Requisito 1				
002	Si deve leggere correttamente il file JSON				
003	Si necessita il corretto funzionamento del login (Requisito 5)				

**HangZone**

Requisito	Requisito 3	Priorità	1	Versione	1.0
Nome	Inserimento lettere scritte				
Note	L'utente deve poter inserire le lettere tramite: Una casella di testo (entry)				
Sotto requisiti					
001	Togliere una vita se la lettera immessa è sbagliata				

Requisito	Requisito 4	Priorità	1	Versione	1.0
Nome	Gestione vite				
Note	L'utente parte con 6 vite Si perde una vita per ogni lettera sbagliata				
Sotto requisiti					
001	Se tutte le vite finiscono, il gioco termina e si ritorna alla schermata home (Requisito 1)				

Requisito	Requisito 5	Priorità	1	Versione	1.0
Nome	Login				
Note	Tutti devono accedere al login per poter giocare				

Requisito	Requisito 6	Priorità	1	Versione	1.0
Nome	Cambio lingua				
Note	È possibile cambiare lingua dell'esperienza e delle parole				

Requisito	Requisito 7	Priorità	1	Versione	1.0
Nome	Visualizzazione parole				
Note	L'amministratore può vedere tutte le parole presenti nel file JSON				
Sotto requisiti					
001	Si necessita il corretto funzionamento del login (Requisito 5)				

Requisito	Requisito 8	Priorità	1	Versione	1.0
Nome	Inserimento parole				
Note	L'amministratore può aggiungere nuove parole al gioco tramite un'interfaccia				
Sotto requisiti					
001	Si necessita il corretto funzionamento della visualizzazione parole (Requisito 6)				

**HangZone**

Requisito	Requisito 9	Priorità	1	Versione	1.0
Nome	Modifica parole				
Note	L'amministratore può modificare le parole del gioco tramite un'interfaccia				
Sotto requisiti					
001	Si necessita il corretto funzionamento della visualizzazione parole (Requisito 6)				

Requisito	Requisito 10	Priorità	1	Versione	1.0
Nome	Eliminazione parole				
Note	L'amministratore può eliminare parole dal gioco tramite un'interfaccia				
Sotto requisiti					
001	Si necessita il corretto funzionamento della visualizzazione parole (Requisito 6)				

**Spiegazione elementi tabella dei requisiti:**

**ID:** identificativo univoco del requisito

**Nome:** breve descrizione del requisito

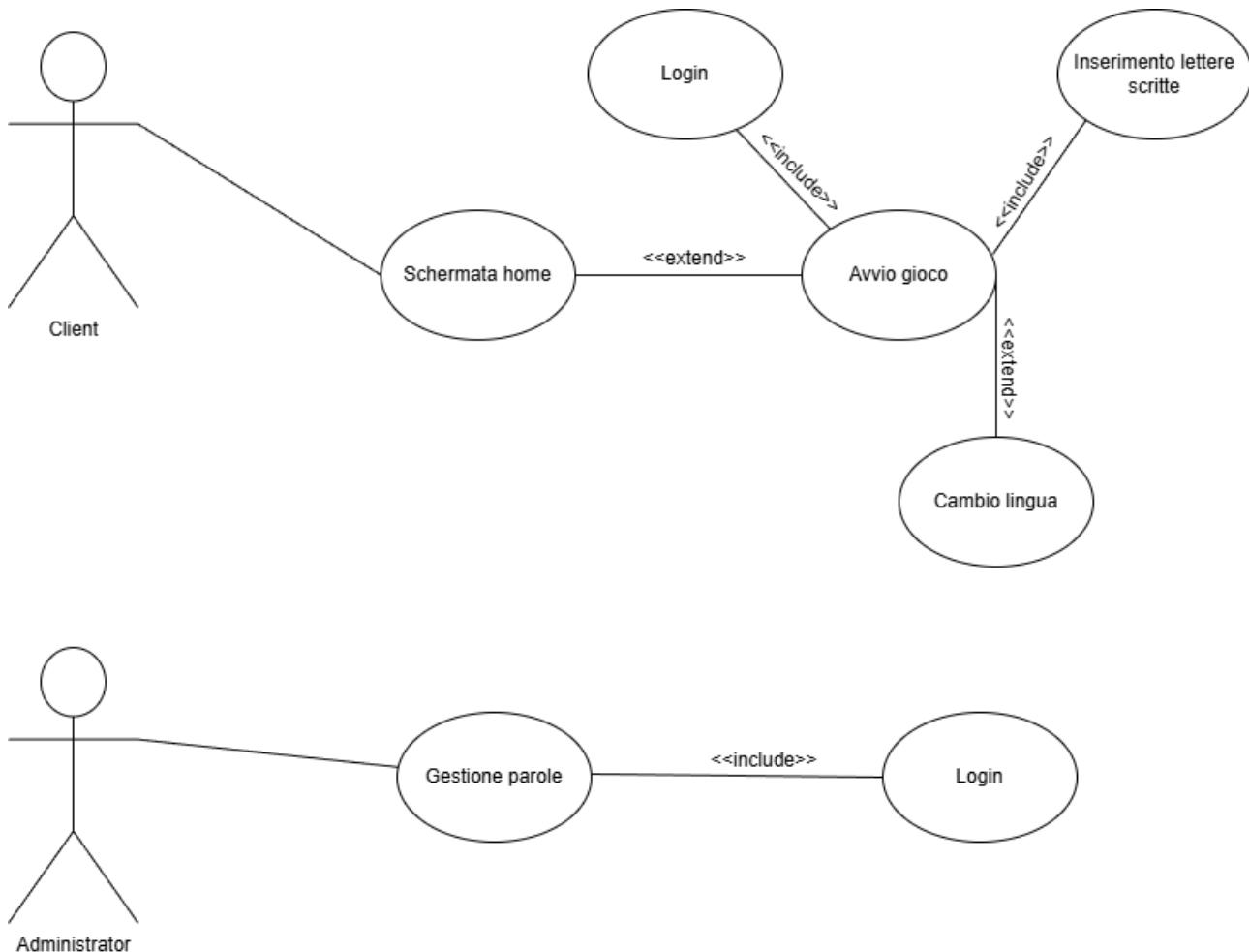
**Priorità:** indica l'importanza di un requisito nell'insieme del progetto, definita assieme al committente. Ad esempio, poter disporre di report con colonne di colori diversi ha priorità minore rispetto al fatto di avere un database con gli elementi al suo interno. Solitamente si definiscono al massimo 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.

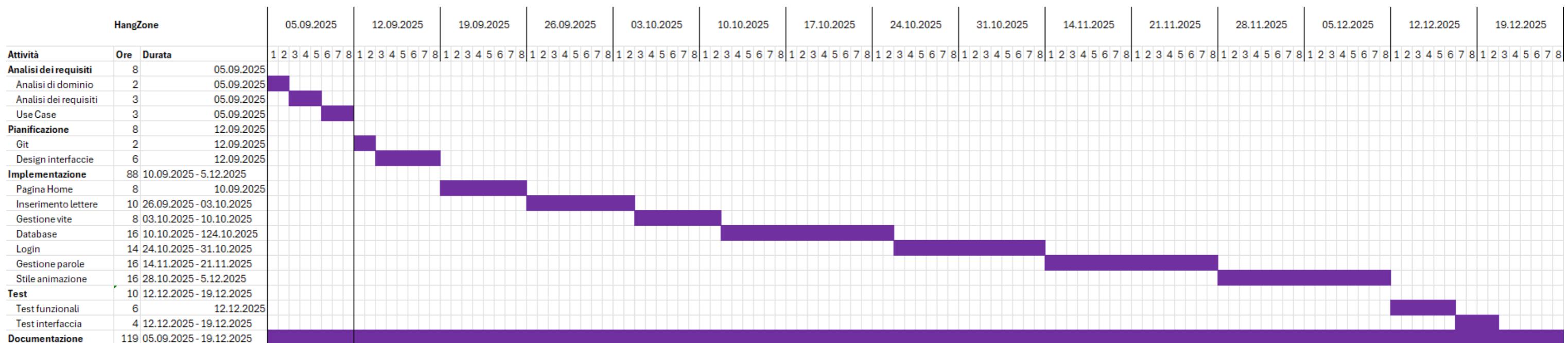
### 2.3 Use case



Questo è lo Use Case che descrive il funzionamento della mia applicazione web. Come si può vedere l'utente può entrare nella schermata home, poi facendo il login, si sceglie di diventare admin oppure semplici client, da lì sarà possibile giocare, cambiare lingua, inserire le lettere e per chi ha le credenziali poter modificare le parole del gioco.

## 2.4 Pianificazione

Questo è il Gantt realizzato per lo svolgimento di questo progetto. Il metodoutilizzato è quello di Waterfall.



#### 2.4.1 Analisi

La fase di analisi del progetto comprende tutte le attività preliminari fondamentali per capire cosa realizzare e come farlo. In questa fase ho letto attentamente il QdC, individuato i requisiti funzionali e non funzionali dell'applicazione.

