DattiloKing

1 Introduzione 3

1.1 Informazioni sul progetto 3

1.2 Abstract 3

1.3 Scopo 3

2 Analisi 4

2.1 Analisi del dominio 4

2.2 Analisi e specifica dei requisiti 4

2.3 Use case 9

2.4 Pianificazione 10

2.5 Analisi dei mezzi 11

2.5.1 Software 11

2.5.2 Hardware 11

3 Progettazione 12

3.1 Design dell’architettura del sistema 12

3.2 Design dei dati e database 13

3.3 Design delle interfacce 14

3.4 Design procedurale 21

3.4.1 Diagramma delle classi 21

3.4.2 Diagramma di flusso 22

4 Implementazione 24

4.1 Struttura e ambiente del progetto 24

4.2 Generazione frasi 25

4.2.1 Python per generare le frasi 25

4.2.2 Formattazione SQL 26

4.3 Implementazione Database 27

4.4 Implementazione SignUp 28

4.5 Implementazione validazione SignUp 30

4.6 Implementazione SignIn 31

4.7 Validazione SignIn 32

4.8 Implementazione recupero frase 33

4.9 Gestione degli eventi 34

4.10 Implementazione Stampa del testo 35

4.11 Implementazione salvataggio statistiche 36

4.12 Implementazione visualizzazione statistiche 38

4.13 Audio 40

4.14 Delete User 42

5 Test 43

5.1 Protocollo di test 43

5.2 Risultati test 47

5.3 Mancanze/limitazioni conosciute 47

6 Consuntivo 47

7 Conclusioni 47

7.1 Sviluppi futuri 47

7.2 Considerazioni personali 48

7.2.1 Amos Haefliger 48

7.2.2 Nemanja Zecevic 48

7.2.3 Robin Sartore 48

7.2.4 Leonardo Sciara 48

8 Indice delle Figure 49

9 Glossario 49

10 Bibliografia 50

10.1 Bibliografia per articoli di riviste: 50

10.2 Bibliografia per libri 50

10.3 Sitografia 50

11 Allegati 50

# Introduzione

## Informazioni sul progetto

* Allievi:
  + Amos Haefliger
  + Leonardo Sciara
  + Nemanja Zecevic
  + Robin Sartore
* Docente: Michel Palucci
* Scuola: Scuola Arti e Mestieri di Trevano
* Sezione: Informatica
* Materia: M306
* Data di inizio: 29.01.2025
* Data di consegna: 28.05.2025

## Abstract

**DA ELIMINARE UNA VOLTA SCRITTO L’ABSTRACT**

È una breve e accurata rappresentazione dei contenuti di un documento, senza notazioni critiche o valutazioni. Lo scopo di un abstract efficace dovrebbe essere quello di far conoscere all’utente il contenuto di base di un documento e metterlo nella condizione di decidere se risponde ai suoi interessi e se è opportuno il ricorso al documento originale.

Può contenere alcuni o tutti gli elementi seguenti:

* **Background/Situazione iniziale**
* **Descrizione del problema e motivazione**: Che problema ho cercato di risolvere? Questa sezione dovrebbe includere l'importanza del vostro lavoro, la difficoltà dell'area e l'effetto che potrebbe avere se portato a termine con successo.
* **Approccio/Metodi**: Come ho ottenuto dei progressi? Come ho risolto il problema (tecniche…)? Quale è stata l’entità del mio lavoro? Che fattori importanti controllo, ignoro o misuro?
* **Risultati**: Quale è la risposta? Quali sono i risultati? Quanto è più veloce, più sicuro, più economico o in qualche altro aspetto migliore di altri prodotti/soluzioni?

Esempio di abstract:

*As the size and complexity of today’s most modern computer chips increase, new techniques must be developed to effectively design and create Very Large-Scale Integration chips quickly. For this project, a new type of hardware compiler is created. This hardware compiler will read a C++ program, and physically design a suitable microprocessor intended for running that specific program. With this new and powerful compiler, it is possible to design anything from a small adder, to a microprocessor with millions of transistors. Designing new computer chips, such as the Pentium 4, can require dozens of engineers and months of time. With the help of this compiler, a single person could design such a large-scale microprocessor in just weeks.*

## Scopo

Lo scopo del nostro sito web “DattiloKing” è quello di migliorare la propria scrittura da tastiera in modo didattico e simpatico. Si avrà la possibilità di giocare in single player per migliorare la propria abilità in modo autonomo e di selezionare la modalità multiplayer per poter sfidare i propri amici nello scrivere la frase nel modo più veloce, avendo poi la possibilità di vedere il proprio score e quello dei nostri amici.

Questo sito è adatto a chiunque si interfacci con un Pc, indipendentemente dall’età.

Questo progetto permette al nostro team di migliorare le proprie skill con i linguaggi che utilizzeremo per costruire il sito web nonché HTML, BOOTSTRAP, JS, PHP, MYSQL.

Grazie al progetto miglioreremo anche le nostre abilità di lavorare in gruppo, di sperimentare e approfondire la metodologia di progetto Agile.

Con questo progetto puntiamo a rendere divertente e stimolante l’apprendimento e il miglioramento a scrivere con la tastiera. Grazie alla possibilità di sfidare i propri amici puntiamo a una voglia di superarsi e di superare i nostri amici che migliorerà notevolmente la nostra velocità nella scrittura da tastiera.

# Analisi

## Analisi del dominio

Il nostro sito web sarà utilizzabile sui più famosi sistemi operativi quali Windows, MacOs e tutte le distro di Linux da un qualsiasi browser. Può venir utilizzato da chiunque, e da qualsiasi fascia di età (a patto che sappia utilizzare la tastiera). Chiunque abbia il bisogno di migliorare le proprie skill dattili può utilizzare DattiloKing per migliorare e per sfidare i propri amici migliorando assieme. Sul web ci sono numerosi siti web che permettono di migliorare la propria dattilografia quali “dattilografia-online” o “agilefingers” ma con DattiloKing abbiamo implementato un’interfaccia innovativa rendendo unico il nostro sito web.

Naturalmente non sono necessarie skill, conoscenze o competenze particolari per utilizzare il sito web, un completo principiante che utilizza per la prima volta il suo dispositivo desktop o laptop che sia può utilizzare DattiloKing per migliorare le proprie skill e imparare a digitare da tastiera in modo più rapido.