Una parte consistente del tempo è stata dedicata alla scelta delle tecnologie da utilizzare. Poiché la mia applicazione deve funzionare sia su computer che su smartphone, ho dovuto informarmi sulle soluzioni più adatte per uno sviluppo web responsivo, in particolare su PHP, MySQL, HTML, CSS e JavaScript. In più, per poter implementare il sistema di login, la gestione degli utenti e il pannello amministratore, ho impiegato tempo a comprendere come strutturare correttamente il database.

La scrittura dei requisiti ha richiesto più ore del previsto, anche per via delle presentazioni in classe e della disponibilità non costante del docente. Per pianificare in modo più preciso le tempistiche ho preso come riferimento il Gantt consuntivo del mio progetto precedente, adattandolo a quello recente.

#### 2.4.2 Progettazione

In questa fase ho progettato tutte le parti principali del sistema. Ho realizzato lo Use Case del progetto, il diagramma ER del database con le tabelle dedicate agli utenti e pure il JSON per l'elenco delle parole in tutte le 4 lingue, oltre alla pianificazione tramite diagramma di Gantt.

Ho inoltre iniziato a progettare il design delle interfacce, come la schermata di login e registrazione, la pagina di gioco e il pannello amministratore per la gestione delle parole. Poiché ho già eseguito questo tipo di attività in altri progetti, la progettazione è risultata relativamente veloce e non troppo problematica. Mi sono basata sui tempi del mio progetto precedente e ho stimato circa 8 ore per completare questa fase.

#### 2.4.3 Implementazione

Questa è stata la fase più estesa del lavoro, in cui ho realizzato concretamente tutte le funzionalità dell'applicazione. L'implementazione è stata suddivisa per interfacce: per ogni pagina ho prima creato la struttura grafica "alla buona" (HTML), successivamente ho sviluppato la logica funzionale tramite PHP e JavaScript e infine sono andata a rendere visibilmente bello il sito con CSS.

Ho iniziato dalla parte del login e registrazione per poi passare alla pagina di gioco con la gestione delle vite e delle lettere. In seguito ho sviluppato il pannello amministratore, che permette di aggiungere, modificare o eliminare le parole dal database MySQL.

Pur avendo già esperienza nei progetti web, questa applicazione include funzionalità più complesse, come l'integrazione di un database con PHP, che abbiamo cominciato ad imparare solo pochi mesi fa, quindi inizialmente ho impiegato più tempo. Una volta completata la prima interfaccia, lo sviluppo delle altre è diventato più rapido grazie alla struttura già definita.

L'interfaccia principale è risultata la più semplice poiché contiene soltanto collegamenti verso le altre parti dell'applicazione, e per questo ho pianificato un tempo di sviluppo più breve.

#### 2.4.4 Documenti

Questo capitolo comprende la realizzazione del Gantt consuntivo, la scrittura della documentazione e la compilazione dei diari di lavoro. Si tratta di attività distribuite lungo tutto il progetto e aggiornate man mano che il lavoro andava avanti.

#### 2.5 Analisi dei mezzi

**HTML** (HyperText Markup Language), **CSS** (Cascading Style Sheets) **JavaScript**, **PHP (Hypertext Preprocessor)** e **MySQL(Structured Query Language)** sono le 5 tecnologie fondamentali per lo sviluppo di pagine e applicazioni web, per questo progetto le creerò tutte su Visual Studio Code

**HTML** è un linguaggio di markup pubblicato nel 1993 utilizzato per strutturare i contenuti di una pagina web, come titoli, paragrafi, immagini, pulsanti e form.

Fornisce la struttura base e definisce quali e dove saranno gli elementi nell'interfaccia utente.

**CSS** invece viene utilizzato per definire lo stile e l'aspetto grafico della pagina. Permette di modificare colori, dimensioni, layout, animazioni e transizioni, rendendo l'interfaccia visivamente gradevole e simile a un'app mobile.

**JavaScript** serve a rendere la pagina interattiva, consente di gestire eventi, modificare i contenuti, comunicare con file JSON, ecc... e implementare logiche di gioco complesse, come il conteggio delle vite oppure in questo caso il riconoscimento vocale.

**PHP** è un linguaggio di programmazione lato server, cioè un linguaggio che funziona sul server e serve per creare pagine web dinamiche.

Permette al sito web di pensare, decidere, salvare dati, controllare login, gestire utenti, e generare contenuti che cambiano in base a quello che fa l'utente.

**MySQL** è un Sistema di Gestione di Database Relazionali (RDBMS – Relational Database Management System) basato sul linguaggio SQL.

Viene utilizzato per creare, gestire e interrogare database strutturati, cioè insiemi di dati organizzati in tabelle collegate tra loro tramite relazioni.

Sviluppato inizialmente da MySQL AB e oggi mantenuto da Oracle, MySQL è uno dei database più popolari al mondo grazie alla sua velocità, stabilità, sicurezza e al fatto che è open-source.

Queste tecnologie insieme permettono di realizzare applicazioni web completamente funzionali, interattive e graficamente moderne, senza avere bisogno di un software aggiuntivo e visualizzabile su tutti i dispositivi dotati di rete.

Per ulteriori dettagli e approfondimenti, è consigliato consultare la documentazione ufficiale di [MDN Web Docs.](#)

## 2.5.1 Software

Scriverò i miei codici su **Visual Studio Code**, un'editor gratuito e open-source sviluppato da Microsoft nel 2015. Qui di seguito un po' di informazioni sul prodotto:

**Versione:** 1.78.2 (system setup)

**Data:** 2023-05-10T

**Elettronica:** 22.5.2

**OS:** Windows\_NT x64 10.0.22631

Per fare questo progetto ho utilizzato anche:

- Microsoft Word 16.0
- Microsoft Project 16.0
- XAMPP Control Panel v3.3.0
- MySQL 8.0
- Microsoft Excel 16.0

## 2.5.2 Hardware

Il prodotto può essere eseguito su qualsiasi piattaforma o dispositivo (PC, telefono, iPad) almeno che sia dispenso di un software per poterlo aprire.

Per la realizzazione di questo progetto ho utilizzato un pc scolastico con Windows 11 Education installato.

### HW computer Windows

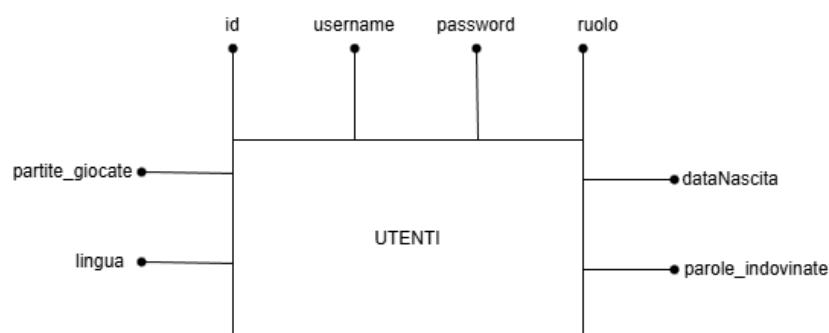
- Processore: 13th Gen Intel(R) Core(TM) i7-13700 2.10 GHz
- Scheda video: NVIDIA T400 4GB
- 32GB di RAM DDR4

### 3 Progettazione

#### 3.1 Design dei dati e database

Per il database ho scelto di utilizzare SQLite visto che l'utente deve avere i suoi dati salvati localmente sul dispositivo. In questo modo posso lavorare semplicemente su un file locale. Design delle interfacce

##### 3.1.1 Diagramma ER



Non c'è un vero e proprio database nel mio progetto, perché esso è formato solo da una tabella denominata utenti, con i suoi campi, dove verranno salvate le caratteristiche di ogni utente che si registrerà per poter giocare.