## Analisi e specifica dei requisiti

|  |  |
| --- | --- |
| **ID: REQ-001** | |
| **Nome** | Visualizzazione testo |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | Visualizzazione della frase nel gioco Singleplayer e Multiplayer |
| **Sotto requisiti** | |
| **001** | Si necessita dell’interfaccia di gioco |
| **002** | Si necessita del DB |

|  |  |
| --- | --- |
| **ID: REQ-002** | |
| **Nome** | Visualizzazione punteggi Multiplayer |
| **Priorità** | 3 |
| **Versione** | 1.0 |
| **Note** | Visualizzazione della classifica e punteggi dei vari utenti |
| **Sotto requisiti** | |
| **001** | Si necessita dell’interfaccia di gioco |
| **002** | Si necessità del DB |

|  |  |
| --- | --- |
| **ID: REQ-003** | |
| **Nome** | Percentuale correttezza |
| **Priorità** | 3 |
| **Versione** | 1.0 |
| **Note** | Calcolo della percentuale delle lettere scritte correttamente |
| **Sotto requisiti** | |
| **001** | Si necessita dell’interfaccia di gioco |
| **002** | Si necessità del DB |

|  |  |
| --- | --- |
| **ID: REQ-004** | |
| **Nome** | Velocità di scrittura |
| **Priorità** | 3 |
| **Versione** | 1.0 |
| **Note** | Calcolo della velocità media in cui si scrive una frase |
| **Sotto requisiti** | |
| **001** | Si necessita dell’interfaccia di gioco |
| **002** | Si necessità del DB |

|  |  |
| --- | --- |
| **ID: REQ-005** | |
| **Nome** | Tempo di scrittura |
| **Priorità** | 3 |
| **Versione** | 1.0 |
| **Note** | Calcolo del tempo in cui viene finita una frase |
| **Sotto requisiti** | |
| **001** | Si necessita dell’interfaccia di gioco |
| **002** | Si necessità del DB |

|  |  |
| --- | --- |
| **ID: REQ-006** | |
| **Nome** | Visualizzazione e animazione grafica della scrittura |
| **Priorità** | 2 |
| **Versione** | 1.0 |
| **Note** | Mostra la tastiera e l’animazione degli input dell’utente |
| **Sotto requisiti** | |
| **001** | Si necessita dell’interfaccia di gioco |

|  |  |
| --- | --- |
| **ID: REQ-007** | |
| **Nome** | Selezione della lingua |
| **Priorità** | 4 |
| **Versione** | 1.0 |
| **Note** | L’utente ha la possibilità di scegliere la lingua |
| **Sotto requisiti** | |
| **001** | Si necessita dell’interfaccia di gioco |
| **002** | Si necessità del DB |

|  |  |
| --- | --- |
| **ID: REQ-008** | |
| **Nome** | Selezione tema di sfondo |
| **Priorità** | 5 |
| **Versione** | 1.0 |
| **Note** | L’utente ha la possibilità di selezionare un tema di sfondo |
| **Sotto requisiti** | |
| **001** | Si necessita dell’interfaccia di gioco |

|  |  |
| --- | --- |
| **ID: REQ-009** | |
| **Nome** | Multiplayer |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | Permette di giocare da più dispositivi simultaneamente |
| **Sotto requisiti** | |
| **001** | Si necessita dell’interfaccia di gioco |
| **002** | Si necessità del DB |

|  |  |
| --- | --- |
| **ID: REQ-010** | |
| **Nome** | Gestione audio |
| **Priorità** | 4 |
| **Versione** | 1.0 |
| **Note** | L’utente ha la possibilità di abilitare o disattivare l’audio |
| **Sotto requisiti** | |
| **001** | Si necessita dell’interfaccia di gioco |

|  |  |
| --- | --- |
| **ID: REQ-011** | |
| **Nome** | Visualizzazione punteggi Singleplayer |
| **Priorità** | 3 |
| **Versione** | 1.0 |
| **Note** | Visualizza i punteggi giornalieri e di tutti i tempi dell’utente |
| **Sotto requisiti** | |
| **001** | Si necessita dell’interfaccia di gioco |
| **002** | Si necessità del DB |

|  |  |
| --- | --- |
| **ID: REQ-012** | |
| **Nome** | Scelta turni multiplayer |
| **Priorità** | 2 |
| **Versione** | 1.0 |
| **Note** | Creatore della stanza multiplayer sceglie il numero di round |
| **Sotto requisiti** | |
| **001** | Si necessita dell’interfaccia di gioco |

|  |  |
| --- | --- |
| **ID: REQ-013** | |
| **Nome** | Cancellazione utente |
| **Priorità** | 2 |
| **Versione** | 1.0 |
| **Note** | L’utente ha la possibilità di cancellare l’utente loggato |
| **Sotto requisiti** | |
| **001** | Si necessita dell’interfaccia di gioco |
| **002** | Si necessità del DB |

|  |  |
| --- | --- |
| **ID: REQ-014** | |
| **Nome** | Controllo battitura |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | Esegue controlli nella battitura |
| **Sotto requisiti** | |
| **001** | Si necessita dell’interfaccia di gioco |
| **002** | Si necessità del DB |

## Use case



Figura 1 Schema Use Case

Questo è lo Use Case del progetto DattiloKing. L’utente può gestire diversi parametri e registrarsi per poter visualizzare le proprie statistiche in modo accurato e accedere al multiplayer.

## Pianificazione



Figura 2 Schema del Gantt

Questo è il Gantt che abbiamo fatto per la pianificazione del nostro progetto, in totale abbiamo 90 ore di lavoro già un paio sono andate per la teoria generale poi abbiamo progettato e iniziato il primo sprint che ha un po’ di task. Poi ci sono altri 5 sprint in cui abbiamo pianificato delle task provvisorie che modificheremo prima di iniziare lo sprint. E infine due milestone e la consegna finale

## Analisi dei mezzi

### Software

Il software include:

* GitHub Desktop: strumento per la gestione di repository Git
* PhpStorm 2024.1.1: IDE per lo sviluppo
* MySQL Workbench 8.0 CE: Strumento per la gestione di database MySQL
* Google Chorme: Browser utilizzato per la navigazione
* FireFox: Browser utilizzato per la navigazione
* Microsoft Edge: Browser utilizzato per la navigazione
* Word: Software per l’elaborazione della documentazione
* Project: Software per l’elaborazione dei Gantt
* Visio: Software per l’elaborazione degli allegati
* Blocco note: Editor per appunti
* Strumento di cattura: Software per la cattura delle schermate

### Hardware

L’hardware include:

* PC di scuola per la realizzazione:
  + CPU: 13th Gen Intel(R) Core(TM) i7-13700
  + GPU: Intel(R) UHD Graphics 770
  + RAM: 32GB
  + SSD: 512GB
* PC di casa sempre per la realizzazione, per poter continuare in caso di ore saltate a lezione

# Progettazione

## Design dell’architettura del sistema

Il nostro sistema delle architetture non è molto grande dato che abbiamo solo un DB e il WebServer con l’applicativo, però abbiamo fatto uno schema per spiegarlo.



Figura 3 Schema delle strutture di sistema

L’utente una volta digitato l’url del nostro applicativo riceverà le view dal WebServer che si situa nel server Infomaniak. Poi l’utente può registrarsi o accedere grazie al database con tutti gli utenti registratati.

## C:\Users\leonardo.sciara\AppData\Local\Packages\Microsoft.Windows.Photos_8wekyb3d8bbwe\TempState\ShareServiceTempFolder\SchemaE-R.jpegDesign dei dati e database

Figura 4 Schema E-R

Ho creato il database di nome dattiloking con quattro tabelle al suo interno, una tabella **“stanza”** la quale ha i campi codice, il quale rappresenta un codice univoco il quale serve ad accedere alla stanza del multiplayer, **“vincitore”** il quale ha salvato il nickname del vincitore della stanza attuale e **“numeroTurni”** il quale indica il numero di turni che la stanza ha avuto.

La tabella **“frase”** contiene il **“testo”** delle frasi che devono apparire quando ci si esercita e quando si gioca alla modalità multiplayer, la **“lingua”** e la **“lunghezza”**.

Inoltre, c’è anche un’altra tabella: **“turno\_storico”,** la quale contiene i dati di tutti i turni dei giocatori, per quanto riguarda la giornata stessa e lo storico totale. Infine è presente una tabella **“utente”** la quale salva lo **“username”** il quale deve essere univoco e la **“password”** degli utenti una volta che si registrano.

## Design delle interfacce

Descrizione delle interfacce interne ed esterne del sistema e dell’interfaccia utente. La progettazione delle interfacce è basata sulle informazioni ricavate durante la fase di analisi e realizzata tramite mockups.



Figura 5 HomePage senza essere loggati



Figura 6 Settings

Queste due interfacce rappresentano la pagina home di DattiloKing, l’utente avrà la possibilità di allenarsi decidendo se ascoltare la musica del gioco e in che lingua verranno proposte le frasi, tutto questo senza essersi loggato, e quindi senza avere la possibilità di poter leggere i punteggi totalizzati nelle varie frasi.



Figura 7 Form di Login



Figura 8 Form di SignUp

Le interfacce soprastanti rappresentano la pagina di gestione dell’account, dove l’utente può registrarsi nella parte di **“Sign Up”** inserendo **“Username”** e **“Password”**. Una volta registrato, l’utente, con quello stesso **“Username”** e **“Password”**, avrà la possibilità di accedere.

Le 2 interfacce che seguono rappresentano la pagina home di DattiloKing, però sta volta, con l’utente loggato. La pagina ha le stesse funzionalità che con l’utente non loggato, con l’aggiunta però, della possibilità di eliminare l’account, la possibilità di fare il **“Log Out”** per poter accedere con un altro utente, la possibilità di visualizzare le statistiche di tutti i tempi e giornaliere dell’utente loggato e come ultimo, l’aggiunta fondamentale, il **“Multy Player”:** L’utente, grazie a questa aggiunta, avrà possibilità di affrontare altri giocatori e vedere chi è il migliore tra loro.



Figura 9 HomePage da loggati



Figura 10 Settings da loggato

La seguente interfaccia rappresenta la pagina principale di DattiloKing, una volta premuto sul tasto **“Single** **player”**, presente nella home, l’utente verrà portato nella zona di Singleplayer, dove verrà illustrata una frase sullo schermo. Per completare la seguente frase e iniziare a esercitarti, verrà proposta una tastiera sotto lo schermo che lo aiuterà nell’intento di completarla e di imparare a scrivere. Durante la scrittura della frase, ogni lettera non premuta correttamente conterà come errore e la frase non continuerà finché l’utente non avrà scritto la lettera corretta. Dopo aver completato la frase, ne apparirà subito un’altra da completare e in caso l’utente non voglia di continuare la frase, gli basterà premere sul logo della casa per tornare alla pagina di home. Per l’utente non loggato a ogni completamento della frase potrà osservare il punteggio a schermo senza però, essere salvato nel database, invece, per quello loggato, ogni frase completata avrà un punteggio dettagliato da poter visualizzare direttamente a schermo che poi verrà salvato nel database per fare una media con gli altri punteggi già totalizzati in passato. Al ritorno della pagina home, durante la frase in corso, non verranno presi in considerazione i punteggi per eseguire delle medie.



Figura 11 Keyboard di gioco singleplayer

La seguente interfaccia rappresenta i punteggi dell’utente loggato. Una volta fatto l’accesso con un utente, esso potrà osservare, se presenti, i punteggi del **“Single player”** che ha totalizzato nel corso della sua creazione e di quelli giornalieri, premendo soltanto sull’icona della persona nella pagina di home.



Figura 12 Statistiche personali

L’interfaccia che segue rappresenta la pagina **“Multi player”** che viene abilitata solo agli utenti che hanno eseguito l’accesso. Questa modalità può essere selezionata solo dalla pagina home. Una volta che l’utente è entrato, può scegliere se entrare in una stanza oppure di crearla.



Figura 13 Multiplayer

Le 2 seguenti sono: La pagina per poter entrare nella stanza, dove l’utente dovrà inserire il codice per poter unirsi, dove una volta unito, potrà vedere i partecipanti e il numero di round che il creatore della stanza sta scegliendo; la pagina per poter creare la stanza, dove viene illustrato il codice per poter far entrare li altri giocatori, la possibilità di scegliere il numero di round, i giocatori presenti nella stanza e la possibilità di far partire il gioco.



Figura 14 Creazione stanza



Figura 15 Inserimento codice, unione stanza

Questa interfaccia rappresenta l’avvio del gioco, che ha più o a meno le stesse funzionalità della modalità **“Single player”**, soltanto, che una volta finita la frase, si dovrà aspettare che tutti li utenti connessi alla stanza abbiano finito la stessa frase per poi partire tutti assieme con un’altra. La stanza ha un numero limitato di round scelti dal creatore della stanza e che una volta finiti i round, si potranno osservare i risultati.