Field	Type	Null	Key	Default	Extra
<code>id</code>	<code>int(11)</code>	NO	PRI	<code>NULL</code>	<code>auto_increment</code>
<code>username</code>	<code>varchar(50)</code>	NO	UNI	<code>NULL</code>	
<code>password</code>	<code>varchar(255)</code>	NO		<code>NULL</code>	
<code>dataNascita</code>	<code>date</code>	NO		<code>NULL</code>	
<code>ruolo</code>	<code>varchar(20)</code>	YES		<code>giocatore</code>	
<code>parole_indotinate</code>	<code>text</code>	YES		<code>NULL</code>	
<code>partite_giocate</code>	<code>int(11)</code>	YES		<code>0</code>	
<code>lingua</code>	<code>varchar(5)</code>	YES		<code>it</code>	

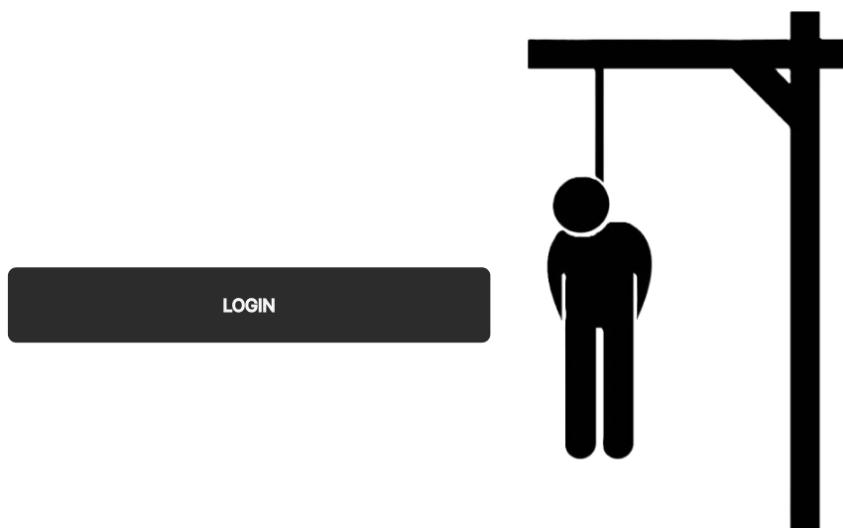
### 3.2 Design delle interfacce

La progettazione delle interfacce è basata sulle informazioni che ho ricavato durante la fase di analisi, le ho realizzate tramite il sito Figma.

Tutte i mockup sono stati pensati per l'utilizzo su dispositivi fissi, come i pc.

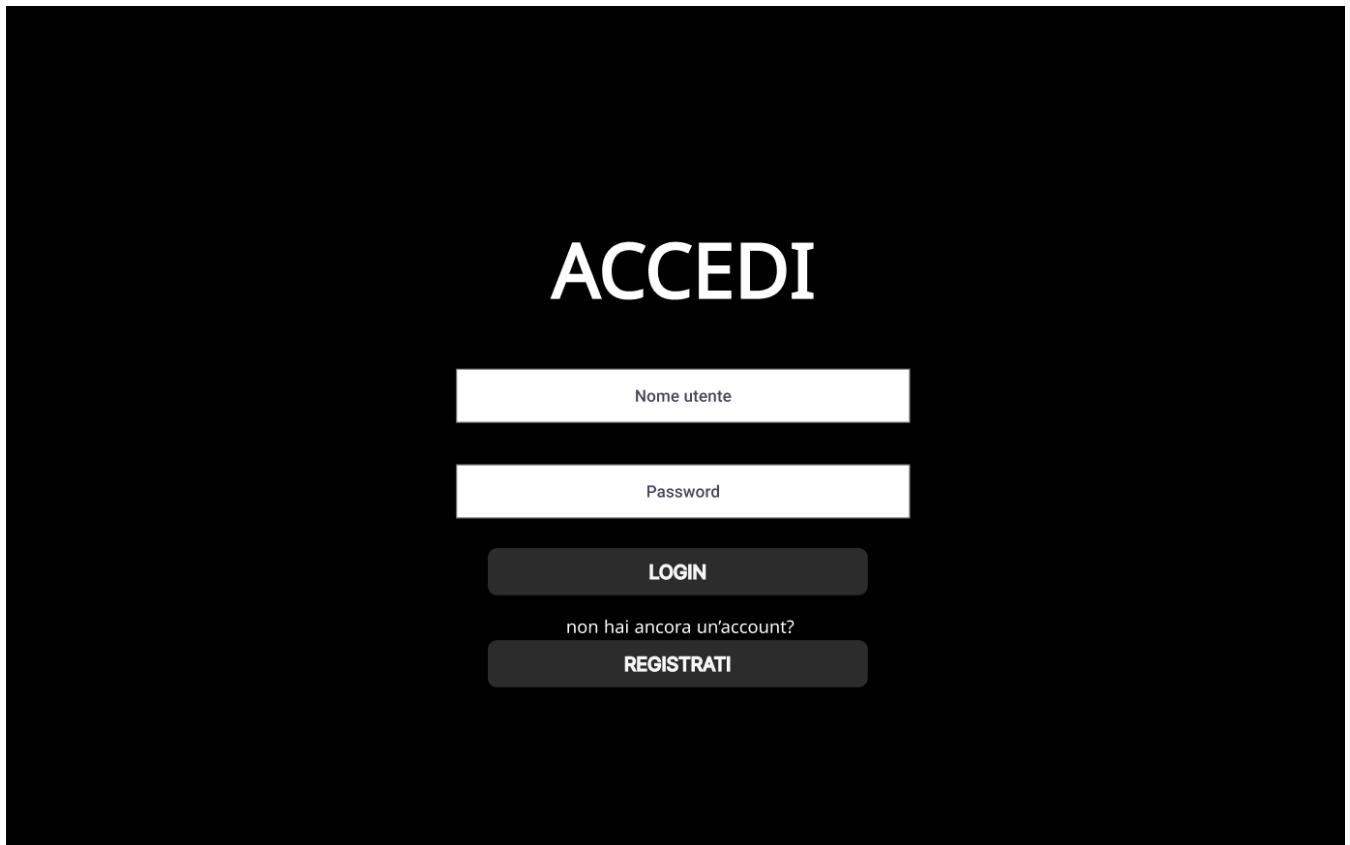
#### 3.2.1 Pagina Home

# HANG ZONE



Questa è l'interfaccia principale, essa è molto semplice infatti contiene solo 1 pulsante, ovvero quello di login, che porta direttamente alla pagina di login.

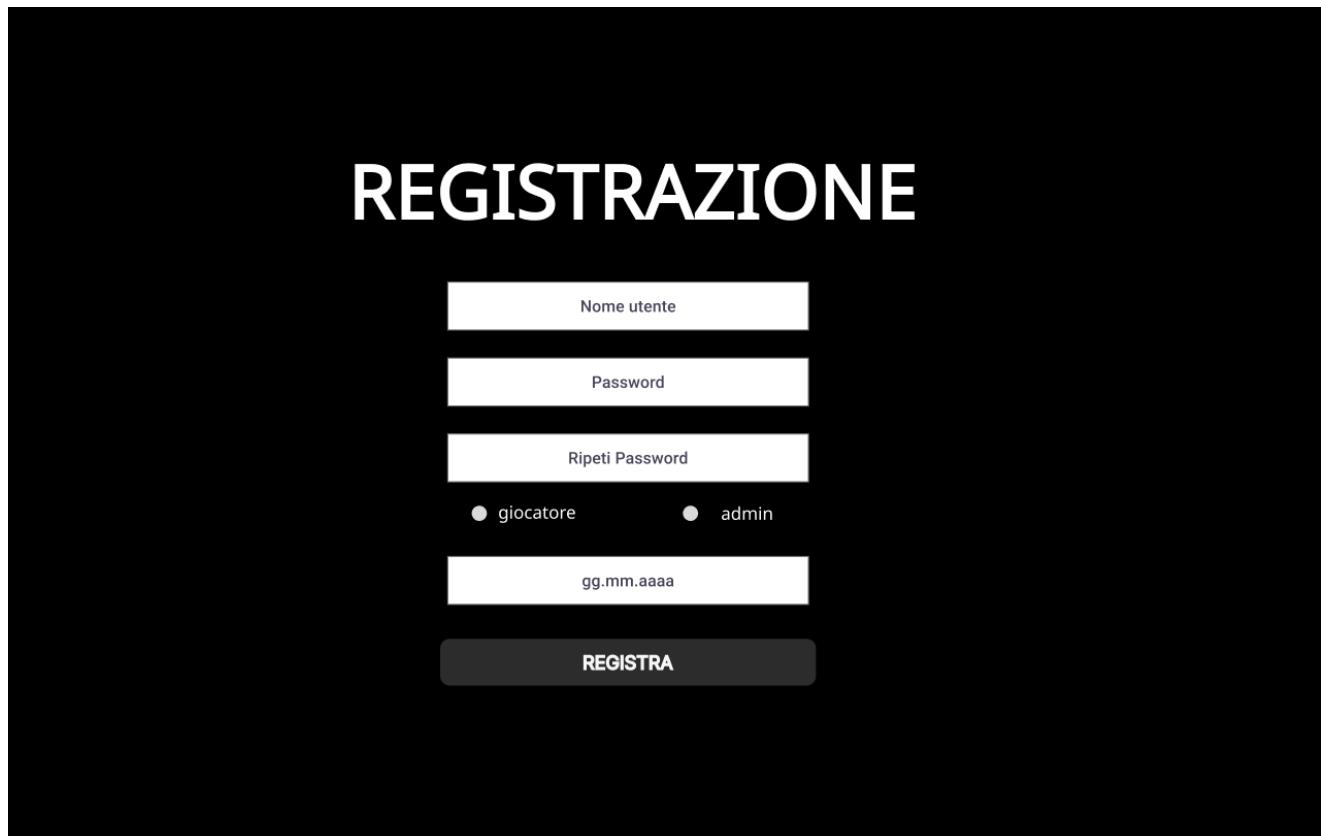
### 3.2.2 Pagina Login



La pagina di login è obbligatoria per poter accedere al gioco.