Figura 16 Keyboard di gioco multiplayer

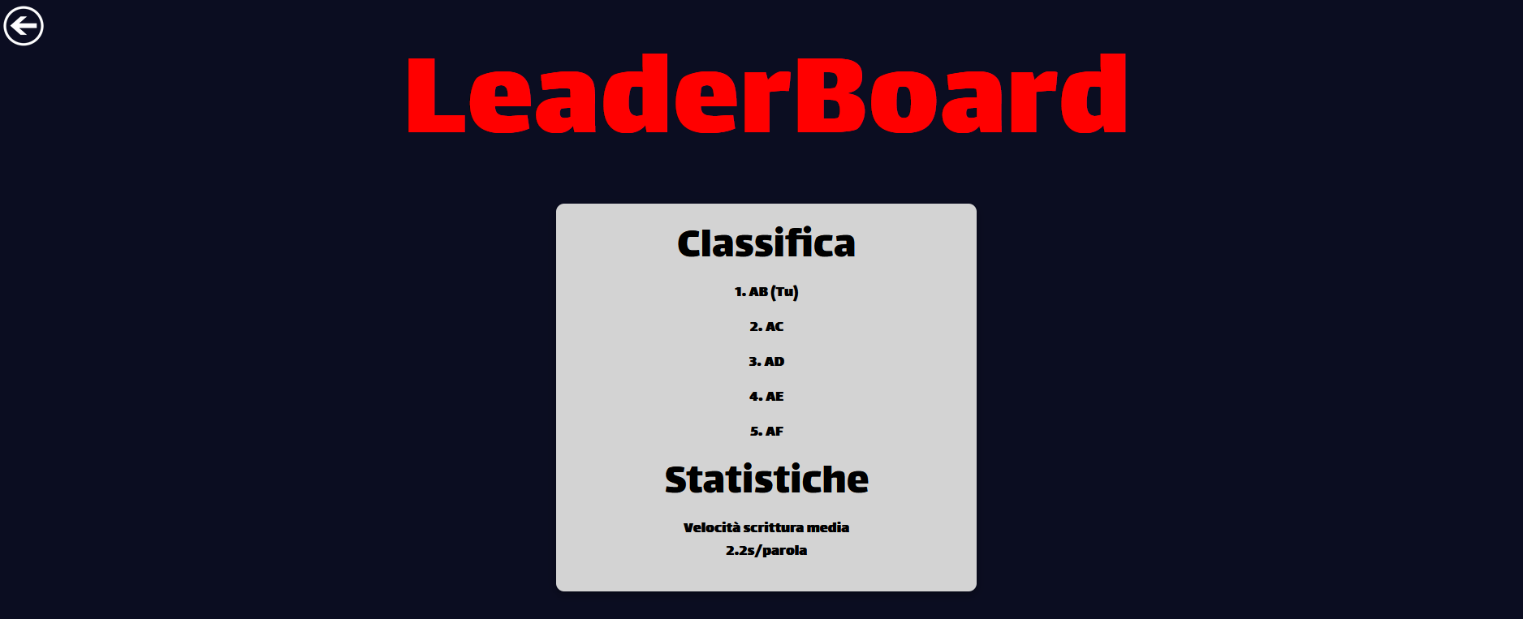
L’ultima interfaccia è la seguente, essa rappresenta la classifica del la modalità “Multi player”. Dopo aver finito i round prestabiliti dal creatore della stanza, si potrà osservare la classifica di tutti i giocatori presenti nella stanza con la medesima velocità media di scrittura.

Figura 17 Leaderboard multiplayer

## Design procedurale

Descrive i concetti dettagliati dello sviluppo:

* Diagramma delle classi
* Diagrammi di flusso

### Diagramma delle classi



Figura 18 Diagramma delle classi

Questo è lo schema delle nostre classi. Abbiamo una classe per le impostazioni in cui si potrà scegliere il volume e la lingua, ci sarà una classe per fare il login o la registrazione e un’altra per verificare l'accesso e creare un nuovo utente.

Ci sarà poi una classe Database che utilizzeranno tutte le classi che devono connettersi al database, come la classe training in cui si gestirà la partita singola e la classe Multiplayer room per la modalità multiplayer, si avrà anche una classe, Leaderoard per salvare la classifica finale della modalità multiplayer e poi la modalità singleplayer c'è la classe scores con le varie statistiche.

In più ci sarà la classe keyboards in cui l'utente vedrà la tastiera e l’input dei pulsanti che schiaccerà mentre la frase da scrivere scorrerà in avanti.

### Diagramma di flusso

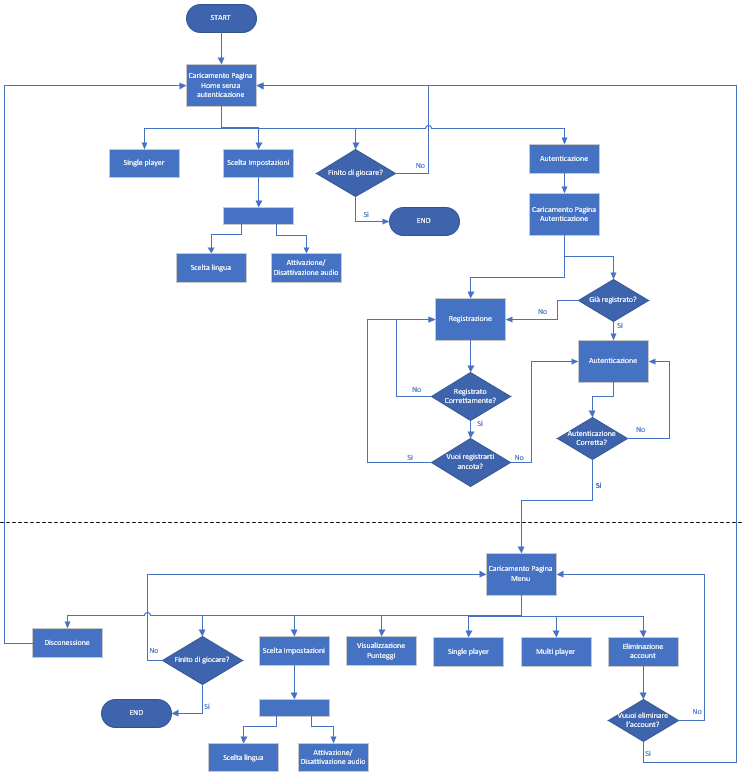


Figura 19 Diagramma di flusso generale

Il soprastante diagramma rappresenta il flusso di navigazione del gioco, distinguendo tra utenti autenticati e non autenticati.

* Avvio dell’applicazione:
  + il sistema carica la pagina home senza l’autenticazione.
  + L’utente può scegliere tra:
    - Single player
    - Scelta impostazioni (audio e lingua)
    - Autenticazione
* Autenticazione dell’utente:
  + L’utente può registrarsi o autenticarsi.
    - Se non registrato, deve completare una registrazione
    - Se già registrato, può registrare altri utenti o effettuare l’accesso
    - Se la registrazione o l’autenticazione falliscono, il processo deve essere ripetuto
  + Dopo aver effettuato l’accesso, viene caricato la pagina della home, dove l’utente può scegliere tra:
    - Single player
    - Multiplayer
    - Visualizzazione punteggi
    - Scelta impostazioni
    - Eliminazione account
* Chiusura della sessione:
  + L’utente può decidere di terminare la sessione in un qualsiasi momento

In allegato si trova lo schema completo: **ADDattiloKing.vsdx**

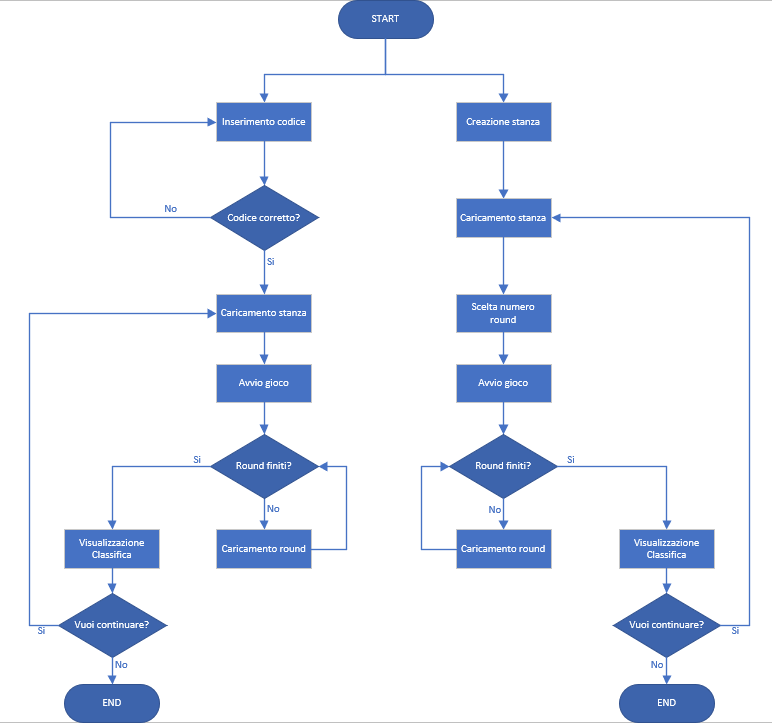


Figura 20 Diagramma di flusso Multiplayer

Il soprastante diagramma rappresenta il flusso di gioco della modalità multiplayer, suddiviso tra l’ingresso in una stanza esistente e la creazione di una nuova stanza.

* Ingresso in una stanza esistente:
  + L’utente deve inserire un codice per accedere.
  + Il sistema verifica se il codice inserito è corretto:
    - Se corretto, si procede con il caricamento della stanza e in attesa dell’avvio del gioco
    - Se sbagliato, l’utente dovrà inserire un novo codice
* Creazione di una nuova stanza:
  + Il creatore crea una stanza, imposta il numero di round e decide quando avviare il gioco
  + Dopo aver impostato i round, viene creata una stanza
  + Il cre
* Fasi del gioco:
  + Dopo l’avvio, dopo la fine di una frase, il sistema verifica se i round sono terminati
  + Se i round non sono finiti, viene caricato il round successivo
  + Se i round sono terminati, il sistema mostra la classifica finale
* Conclusione:
  + Il giocatore sceglie se continuare a giocare o uscire dalla modalità **“Multiplayer”**

In allegato si trova lo schema completo: **ADDattiloKingMultiplayer.vsdx**

# Implementazione

## Struttura e ambiente del progetto

Il progetto, essendo un'applicazione web, è stato sviluppato sfruttando le conoscenze acquisite durante il nostro percorso scolastico. Abbiamo scelto di utilizzare PHP e di adottare una struttura MVC (Model-View-Controller) per organizzare il codice in modo efficiente e modulare.

La struttura del progetto segue il modello classico MVC, con le seguenti componenti:

**Models**: contengono la logica di business e interagiscono con il database.

**Views**: gestiscono la presentazione dei dati all'utente.

**Controllers**: elaborano le richieste dell'utente e coordinano le interazioni tra models e views.

Inoltre, abbiamo incluso:

**Config**: per la definizione delle configurazioni di base e delle costanti.

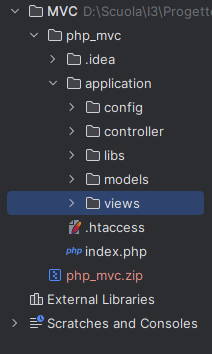
**Libs**: una directory in cui è possibile inserire librerie esterne o classi helper per estendere le funzionalità dell'applicazione

Figura 21 Struttura cartelle

## Generazione frasi

Per il nostro progetto avevamo bisogno erano le frasi che gli utenti dovevano scrivere, farle a mano avrebbe richiesto troppo tempo dato che ci sarebbero state tre lingue e avevamo bisogno di una vasta quantità di frasi. La prima opzione è stata farle generare a un’intelligenza artificiale ma dopo svariati tentativi con varie AI gratuite non siamo riusciti a ottenere delle frasi con lo stesso numero di caratteri, questo è dovuto al modo di ragionare alle AI che non gli permettono di contare con precisione i caratteri.

Alla fine abbiamo deciso di creare un programma Python aiutandoci con le AI per fargli generare le frasi.

### Python per generare le frasi

Per prima cosa abbiamo trovato dei file txt contenenti tutte le parole nelle varie lingue, il codice prende una parola di questo txt e la inserisce nell’array contenente la frase poi aggiunge uno spazio e ne mette un'altra, ogni volta che aggiunge una nuova parola controlla se questa non fa sforare il limite di 200 caratteri se sì la cambia. Per la parte finale cerca direttamente una parola che faccia arrivare la frase a 200 caratteri.

Infine salva tutte le frasi generate su un txt.

# Controlla se l'aggiunta della parola supera la lunghezza massima

if lunghezza\_corrente + len(parola) + spazio > lunghezza:

break # Se la parola non entra, esce dal ciclo

frase.append(parola)  # Aggiunge la parola alla frase

lunghezza\_corrente += len(parola) + spazio # Aggiorna la lunghezza corrente

# Se la frase è più corta della lunghezza desiderata, cerca una parola finale che la completi

caratteri\_mancanti = lunghezza - lunghezza\_corrente

parole\_finali = [p for p in parole if len(p) == caratteri\_mancanti]

# Trova parole esatte

if parole\_finali:

frase.append(random.choice(parole\_finali))

# Aggiunge una parola esatta se esiste

return ' '.join(frase)  # Concatena le parole con uno spazio

# Salva le frasi in un file di testo

output\_file\_path = 'frasi\_generate\_italiano.txt'

with open(output\_file\_path, 'w', encoding='utf-8') as output\_file:

for i, frase in enumerate(frasi\_generate, 1):

output\_file.write(f"Frase {i}: '{frase}' con {len(frase)} caratteri.\n")

print(f"Frasi generate salvate in '{output\_file\_path}'.")

else:

    print("Non è stato possibile generare frasi perché l'elenco di parole è vuoto.")