Se si ha già fatto una registrazione anche in passato, bisogna mettere Username e Password dell'account, e poi cliccare LOGIN, mentre se non si ha ancora un'account si deve cliccare REGISTRATI, e si verrà indirizzati alla pagina registrazione.html.

### 3.2.3 Pagina Registrazione



Nella pagina di registrazione, si ha la possibilità di creare l'account, inserendo nome utente, Password che sarà da ripetere 2 volte, poi bisognerà decidere se essere un giocatore oppure un amministratore. Mettere poi la data di nascita, e per accedere al gioco cliccare il bottone REGISTRA.

### 3.2.4 Pagina gioco

VocalHang

gestione parole

Scegli una lingua: italiano



nuova parola

Nella pagina principale di gioco possiamo notare una navbar con nome del gioco (VocalHang) a sinistra e un bottone gestione parole, che a dipendenza da che utente viene cliccato, se amministratore o giocatore, porta nella pagina per la gestione delle parole oppure in errore.html.

Sotto abbiamo il gioco vero e proprio con un menu a tendina per scegliere la lingua delle parole tra le 4 presenti, i cuori che indicano le vite di ogni giocatore, gli spazi e poi in seguito le lettere della parola da indovinare, e infine un bottone per generare una nuova parola.

### 3.2.5 Pagina Gestione parole

## GESTIONE PAROLE

Scegli una lingua: **italiano**

Casa



Banana



Cane



Javascript



Pittura



Hockey



Indietro

Invia

CREA

Nell'ultima pagina abbiamo la gestione delle parole, a cui possono accedere solo gli amministratori. In questa schermata vengono visualizzate in fila tutte le parole che ci sono nel file, a dipendenza di che cosa si sceglierà le parole verranno visualizzate in quella lingua. Poi nel loro campo ci saranno due immagini che cliccandole permetteranno di modificare la parola o eventualmente eliminarla, per aggiungere invece delle parole si potrà cliccare il button CREA, infine per aggiornare il file JSON si dovrà cliccare INVIA, infine per tornare al gioco con le parole aggiornate, si cliccherà INDIETRO.

### 3.2.6 Pagina di errore

per accedere a questa pagina devi essere

AMMINISTRATORE

[indietro](#)

Ci si imbatte in questa pagina quando si clicca il bottone **gestione parole** in gioca.html quando non si è amministratori ma solo giocatori.

In sostanza ti informa che non ci puoi entrare e ti invita a schiacciare il bottone per ritornare alla pagina di gioco principale.

## 4 Implementazione

In questo capitolo dovrà essere mostrato come è stato realizzato il lavoro. Questa parte può differenziarsi dalla progettazione in quanto il risultato ottenuto non per forza può essere come era stato progettato.

Sulla base di queste informazioni il lavoro svolto dovrà essere riproducibile.

In questa parte è richiesto l'inserimento di codice sorgente - Print Screen - di maschere solamente per quei passaggi particolarmente significativi e/o critici.

Inoltre, dovranno essere descritte eventuali varianti di soluzione o scelte di prodotti con motivazione delle scelte.

Non deve apparire nessuna forma di guida d'uso di librerie o di componenti utilizzati. Eventualmente questa va allegata.

Per eventuali dettagli si possono inserire riferimenti ai diari.

### Introduzione generale

Lo sviluppo del progetto **HangZone** è stato realizzato combinando tecnologie lato **client** (HTML, CSS, JavaScript) e tecnologie lato **server** (PHP e MySQL).

Qui di seguito vi illustrerò i punti che ritengo più importanti o eventualmente complicati del mio codice, in modo tale da poter essere compreso da tutti in futuro.

#### 1. Animazione titolo gioco pagina iniziale

```
.titolo-container {margin-top: 80px;text-align: center;color: #000000;font-size: 130px; }

.titolo-container div {display: inline-block;overflow: hidden;white-space: nowrap; }

.titolo-container div:first-of-type {animation: showup 7s forwards; }

.titolo-container div:last-of-type {width: 0px;animation: reveal 7s forwards; }

.titolo-container div:last-of-type span {margin-left: -800px;animation: slidein 7s forwards; }

@keyframes showup {
 0% {opacity:0; }
 20% {opacity:1; }
 80% {opacity:1; }
 100% {opacity:1; }}

@keyframes slidein {
 0% { margin-left:-900px; }
 20% { margin-left:-900px; }
 35% { margin-left:0px; }
 100% { margin-left:0px; }}

@keyframes reveal {
 0% {opacity:0;width:0px; }
 20% {opacity:1;width:0px; }
 30% {width:360px; }
 80% {opacity:1;width:360px; }
 100% {opacity:1;width:361px; }}
```

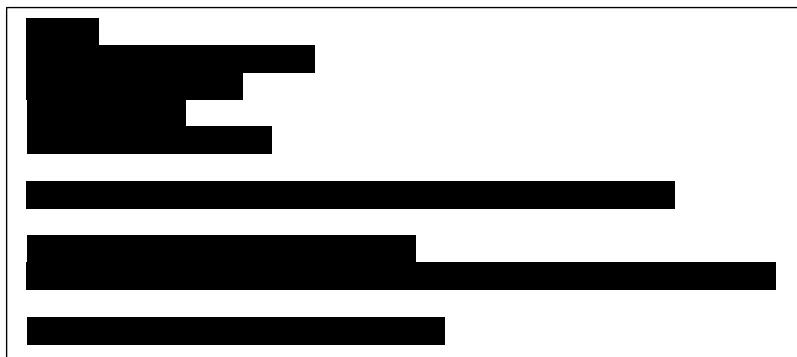
In index.html appena aperto vediamo che il titolo appare diviso in due parti perché ho scelto di animarle separatamente.

La prima metà viene fatta comparire gradualmente usando un'animazione che aumenta l'opacità, così da ottenere un effetto morbido. La seconda metà invece si anima con due movimenti: il contenitore si allarga piano piano e il testo dentro scorre da sinistra verso la sua posizione finale. Per ottenere questi effetti ho usato i keyframes, che permettono di definire passo dopo passo come devono cambiare opacità, posizione e larghezza durante l'animazione.

Unendo queste tre animazioni, il titolo entra in modo fluido interessante.

## 2. Connessione al database

La connessione al database è stata centralizzata nel file **connessione.php**, così da poter essere inclusa in tutti gli script senza dover ripetere le stesse righe in ogni file PHP, serve per connettersi con il database dove man mano si immetteranno tutti i dati dei diversi utenti che si registreranno.



Qui grazie host,user,password(vuota) e nome del database mi conetto a esso tramite l'uso della classe mysqli, faccio poi un breve controllo in caso di errore.

## 3. Login

Il sistema di login è composto da due parti principali:

- **Frontend (login.html)**, che gestisce l'interfaccia grafica e l'interazione con l'utente tramite JavaScript permette all'utente di inserire username e password e inviare i dati al server senza ricaricare la pagina
- **Backend (login.php)**, che verifica le credenziali nel database e in più gestisce la sessione dell'utente

Le due parti comunicano tra loro tramite una richiesta effettuata con fetch

### 3.1 Recupero degli elementi dal DOM

```
const togglePassword = document.getElementById('togglePassword');  
const inputPassword = document.getElementById('password');
```

In questa parte ho utilizzato il metodo **document.getElementById()**  
Serve per ottenere un riferimento a un elemento HTML partendo dal suo id

In JavaScript non si può modificare direttamente la pagina se prima non si recuperano gli elementi dal **DOM (Document Object Model)**

In questo caso ho recuperato:

- l'icona dell'occhio, che l'utente può cliccare
- il campo input della password

### 3.2 Mostra e nascondi la password

```
togglePassword.addEventListener('click', () => {
  const tipo = inputPassword.type === 'password' ? 'text' : 'password';
  inputPassword.type = tipo;
  togglePassword.src = tipo === 'text'
    ? 'img/occhio_aperto.jpg'
    : 'img/occhio_chiuso.jpg';});
```

Qui ho usato **addEventListener**, che mi ha permesso di associare un'azione ad un evento  
L'evento è il click sull'icona dell'occhio

Quando l'utente clicca:

- il codice controlla il tipo del campo password
- se il tipo è password, viene cambiato in text per rendere visibile la password (non crittografato)
- se è text, viene riportato a password (a puntini) per nasconderla
- allo stesso tempo viene cambiata l'immagine per far notare meglio se si sta guardando la password

### 3.3 Gestione dell'invio del form di login

```
document.getElementById("formLogin").addEventListener("submit", async (e) => {
  e.preventDefault();
```

Qui viene intercettato l'evento **submit** del form

Normalmente, quando un form viene inviato, il browser ricarica la pagina quindi ho impostato il metodo **preventDefault()** che serve a bloccare questo comportamento, che è molto fastidioso.