### Formattazione SQL

Dopo aver generato le frasi con il programma python abbiamo dovuto creare un altro programma che crei il codice SQL da implementare nel DB. La tabella frasi a tre colonne frase, lingua, e lunghezza (in questo caso corte).

Il codice crea un file SQL con i comandi per aggiungere alla tabella nuovi record, prende i file generati in precedenza e inserisce per ogni file la frase e la lingua rimpiazzando gli apostrofi ed infine salva il file. Così abbiamo il file SQL pronto per l’inserimento nel database. Grazie a questo metodo possiamo fare quante frasi vogliamo e della lunghezza che vogliamo.

# Itera su ogni lingua e il corrispondente file di testo

for lingua, file in files.items():

with open(file, "r", encoding="utf-8") as f:

# Apre il file di testo in modalità lettura

      for line in f:  # Legge ogni riga del file

             if "Frase" in line:

# Controlla se la riga contiene la parola "Frase"

              # Estrae la frase dalla riga

                  frase = line.split(": ", 1)[1].split(" con ")[0].strip()

# Gestisce gli apostrofi interni, sostituendo un apostrofo con due

                  frase = frase.replace("'", "''")

# Aggiunge la frase formattata alla lista dei valori da inserire

            values.append(f"'{frase}', '{lingua}', 'corto'")

# Scrive i valori nel file SQL, separati da virgole e terminati con un punto e virgola

    out.write(",\n".join(values) + ";\n")

# Stampa un messaggio di conferma

print(f"File SQL generato: {output\_file}")

## Implementazione Database

Dopo aver creato il Databse in locale su MYSQL, ci siamo fatti dare un VM dai sistemisti per caricarlo.

Dopo che ci hanno dato l’IP della VM 10.100.1.17 ci siamo collegati tramite SSH con l’utente datoci,

ssh user@10.100.1.17

Una volta dentro ho fatto un update a apt e subito dopo ho installa il servizio di MYSQL.

sudo apt update

sudo apt install mysql-server -y

Dopo abbiamo fatto una compia del file SQL da implementare per creare il DB tramite Powershell con il comando scp che copia dalla mia cartella a una posizione che voglio io nella VM.

scp D:/dattiloking.sql [user@10.100.1.17:/homr/user](mailto:user@10.100.1.17:/homr/user)

scp D:/dattiloking.sql user@10.100.1.17:/homr/user

Dopo rientrato nella VM sono entrato in MYSQL, e ho implementato il DB con SOURCE comando di MYSQL che ti fa importare file SQL.

sudo mysql

SOURCE /home/user/dattiloking.sql ;

Infine abbiamo implementato il file SQL creato con il programma python per caricare le frasi. Per farlo abbiamo usato lo stesso procedimento di prima.

scp D:/insert\_frasi.sql [user@10.100.1.17:/homr/user](mailto:user@10.100.1.17:/homr/user)

SOURCE /home/user/insert\_frasi.sql;

## Implementazione SignUp

**Models\UserMapper.php**

La funzione **signUpManageModel()** si occupa di gestire la registrazione di un nuovo utente nel sistema. Essa esegue diverse operazioni di controllo e salvataggio, come la verifica dell'unicità del nome utente, l'hashing della password e l'inserimento dei dati nel database. Inoltre, al termine del processo, la funzione imposta le variabili di sessione per tenere traccia dell'utente loggato.

public function signUpManageModel($username,$password, $passwordConfirm){

    $utenti = $this->connection->prepare('SELECT \* from utente');

    $utenti->execute();

    foreach ($utenti as $utente){

        if($utente['username'] == $username){

            return false;

        }

    }

    $hashedPassword = password\_hash($password, PASSWORD\_BCRYPT);

    $salvataggioUtente = $this->connection->prepare('INSERT INTO utente (username, password) VALUES (?, ?)');

    $salvataggioUtente->bindParam(1, $username);

    $salvataggioUtente->bindParam(2, $hashedPassword);

    $salvataggioUtente->execute();

    $\_SESSION['username'] = $utente['username'];

    $\_SESSION['logged'] = true;

    return true;

}

**Controller\signup.php**

La funzione **signUpManage()** si occupa della gestione del processo di registrazione di un nuovo utente.

Viene invocata quando l’utente invia il modulo di registrazione con i dati necessari. La funzione esegue una serie di controlli per garantire che i dati forniti siano validi, sicuri e completi prima di procedere con la registrazione.

public function signUpManage(){

    if(isset($\_POST['username']) && !empty($\_POST['username']) &&

        isset($\_POST['password']) && !empty($\_POST['password']) &&

        isset($\_POST['passwordConfirm']) && !empty($\_POST['passwordConfirm'])) {

        $username = $\_POST['username'];

        $password = $\_POST['password'];

        $passwordConfirm = $\_POST['passwordConfirm'];

        $regex = "/^(?=.\*[a-z])(?=.\*[A-Z])(?=.\*\d)(?=.\*[@$!%\*?&\_])[A-Za-z\d@$!%\*?&\_]{8,}$/";

        if(empty($password)) {

            header("Location: " . URL . "signup/form?error=password\_empty");

            exit;

        }

        if(!preg\_match($regex, $password)) {

            header("Location: " . URL . "signup/form?error=password\_invalid");

            exit;

        }

        if($password !== $passwordConfirm) {

            header("Location: " . URL . "signup/form?error=password\_mismatch");

            exit;

        }

        $userMapper = new UserMapper();

        $result = $userMapper->signUpManageModel($username, $password, $passwordConfirm);

        if ($result === false) {

            header("Location: " . URL . "signup/form?error=user\_exists");

            exit;

        } else {

            header("Location: " . URL . "signup/form?success=registered");

            exit;

        }

    } else {

        header("Location: " . URL . "signup/form?error=missing\_fields");

        exit;

    }

}

## Implementazione validazione SignUp

**Views\Account\index.php**

La funzione **validateSignUp()** viene utilizzata per eseguire la validazione dei dati immessi dall'utente nel modulo di registrazione. La funzione verifica che i dati inseriti (nome utente e password) rispettino determinati criteri prima di consentire l'invio del modulo. In caso di errore, viene visualizzato un messaggio di avviso per informare l'utente.