Questo permette di gestire il login senza ricaricare la pagina, rendendo così l'applicazione più moderna e anche fluida

La parola chiave **async** indica che all'interno della funzione verranno eseguite operazioni asincrone, come la richiesta fetch

---

### 3.4 Creazione dei dati da inviare

```
const formData = new FormData(e.target);
```

**FormData** è un oggetto JavaScript che raccoglie automaticamente tutti i dati del form

In questo modo non è necessario leggere manualmente ogni input: username e password vengono inclusi in automatico

Questo rende il codice più semplice, pulito e meno soggetto a errori di ogni tipo

### 3.5 Invio dei dati al server con fetch

```
const risposta = await fetch("login.php", {  
    method: "POST",  
    body: formData});
```

**Fetch** è un'API JavaScript che permette di **comunicare con il server** come apache

In questo caso invia i dati al file **login.php** usando il metodo POST

Ho scelto il metodo POST perché:

- non mostra i dati nella barra degli indirizzi
- è più sicuro per l'invio di credenziali
- è adatto a operazioni di autenticazione

Grazie ad **await**, il codice aspetta la risposta del server prima di continuare

### 3.6 Lettura della risposta del server

```
const esito = await risposta.text();
```

Questa istruzione legge la risposta del server come **testo**

Il backend restituisce stringhe come "ok" oppure "errore", che il frontend interpreta molto facilmente

### 3.7 Gestione dell'esito del login

```
if (esito === "ok") {  
    window.location.href = "gioca.php";  
} else {  
    document.getElementById("messaggioErrore").style.display = "block";}
```

Se il server restituisce "ok", significa che le credenziali sono corrette e l'utente viene reindirizzato alla pagina di gioco, a cui può accedere SOLO se si è autenticato correttamente

In caso contrario, viene mostrato il messaggio di errore senza ricaricare la pagina  
Questo fornisce un feedback immediato all'utente che può capire cosa ha fatto di sbagliato

#### 4. Implementazione Backend – login.php

Il file login.php gestisce l'autenticazione lato server  
Verificando le credenziali e creando la sessione dell'utente

##### 4.1 Avvio della sessione

```
session_start();
```

Questo comando avvia una sessione PHP  
Le sessioni servono per mantenere l'utente loggato mentre naviga tra le varie pagine del sito

##### 4.2 Connessione al database

```
include 'connessione.php';
```

Il file di connessione viene incluso separatamente per mantenere il codice ordinato e riutilizzabile

##### 4.3 Controllo del metodo POST

```
if ($_SERVER['REQUEST_METHOD'] === 'POST') {
```

Questo controllo garantisce che il file venga usato solo tramite richieste POST, evitando accessi non autorizzati

##### 4.4 Lettura dei dati inviati

```
$username = $_POST['username'];
$password = $_POST['password'];
```

Qui vengono recuperati i dati (username e password) inviati dal form di login

#### 4.5 Query preparata per il controllo utente

```
$stmt = $conn->prepare("SELECT id, password, ruolo FROM utenti WHERE username=?");  
$stmt->bind_param("s", $username);
```

La query cerca nel database un utente con uno specifico **username**, recuperando **id**, **password** e **ruolo** in modo sicuro tramite una query preparata.

#### 4.6 Verifica della password

```
if (password_verify($password, $hash)) {
```

La funzione `password_verify` confronta la password inserita con l'hash salvato nel database. In questo modo le password non vengono mai salvate in chiaro, aumentando la sicurezza del sistema.

#### 4.7 📁 Creazione della sessione utente

```
$_SESSION['utente'] = [  
    "id" => $id,  
    "username" => $username,  
    "ruolo" => $ruolo,];
```

Se il login è corretto, i dati principali dell'utente vengono salvati nella sessione. Queste informazioni verranno usate nelle altre pagine per:

- riconoscere l'utente
- gestire i permessi in base al ruolo

### 5. Implementazione del sistema di Registrazione

Il sistema di registrazione è composto da due parti principali:

- **Frontend (`registrazione.html`)**, che gestisce l'interfaccia grafica, i controlli lato client e l'invio dei dati
- **Backend (`registrazione.php`)**, che valida i dati, controlla il database e registra l'utente in modo sicuro

Le due parti comunicano tra loro tramite una richiesta asincrona usando `fetch`

## 5.1 Frontend – registrazione.html

La pagina registrazione.html permette all'utente di creare un nuovo account inserendo i propri dati personali

Il JavaScript serve a **controllare i dati prima dell'invio**, migliorare l'usabilità e gestire la risposta del server

## 5.2 Recupero degli elementi dal DOM

```
const form = document.getElementById('formRegistrazione');
const messaggioErrore = document.getElementById('messaggioErrore');
const messaggioSuccesso = document.getElementById('messaggioSuccesso');
const togglePassword = document.getElementById('togglePassword');
const inputPassword = document.getElementById('password');
```

In questa sezione viene utilizzato `document.getElementById()`, un metodo JavaScript che consente di recuperare elementi HTML dalla pagina tramite il loro id

Questo passaggio è fondamentale perché:

- JavaScript non può modificare la pagina senza accedere al DOM
- ogni variabile rappresenta un elemento preciso (form, messaggi, input)
- il codice diventa più leggibile e facile da mantenere

## 5.3 Mostra / Nascondi password

```
togglePassword.addEventListener('click', () => {
  const tipo = inputPassword.type === 'password' ? 'text' : 'password';
  inputPassword.type = tipo;
  togglePassword.src = tipo === 'text'
    ? 'img/occhio_aperto.jpg'
    : 'img/occhio_chiuso.jpg';});
```

Qui ho deciso di usare `addEventListener`, che permette di associare una funzione a un evento  
L'evento in questo caso è il click sull'icona dell'occhio

Il codice:

- controlla il tipo del campo password
- alterna tra password e text
- aggiorna l'immagine dell'icona per dare un feedback visivo all'utente

Questa scelta migliora l'usabilità perché consente all'utente di verificare la password senza inserire nuovi pulsanti o campi

#### 5.4 Gestione dell'invio del form

```
form.addEventListener('submit', async (e) => {  
    e.preventDefault();
```

Qui viene intercettato l'evento **submit** del form

Normalmente un form ricarica la pagina, ma **e.preventDefault()** blocca questo comportamento

Questo permette di:

- gestire la registrazione senza refresh
- rendere l'esperienza più fluida
- usare comunicazione asincrona con il server

La parola chiave **async** indica che verranno eseguite operazioni asincrone come **fetch**

#### 5.5 Reset dei messaggi

```
messaggioErrore.style.display = 'none';  
messaggioSuccesso.style.display = 'none';
```

Prima di effettuare nuovi controlli, i messaggi vengono nascosti

Questo evita di mostrare errori o successi precedenti che non sono più validi

#### 5.6 Lettura dei dati inseriti

```
const username = document.getElementById('username').value.trim();  
const password = document.getElementById('password').value.trim();  
const password2 = document.getElementById('password2').value.trim();  
const ruolo = document.getElementById('ruolo').value;  
const dataNascita = document.getElementById('dataNascita').value;
```

In questa parte vengono recuperati i valori inseriti dall'utente

Il metodo **.trim()** serve a rimuovere spazi inutili prima o dopo il testo

Questo riduce errori e migliora l'affidabilità dei controlli

## 5.7 Controllo campi vuoti

```
if (!username || !password || !password2 || !ruolo || !dataNascita) {  
    messaggioErrore.textContent = "Compila tutti i campi";  
    messaggioErrore.style.display = 'block';  
    return;}
```

Questo controllo verifica che tutti i campi siano compilati

Se anche uno solo è vuoto:

- viene mostrato un messaggio di errore
- l'invio dei dati viene bloccato con return

Questo evita richieste inutili al server

## 5.8 Controllo password uguali

```
if (password !== password2) {  
    messaggioErrore.textContent = "Le password non coincidono";  
    messaggioErrore.style.display = 'block';  
    return;
```

Qui ho verificato che le due password inserite siano identiche  
Questo previene errori di registrazione e migliora la sicurezza

## 5.9 Invio dei dati al server con fetch

```
const formData = new FormData(form);  
const risposta = await fetch('registrazione.php', {  
    method: 'POST',  
    body: formData});}
```

FormData raccoglie automaticamente tutti i dati del form  
fetch invia questi dati al file registrazione.php usando il metodo POST

Questa scelta:

- è più sicura del metodo GET
- non mostra i dati nell'URL
- permette comunicazione asincrona

## 5.10 Lettura e gestione risposta del server

```
const esito = await risposta.text();
```

La risposta del server viene letta come testo  
Il backend restituisce stringhe semplici (ok, username\_esistente, ecc...)