function validateSignUp() {

    var username = document.getElementById("username").value;

    var password = document.getElementById("password").value;

    var passwordConfirm = document.getElementById("passwordConfirm").value;

    var regex = /^(?=.\*[a-z])(?=.\*[A-Z])(?=.\*\d)(?=.\*[@$!%\*?&\_])[A-Za-z\d@$!%\*?&\_]{8,}$/;

    if (!password) {

        alert("La password non può essere vuota!");

        return false;

    }

    if(!username){

        alert("Lo username non può essere vuoto!");

        return false;

    }

    if (!regex.test(password)) {

        alert("La password deve rispettare:\n- Almeno una lettera maiuscola\n- Almeno una lettera minuscola\n- Almeno un numero\n- Almeno un carattere speciale\n- Almeno 8 caratteri di lunghezza");

        return false;

    }

    if (password !== passwordConfirm) {

        alert("Le password non coincidono!");

        return false;

    }

    return true;

}

## Implementazione SignIn

**Controller\signin.php**

La funzione **signInManage()** si occupa di gestire il processo di login di un utente. Viene invocata quando l'utente invia il modulo di login con le proprie credenziali (nome utente e password). La funzione esegue una serie di controlli per verificare la validità dei dati e consentire l'accesso al sistema.

public function signInManage(){

    if(isset($\_POST['username']) && !empty($\_POST['username']) && isset($\_POST['password']) && !empty($\_POST['password'])){

        $username = $\_POST['username'];

        $password = $\_POST['password'];

        $userMapper = new UserMapper();

        $result = $userMapper->loginInManageModel($username, $password);

        if($result === true){

            header("Location: " . URL . "home/logged");

            exit;

        } else {

            header("Location: " . URL . "signin/form?error=invalid\_credentials");

            exit;

        }

    } else {

        header("Location: " . URL . "signin/form?error=missing\_fields");

        exit;

    }

}

**Models\UserMapper.php**

La funzione **loginInManageModel()** gestisce il processo di autenticazione dell'utente nel sistema. Essa verifica le credenziali (nome utente e password) inserite dall'utente confrontandole con quelle presenti nel database. Se le credenziali sono corrette, l'utente viene autenticato e la sessione viene aggiornata per tenere traccia dello stato di login. In caso contrario, viene restituito un errore di login.

public function loginInManageModel($username, $password){

    $utenti = $this->connection->prepare('SELECT \* from utente');

    $utenti->execute();

    foreach ($utenti as $utente) {

        if($utente['username'] == $username){

            if(password\_verify($password, $utente['password'])){

                $\_SESSION['username'] = $utente['username'];

                $\_SESSION['logged'] = true;

                return true;

            }

        }

    }

    $\_SESSION['logged'] = false;

    return false;

}

## Validazione SignIn

**Views\Account\index.php**

La funzione **validateSignIn()** viene utilizzata per eseguire la validazione dei dati immessi dall'utente nel modulo di accesso (login). La funzione verifica che i campi di nome utente e password siano stati compilati correttamente prima di inviare il modulo di login. Se uno dei campi è vuoto, viene visualizzato un messaggio di errore.

function validateSignIn() {

        var username = document.getElementById("signin-username").value.trim();

        var password = document.getElementById("signin-password").value.trim();

        if (!username && !password) {

            document.getElementById("errors").innerHTML = "Inserisci username e password!";

            return false;

        }

        if(!username){

            document.getElementById("errors").innerHTML = "Lo username non può essere vuoto!";

            return false;

        }

        if (!password) {

            document.getElementById("errors").innerHTML = "La password non può essere vuota!";

            return false;

        }

        var formData = new FormData(document.getElementById("signin-form"));

        fetch('<?php echo URL; ?>signin/signInManage', {

            method: 'POST',

            body: formData

        })

            .then(response => response.json())

            .then(data => {

                if (data.error === 'user\_not\_exists') {

                    document.getElementById("errors").innerHTML = "Utente non esiste o Credenziali non valide!";

                } else if (data.success) {

                    window.location.href = "<?php echo URL; ?>home/logged";

                }

            })

        return false;

    }

## Implementazione recupero frase

**Models\UserMapper.php**

La funzione **getPhraseModel()** si occupa di recuperare una frase casuale dalla tabella frase del database, filtrando per la lingua selezionata dall'utente (memorizzata nella variabile di sessione $\_SESSION['lingua']). Se non ci sono frasi disponibili nella lingua specificata, o se si verificano altri errori durante l'esecuzione, la funzione restituisce messaggi di errore appropriati.

public function getPhraseModel() {

    $result = $this->connection->prepare("SELECT COUNT(\*) as total FROM frase WHERE lingua = ?");

    $result->bindParam(1,$\_SESSION['lingua']);

    $result->execute();

    $row = $result->fetch(PDO::FETCH\_ASSOC);

    $totalRow = $row['total'];

    if ($totalRow <= 0) {

        return "Nessun dato presente nel DB";

    }

    $casual = rand(0, $totalRow - 1);

    $query = "SELECT testo FROM frase WHERE lingua = ? LIMIT 1 OFFSET ?";

    $frase = $this->connection->prepare($query);

    $frase->bindParam(1, $\_SESSION['lingua']);

    $frase->bindParam(2, $casual, PDO::PARAM\_INT);

    $frase->execute();

    $result = $frase->fetch(PDO::FETCH\_ASSOC);

    if($result){

        return $result['testo'];}else{

        return "Nessuna frase trovata";}}

## Gestione degli eventi

**Views\singlePlayerFrontend\index.php**

Il codice gestisce gli eventi di pressione dei tasti sulla tastiera (keydown) e fornisce un feedback visivo in tempo reale per gli utenti mentre scrivono. Quando l'utente preme un tasto, la funzione aggiorna lo stato visivo delle tastiere virtuali e invoca la funzione di scrittura per aggiornare la frase in base alla lettera digitata.

document.addEventListener('keydown', function(event) {

    if(primoTastoPremuto == false){

        intervalloTempo = setInterval(incrementaTempo, 1000);

        primoTastoPremuto = true;

    }

    if (event.key === "Shift") {

        let key = event.key.toUpperCase();

        const button = document.getElementById('key-' + key);

        if (button) {

            button.classList.add('pressed');

        }

        return;

    }

    let key = event.key.toUpperCase();

    if (key === " ") key = "space";

    if (key === ",") key = "comma";

    if (key === ".") key = "point";

    const button = document.getElementById('key-' + key);

    if (button) {

        button.classList.add('pressed');

    }

    if (!tastiPremuti[key]) {

        tastiPremuti[key] = true;

        stampaTesto(event.key);

    }});