## 5.11 Gestione degli esiti

```
if (esito === 'ok') {
    messaggioSuccesso.style.display = 'block';
    setTimeout(() => {
        window.location.href = 'login.html';
    }, 1000);}
```

Se la registrazione è riuscita:

- viene mostrato un messaggio di successo
- dopo 1 secondo l'utente viene reindirizzato alla pagina di login

Questo consente all'utente di vedere il messaggio prima del cambio pagina  
Negli altri casi vengono mostrati messaggi di errore più specifici

## 6. Implementazione Backend – registrazione.php

Il file `registrazione.php` gestisce tutta la logica lato server, garantendo sicurezza e integrità dei dati

### 6.1 Connessione al database

```
include 'connessione.php';
```

Il file di connessione è separato per mantenere il codice ordinato e riutilizzabile

### 6.2 Controllo metodo POST

```
if ($ SERVER['REQUEST METHOD'] === 'POST') {
```

Questo controllo assicura che lo script venga eseguito solo tramite richieste POST

### 6.3 Controllo campi vuoti

Questo controllo serve come **secondo livello di sicurezza**, anche se esiste già lato client  
Il doppio controllo (client + server) è una buona pratica fondamentale

### 6.4 Controllo password

```
if ($password !== $password2) {  
    echo "errore_password";  
    exit;  
}
```

Dice se le due password sono uguali o meno  
Se sono diverse printa un'errore

### 6.5 Controllo ruolo valido

```
$ruoli_validi = ['giocatore', 'admin'];
```

Questo impedisce l'inserimento manuale di ruoli non autorizzati (diversi da giocatore e admin)

### 6.6 Crittografia della password

```
$password_hash = password_hash($password, PASSWORD_DEFAULT);
```

La password viene salvata **solo in forma criptata**  
Questo è fondamentale per la sicurezza del database

## 7. Gioca.php

### 7.1 Gestione della sessione e sicurezza iniziale

All'inizio del file PHP uso `session_start()` perché mi serve accedere ai dati dell'utente loggato.

Subito dopo controllo se la sessione contiene l'utente:

```
if (!isset($_SESSION['utente'])) {  
    header("Location: login.html");  
    exit;}
```

Questo controllo serve a **proteggere la pagina**: se uno non è loggato, non può giocare e viene rimandato al login. È una sicurezza base ma fondamentale

## 7.2 Recupero ruolo e username dell'utente

Poi salvo il ruolo e l'username in variabili PHP:

```
$ruolo = $_SESSION['utente']['ruolo'];
$username = $_SESSION['utente']['username'];
```

Lo faccio perché il ruolo mi serve dopo in JavaScript per decidere cosa l'utente può fare

## 7.3 Passaggio dei dati dal PHP al JavaScript

Nel JavaScript prendo il ruolo direttamente dal PHP:

```
const ruoloUtente = "<?php echo $ruolo; ?>";
```

Questo serve a passare informazioni dal server al client, in questo modo JavaScript sa se l'utente è admin o no

## 7.4 Controllo accesso alle funzionalità admin

Quando l'utente clicca il bottone “gestione parole” controllo il ruolo:

```
btnGestione.addEventListener('click', () => {
    if(ruoloUtente === "admin") {
        window.location.href = 'gestione.php';
    } else {
        window.location.href = 'errore.html';
    });
});
```

Qui ho fatto così perché non tutti devono poter gestire le parole: solo l'admin

Se non lo è, viene bloccato

## 7.5 Variabili globali del gioco

Per il gioco dichiaro alcune variabili globali:

```
let paroleData = null, parolaCorrente = "", lettereScoperte = [], vite = 0;
```

Mi servono per tenere in memoria i dati del gioco: la lista delle parole, la parola scelta, quali lettere sono state indovinate e quante vite restano

## 7.6 Caricamento parole dal file JSON

La funzione **inizializzaGioco** carica le parole dal file JSON:

```
const risposta = await fetch('./file/parole.json');
paroleData = await risposta.json();
```

Uso `fetch` perché i dati arrivano da un file esterno. Se il caricamento fallisce, imposto dei dati di emergenza così il gioco non si rompe:

```
paroleData = { viteIniziali:6, it:['prova','fallback'] };
```

Alla fine avvio subito una nuova partita:

```
nuovaPartita();
```

## 7.7 Inizializzazione di una nuova partita

La funzione `nuovaPartita` serve a **resettere il gioco**:

```
parolaCorrente = listaParole[Math.floor(Math.random() * listaParole.length)].toLowerCase();
```

Poi scelgo una parola casuale.

Preparo l'array che tiene traccia delle lettere indovinate

Uso questo sistema perché mi permette di sapere **lettera per lettera** cosa mostrare a schermo

## 7.8 Aggiornamento della parola a schermo

La funzione `aggiornaDisplay` serve solo a mostrare la parola:

```
displayParola.textContent = Array.from(parolaCorrente)
.map((ch,i)=>lettereScoperte[i]?.toUpperCase():'_')
.join(' ');
```

Se una lettera è stata indovinata la mostro, altrimenti metto `_`. È separata dalle altre funzioni per tenere il codice ordinato

### 7.9 Gestione delle vite e dei cuori

Per le vite aggiorno sia il numero che i cuori:

```
function aggiornaVite(){ viteRimanenti.textContent = vite; }
function aggiornaCuori(){
    cuoriContainer.innerHTML = '';
    for(let i=0;i<viteTotali;i++){
        const cuore = document.createElement('span');
        if(i >= vite) cuore.classList.add('vuoto');
        cuoriContainer.appendChild(cuore);}}
```

Ho fatto così perché il numero e la grafica sono due cose diverse ma dipendono dallo stesso valore

### 7.10 Gestione del tentativo dell'utente

Quando l'utente preme una lettera, viene chiamata tentativoLettera:

```
if(parolaCorrente[i] === lettera){
    lettereScoperte[i] = true;
    trovata = true;}
```

Se la lettera non c'è, tolgo una vita:

```
if(!trovata){ vite--; }
```

Dopo ogni tentativo aggiorno tutto e controllo se la partita è finita

### 7.11 Controllo vittoria o sconfitta

La funzione controllaEsito verifica vittoria o sconfitta:

```
if(lettereScoperte.every(v => v === true)){
    messaggio.textContent='hai vinto!';
}

if(vite <= 0){
    messaggio.textContent='hai perso!';}
```

Ho separato questo controllo per non mischiare la logica del gioco con l'input dell'utente

## 7.12 Controllo input da tastiera

Infine ascolto la tastiera:

Così l'utente può giocare **senza cliccare**, solo scrivendo, rendendo il gioco più naturale

```
document.addEventListener('keydown', (e)=>{
    if(e.key >= 'a' && e.key <= 'z') tentativoLettera(e.key);
});
```

## 8 Gestione.html

### 8.1 Controllo ruolo lato server

Poi faccio un secondo controllo, questa volta sul **ruolo**:

```
if ($_SESSION['utente']['ruolo'] !== "admin") {
    echo "Accesso negato: non hai i permessi per entrare in questa pagina.";
    exit;}
```

Qui ho scelto di controllare il ruolo lato server perché è la parte **più sicura**: anche se qualcuno modifica il JavaScript, il PHP non lo farà entrare comunque

### 8.2 Recupero elementi principali della pagina

Nel JavaScript inizio prendendo gli elementi principali della pagina:

```
const lista = document.getElementById('lista-parole');
const selectLingua = document.getElementById('lingua');
let paroleJSON = {};
```

**paroleJSON** è un oggetto che conterrà tutte le parole lette dal file JSON, divise per lingua. Lo tengo globale perché viene usato da più funzioni

### 8.3 Rendering dinamico della lista parole

La funzione `renderParole` serve a **disegnare dinamicamente la lista delle parole**:

```
function renderParole() {
    lista.innerHTML = '';
    const lingua = selectLingua.value;
    if (!paroleJSON[lingua]) return;
```

Svuoto sempre la lista prima di ridisegnarla, così evito duplicati. Creo poi un elemento per ogni parola:

```
const div = document.createElement('div');
div.className = 'parola-item';
```

#### 8.4 Modifica diretta delle parole

Dentro ogni riga inserisco un input:

```
const input = document.createElement('input');
input.type = 'text';
input.value = parola;
```

Uso un input perché l'admin deve poter **modificare direttamente** la parola.

#### 8.5 Icone di modifica ed eliminazione

Per rendere l'interfaccia più intuitiva aggiungo due icone: modifica ed elimina

L'icona modifica mette solo il focus sull'input:

```
imgMod.onclick = () => input.focus();
```

Quella elimina rimuove la parola dall'array e aggiorna la lista:

```
imgDel.onclick = () => {
imgDel.onclick = () => {
    paroleJSON[lingua].splice(index, 1);
    renderParole();
```

Ho fatto così per mantenere **sincronizzati dati e interfaccia**

#### 8.6 Cambio lingua dinamico

Quando cambio lingua dal select, ricarico le parole:

```
selectLingua.addEventListener('change', renderParole);
```

In questo modo la lista si aggiorna automaticamente senza ricaricare la pagina.

## 8.7 Creazione di nuove parole

Il bottone “Crea” aggiunge una nuova parola vuota:

```
paroleJSON[lingua].push('');  
renderParole();
```

Subito dopo metto il cursore sull’ultimo input:

```
inputs[inputs.length - 1].focus();
```

Così l’admin può iniziare subito a scrivere senza clic extra

## 8.8 Salvataggio delle parole modificate

Il bottone “Salva” legge tutti gli input e ricostruisce l’array:

```
paroleJSON[lingua] = Array.from(inputs)  
  .map(i => i.value.trim())  
  .filter(i => i);  
};
```

Uso `trim()` per togliere spazi inutili e `filter` per evitare parole vuote

Alla fine mostro un messaggio e stampo il JSON in console:

```
alert('JSON aggiornato! Controlla console.');//  
console.log(paroleJSON);
```

Questo serve come **conferma visiva** e per debug

## 8.9 Caricamento iniziale del file JSON

Infine carico il file JSON all’avvio:

```
fetch('./file/parole.json')  
  .then(response => response.json())  
  .then(data => {  
    paroleJSON = data;  
    renderParole();  
  });
```

Uso `fetch` perché il file è esterno e voglio separare i dati dal codice  
Appena i dati arrivano, li salvo e disegno subito la lista

## 8. Test

### 8.1 Protocollo di test

Ho deciso di eseguire i test manuali per questo perché ritenevo poco utili farli automatizzati, date le funzioni basi che sono presenti nel sito HangZone.

I test verificano il corretto comportamento del sistema rispetto ai requisiti funzionali definiti in precedenza.

ID Test: T-01	
Requisito testato:	Requisito 1 – Schermata home
Descrizione test	Verificare che all'avvio dell'applicazione venga visualizzata la schermata home con il pulsante LOGIN
Precondizioni	<ul style="list-style-type: none"><li>- Applicazione avviata correttamente</li></ul>
Passi da seguire	<ol style="list-style-type: none"><li>1. Avviare l'applicazione</li><li>2. Osservare la schermata iniziale</li><li>3. Vedere se c'è un pulsante con scritto LOGIN</li></ol>
Risultato atteso	Viene visualizzata la schermata home contenente il pulsante LOGIN
Esito	<input checked="" type="checkbox"/> Superato <input type="checkbox"/> Non superato
Note	

ID Test: T-02	
Requisito testato:	Requisito 2 – Avvio gioco
Descrizione test	Verificare che il sistema selezioni una parola dal file JSON e la visualizzi sotto forma di trattini
Precondizioni	<ul style="list-style-type: none"><li>- Requisito 1 funzionante</li><li>- Login effettuato correttamente</li></ul>
Passi da seguire	<ol style="list-style-type: none"><li>1. Effettuare il login</li><li>2. Avviare una nuova partita</li><li>3. Vedere se le parole che escono sono prese dal nostro file JSON</li></ol>
Risultato atteso	La parola viene scelta dal file JSON e mostrata con trattini al posto delle lettere
Esito	<input checked="" type="checkbox"/> Superato <input type="checkbox"/> Non superato
Note	

**ID Test: T-03****Requisito testato:** Requisito 3 – Inserimento lettere scritte**Descrizione test** Verificare che l'utente possa inserire lettere tramite una casella di testo**Precondizioni**

- Gioco avviato correttamente.

**Passi da seguire**

1. Digitare una lettera sulla tastiera
2. Vedere se viene inserita o se, è sbagliata, scritta sotto

**Risultato atteso** La lettera viene accettata dal sistema e inserita**Esito** Superato Non superato**Note****ID Test: T-04****Requisito testato:** Requisito 4 – Gestione vite**Descrizione test** Verificare che il numero di vite diminuisca in caso di lettera errata e che il gioco termini a vite finite**Precondizioni**

- Gioco avviato correttamente

**Passi da seguire**

1. Inserire una lettera errata.
2. Ripetere fino a esaurimento delle vite.

**Risultato atteso** Ogni lettera errata riduce le vite di una unità (cuore)  
A zero vite il gioco termina e bisogna generare una nuova parola**Esito** Superato Non superato**Note**

**ID Test:** T-05

**Requisito testato:** Requisito 5 – Login

**Descrizione test** Verificare che l'utente debba effettuare il login per poter giocare

**Precondizioni**

- Applicazione avviata
- Account già esistente

**Passi da seguire**

1. Tentare di avviare il gioco senza login.
2. Effettuare il login.
3. Avviare il gioco.

**Risultato atteso**

Senza login il gioco non è accessibile  
Dopo il login il gioco si avvia correttamente

**Esito**

Superato

Non superato

**Note**

**ID Test:** T-06

**Requisito testato:** Requisito 6 – Fine gioco

**Descrizione test** Verificare che il gioco termini quando le vite finiscono

**Precondizioni**

- Gioco avviato correttamente
- Gestione vite funzionante

**Passi da seguire**

1. Inserire lettere errate fino a esaurire tutte le vite
2. Osservare il comportamento del sistema

**Risultato atteso**

Il gioco termina ed è necessario generare una nuova parola

**Esito**

Superato

Non superato

**Note**

**ID Test:** T-07

**Requisito testato:** Requisito 8 – Cambio lingua

**Descrizione test** Verificare che sia possibile cambiare la lingua delle parole

**Precondizioni**

- Applicazione avviata correttamente e funzionante

**Passi da seguire**

1. Selezionare una lingua diversa
3. Osservare se le parole che appaiono sono cambiate

**Risultato atteso** La lingua dell'esperienza e delle parole viene modificata correttamente

**Esito**

Superato

Non superato

**Note**

**ID Test:** T-07

**Requisito testato:** Requisito 9 – Visualizzazione parole

**Descrizione test** Verificare che l'amministratore possa visualizzare tutte le parole presenti nel file JSON

**Precondizioni**

- Login effettuato come amministratore

**Passi da seguire**

1. Accedere all'area amministratore
2. Visualizzare l'elenco delle parole e vedere se ci sono tutte

**Risultato atteso** Tutte le parole presenti nel file JSON vengono mostrate correttamente

**Esito**

Superato

Non superato

**Note**

**ID Test:** T-09

**Requisito testato:** Requisito 9 – Inserimento parole

**Descrizione test** Verificare che l'amministratore possa aggiungere nuove parole al gioco

**Precondizioni**

- Login amministratore effettuato
- Visualizzazione parole funzionante

**Passi da seguire**

1. Accedere all'interfaccia di inserimento parole (gestione parole)
2. Inserire una nuova parola
3. Salvare

**Risultato atteso**

La nuova parola viene aggiunta al file JSON ed è disponibile nel gioco

**Esito**

Superato

Non superato

**Note**

**ID Test:** T-010

**Requisito testato:** Requisito 10 – Modifica parole

**Descrizione test** Verificare che l'amministratore possa modificare una parola esistente

**Precondizioni**

- Login amministratore effettuato
- Elenco parole visibile

**Passi da seguire**

1. Selezionare una parola esistente
2. Modificarla
3. Salvare le modifiche

**Risultato atteso**

La parola viene modificata correttamente nel sistema

**Esito**

Superato

Non superato

**Note**

ID Test: T-010	
<b>Requisito testato:</b>	Requisito 11 – Eliminazione parole
<b>Descrizione test</b>	Verificare che l'amministratore possa eliminare una parola dal gioco
<b>Precondizioni</b>	<ul style="list-style-type: none"><li>- Login amministratore effettuato</li><li>- Elenco parole visibile</li></ul>
<b>Passi da seguire</b>	<ol style="list-style-type: none"><li>1. Scegliere una parola</li><li>2. Eliminarla tramite l'interfaccia</li></ol>
<b>Risultato atteso</b>	La parola viene rimossa dal file JSON e non è più utilizzabile nel gioco
<b>Esito</b>	<input checked="" type="checkbox"/> Superato <input type="checkbox"/> Non superato
<b>Note</b>	

## 8.2 Mancanze/limitazioni conosciute

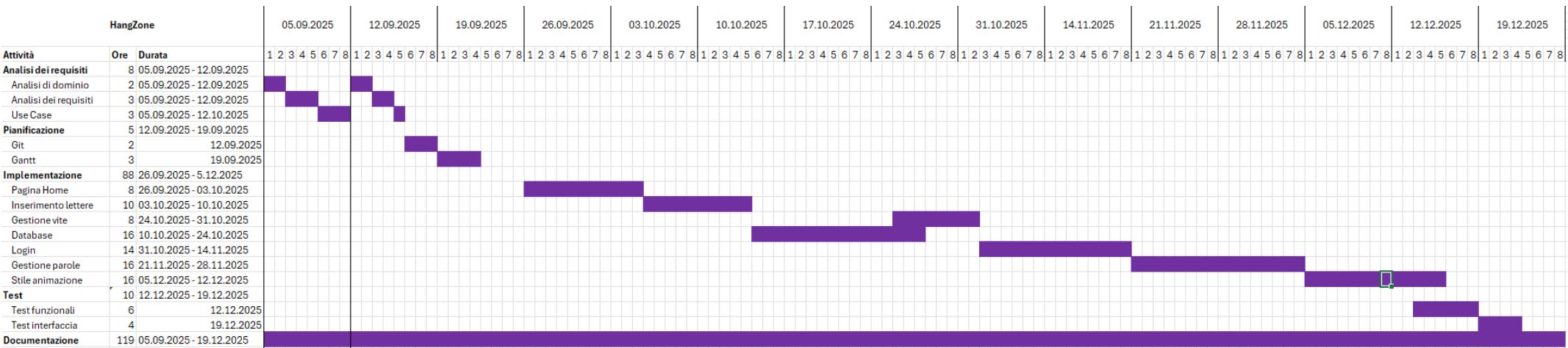
Ci sono un paio di cose che purtroppo non sono riuscita ad implementare nel mio progetto.  
Io ero partita come idea di base per questo progetto di fare applicazione web per il gioco dell'impiccato dove le parole erano inseriti tramite la voce, quindi parlando.

Quando sono arrivata al momento di testare il codice, il riconoscimento vocale (Speech-to-Text), implementato in Javascript con l'API Web Speech, non riusciva, anche facendo uso di un microfono professionale a capire e interpretare le singole lettere.

Ho dovuto quindi scartare l'idea, e trovare un modo alternativo per immettere le lettere, ovvero la semplice inserzione tramite tastiera.

A quel punto il progetto mi sembrava troppo semplice, quindi ho deciso di aggiungere il login e la registrazione dove gli utenti registrati e i propri dati vengono salvati in un database quindi ricordati nel tempo.

## 9. Consuntivo



All'inizio del progetto avevo pianificato le attività con un Gantt preventivo, ma durante lo sviluppo reale alcune cose sono andate diversamente.

Alcune fasi hanno richiesto più tempo del previsto, soprattutto lo sviluppo delle funzionalità principali, mentre altre sono state fatte in parallelo

Per esempio, l'analisi dei requisiti non è rimasta solo all'inizio, ma è continuata anche durante il lavoro perché alcune scelte sono state chiarite meglio strada facendo.

Le attività nelle diverse pagine web, avevo immaginato fossero più corte di quanto effettivamente sono stato, a parer mio è capitato perché sono alle prime armi, ho ancora fatto troppi pochi progetti da quando sono in questa scuola e come è normale che sia, non sono ancora in grado di capire quanto tempo impiegherà a fare una singola attività.

Sicuramente il fatto di aver trovato un'impossibilità nel poter utilizzare il riconoscimento vocale, mi ha parecchio rallentato e demotivato. Sono comunque riuscita a concludere il mio progetto entro il tempo stabilito, cosa fondamentale nei progetti.

Nel complesso quindi il progetto è andato bene e ha raggiunto gli obiettivi principali, anche se non ho seguito perfettamente la pianificazione iniziale. Questa esperienza mi ha fatto capire che nella pratica è spesso necessario adattare il piano di lavoro mentre si sviluppa il progetto.

## 10. Conclusioni

Con questo progetto sono riuscita a creare un'applicazione funzionante che rispetta i requisiti iniziali e che mette insieme diversi linguaggi e argomenti che ho studiato, come PHP, JavaScript, JSON e la gestione degli utenti.

Sicuramente non è un progetto che cambierà il mondo, in tutta sincerità dopo la consegna non sarà utilizzato da nessuno, neanche come gioco di svago, ma è stato molto utile per capire meglio come funziona un'applicazione web completa.

La soluzione realizzata ha un impatto soprattutto a livello personale, a livello scolastico: mi ha aiutato a capire come collegare il lato server con il lato client, come gestire i ruoli degli utenti e come organizzare il codice in modo più ordinato, queste cose mi hanno molto aiutato anche per le altre materie dove tratto questi argomenti (come php per esempio, dove la mia nota è molto migliorata dopo aver capito la connessione al server, tramite il progetto).

Considero il mio progetto un discreto risultato, perché non è solo una prova teorica ma qualcosa che funziona davvero. Le soluzioni utilizzate non valgono solo per questo gioco, ma le potrò riutilizzare anche in altri progetti simili, ad esempio per la gestione degli utenti o dei dati.

In conclusione, questo lavoro non è stato affatto una perdita di tempo o un lavoro imposto: mi ha permesso di fare pratica, di sbagliare e correggere errori, e di migliorare il mio modo di programmare, essendo che è il nostro primo progetto che definisco serio, sono riuscita anche a capire cosa aspettarmi dall'anno prossimo e soprattutto come deve essere fatta una documentazione professionale.

È stato un passo importante per capire se questo percorso mi interessa davvero e per prepararmi a progetti più complessi in futuro.

### 10.1 Sviluppi futuri

Ci sono diverse migliorie che vorrei applicare al mio progetto, perché in futuro potrebbe diventare un'applicazioni web interessante nel mondo dei giochi online.

Prima di tutto vorrei trovare un modo (sempre se esistente) di poter immettere il riconoscimento vocale delle lettere che si vogliono provare ad aggiungere per creare la parola, perché la ritenevo un'aggiunta al semplice gioco dell'impiccato molto interessante, per poter permettere a tutti quanti di giocarci, chi ha impedimenti nello scrivere per esempio.

Inoltre vorrei salvare le parole in un database, non in un semplice e molto poco sicuro file JSON, ho scelto di salvarle li quando ho fatto il progetto, perché non volevo addentrarmi in ambienti che non conosco essendo che ero già in ritardo rispetto alla tabella di marcia che mi ero imposta.

Sicuramente poi sarebbe molto utile, salvare le singole partite e tenere nel database degli utenti, i risultati, cosicché ogni utente possa vedere quante parole ha indovinato, quanto tempo ci ha messo, in quale lingua si trova meglio, ecc..

Queste sono certamente delle cose che farò se dovesse capitarmi di riprendere in mano questo progetto in futuro, in un momento di noia o forse per un altro progetto.

## 10.2 Considerazioni personali

Durante lo sviluppo di questo progetto ho capito meglio come funziona il lavoro su un progetto professionale e non solo su singoli esercizi. All'inizio alcune parti mi sono risultate difficili, soprattutto organizzare bene il lavoro e stimare i tempi, ma andando avanti ho imparato a gestire meglio le varie fasi.

La parte più difficile è stata l'implementazione di alcune funzionalità, perché spesso ho dovuto correggere errori o rivedere il codice già scritto. Allo stesso tempo, questo mi ha aiutato a migliorare la mia capacità di risolvere i problemi e a capire meglio come funzionano le cose nella pratica, cosa che come detto prima mi ha molto aiutato in questo semestre scolastico, per capire meglio i concetti usandoli non studiandoli.

Sono soddisfatta del risultato finale, anche se so che il progetto può essere molto migliorato, ha diverse piccole imperfezioni che ad un occhio attento potrebbero dare fastidio. Se dovessi rifarlo, dedicherei più tempo alla fase di analisi iniziale e alla pianificazione, in modo da evitare alcuni problemi durante lo sviluppo. In generale questo progetto è stato utile perché mi ha fatto fare esperienza e mi ha aiutato a capire meglio come affrontare un lavoro più strutturato come quello che potrei fare in un'azienda in futuro.

## 11. Glossario

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

Esempio:

Termine	Descrizione
AJAX	<b>Asynchronous JavaScript And XML:</b> una tecnica che permette di eseguire richieste ed ottenere dati da una pagina web in modo asincrono.
CSS	<b>Cascading Style Sheets:</b> linguaggio che permette di definire il layout e la grafica di una pagina web.
API	<b>Application Programming Interface:</b> insieme di funzioni e procedure che permettono a un software di comunicare con un altro
DOM	<b>Document Object Model:</b> rappresentazione della struttura HTML di una pagina che permette di manipolare elementi tramite JavaScript
ER (Diagramma ER)	<b>Entity-Relationship:</b> diagramma che rappresenta visivamente le tabelle di un database e le relazioni tra di esse
Frontend	Parte di un'applicazione web che viene eseguita sul client, cioè nel browser dell'utente
POST	<b>Metodo HTTP</b> per inviare dati al server in modo sicuro, senza mostrarli nell'URL
SQLite	Database relazionale leggero che salva i dati in un singolo file locale
Backend	Parte di un'applicazione web che viene eseguita sul server e gestisce logica, database e sicurezza
FormData	Oggetto JavaScript che raccoglie automaticamente i dati di un form per inviarli al server
MDN Web Docs	Risorsa ufficiale di Mozilla per documentazione su HTML, CSS, JavaScript e altre tecnologie web
RDBMS	<b>Relational Database Management System:</b> sistema per gestire database relazionali

## **12. Bibliografia**

---

### **12.1 Sitografia**

- [developer.mozilla](https://developer.mozilla.org) -> MDN Web Docs
- [php](https://www.php.net) -> PHP.net
- [dev.mysql](https://dev.mysql.com/doc/) -> MySQL Documentation
- [w3schools](https://www.w3schools.com) -> W3School
- [stackoverflow](https://stackoverflow.com) -> StackOverflow
- [youtube](https://www.youtube.com) -> Video tutorial per la connessione al database