Il codice gestisce gli eventi di rilascio dei tasti sulla tastiera (keyup) e fornisce un feedback visivo in tempo reale, rimuovendo l'indicazione visiva dei tasti premuti quando l'utente li rilascia. Questo aiuta a mantenere un'interazione dinamica con l'utente durante il processo di scrittura.

document.addEventListener('keyup', function(event) {

    let key = event.key.toUpperCase();

    if (key === " ") key = "space";

    if (key === ",") key = "comma";

    if (key === ".") key = "point";

    const button = document.getElementById('key-' + key);

    if (button) {

        button.classList.remove('pressed');

    }

    tastiPremuti[key] = false;

});

## Implementazione Stampa del testo

**Views\singlePlayerFrontend\index.php**

La funzione **stampaTesto()** è progettata per gestire la visualizzazione dinamica di un testo durante una sessione di scrittura, monitorando le lettere digitate dall'utente e fornendo feedback in tempo reale. Gestisce anche la valutazione delle prestazioni, calcolando la velocità di scrittura, la percentuale di correttezza e il tempo impiegato per completare la frase.

function stampaTesto(lettera){

    fetch('../../php\_mvc/application/controller/phrase.php')

        .then(response => response.text()).then(data => {if(primoAccesso){

                fraseArray = data.split(""); // Converte la frase in un array di caratteri

                document.getElementById("frase").innerText = data;

                primoAccesso = false;}else{let prossimaLettera = fraseArray[indiceLettera].replace(/\u0332/g, '');

                if (prossimaLettera === lettera) {

                    if(indiceLettera <= fraseArray.length-2){

                        let letteraDopoProssimaLettera = fraseArray[indiceLettera+1];

                        fraseArray[indiceLettera+1] = letteraDopoProssimaLettera  + '\u0332';}if(letteraPrecedentementeSbagliata === true){

                        fraseArray[indiceLettera] = `<span class="wrong">${prossimaLettera}</span>`;

                        letteraPrecedentementeSbagliata = false;

                        numeroErrori++;}else{

                        fraseArray[indiceLettera] = `<span class="highlight">${prossimaLettera}</span>`;

                        if(indiceLettera === fraseArray.length-1){

                            let percentualeCorrettezza = 100-(((numeroErrori/fraseArray.length)\*100).toFixed(1));

                 document.getElementById('percentualeCorrettezza').innerText = percentualeCorrettezza+"%";

                            clearInterval(intervalloTempo);

                            //let velocita = ((fraseArray.length/tempo)\*60).toFixed(1);

                            let velocita = (fraseArray.length/5)/(tempo/60);

                            document.getElementById('velocita').innerHTML = velocita + " WPM";

                            document.getElementById('tempo').innerHTML = tempo + " sec";

                            tempo = 0;primoTastoPremuto = false;indiceLettera = 0;primoTastoPremuto = false;primoAccesso = true;fraseArray = [];numeroErrori = 0;stampaTesto();}}indiceLettera++;}

                else{letteraPrecedentementeSbagliata = true;}

                document.getElementById("frase").innerHTML =       fraseArray.join("");}})

        .catch(error => console.error('Errore:', error));}

## Implementazione salvataggio statistiche

**Models\UserMapper.php**

La funzione **saveStatsModel()** si occupa di salvare le statistiche di un turno di gioco dell'utente nel database. Le statistiche includono l'accuratezza, la velocità, il tempo, la frase utilizzata durante il gioco e l'identificativo del turno. La funzione gestisce anche il calcolo del numero del turno successivo per l'utente, in modo che i dati vengano correttamente inseriti nella tabella turno\_storico.

public function saveStatsModel($accuratezza,$velocita,$tempo,$username,$frase){

    $pdoQueryTurno = $this->connection->prepare('SELECT MAX(numero) FROM turno\_storico WHERE utente\_username = ?');

    $pdoQueryTurno->bindParam(1,$username);

    $pdoQueryTurno->execute();

    $ultimoTurnoUtente = $pdoQueryTurno->fetchColumn();

    if ($ultimoTurnoUtente !== false) {

        $turno = $ultimoTurnoUtente + 1;

    } else {

        $turno = 0;

    }

    $pdoQuerySalvataggio = $this->connection->prepare('INSERT INTO turno\_storico(accuratezza, velocita, tempo, data, utente\_username, frase\_testo, numero) VALUES (?, ?, ?, ?, ?,?,?)');

    $pdoQuerySalvataggio->bindParam(1, $accuratezza);

    $pdoQuerySalvataggio->bindParam(2, $velocita);

    $pdoQuerySalvataggio->bindParam(3, $tempo);

    $dataAttuale = date('Y-m-d');

    $pdoQuerySalvataggio->bindParam(4, $dataAttuale);

    $pdoQuerySalvataggio->bindParam(5, $username);

    $pdoQuerySalvataggio->bindParam(6, $frase);

    $pdoQuerySalvataggio->bindParam(7, $turno);

    $executed = $pdoQuerySalvataggio->execute();

    if($executed && $pdoQuerySalvataggio->rowCount() > 0){

        return true;

    }else {

        return false;

    }

}

**Controller\save.php**

La funzione **saveStats()** gestisce il salvataggio delle statistiche di un utente autenticato. Legge i dati JSON ricevuti via php://input (accuratezza, velocità, tempo e frase) e li salva nel database tramite il metodo saveStatsModel() della classe UserMapper. Prima di procedere verifica che l’utente sia autenticato ($\_SESSION['logged'] == true) e che i dati richiesti siano presenti. Restituisce una risposta JSON che indica l'esito dell'operazione (successo, errore, dati mancanti o utente non autenticato).

public function saveStats(){

    if(isset($\_SESSION['logged']) && $\_SESSION['logged']){

        $data = json\_decode(file\_get\_contents('php://input'), true);

        if (isset($data['accuratezza']) && isset($data['velocita']) && $data['tempo']) {

            $accuratezza = $data['accuratezza'];

            $velocita = $data['velocita'];

            $tempo = $data['tempo'];

            if(isset($\_SESSION['username'])){

                $username = $\_SESSION['username'];

            }

            $frase = $data['frase'];

            $userMapper = new UserMapper();

            $result = $userMapper->saveStatsModel($accuratezza, $velocita, $tempo, $username, $frase);

            if($result){

                echo json\_encode(['status' => 'success', 'message' => 'Dati salvati']);

            }else{

                echo json\_encode(['status' => 'error', 'message' => 'Errore nel salvataggio dei dati']);

            }

        }else{

            echo json\_encode(['status' => 'error', 'message' => 'Dati mancanti']);

        }

    }else{

        echo json\_encode(['status' => 'error', 'message' => 'Utente non autenticato']);

    }

}

## Implementazione visualizzazione statistiche

**Controller\profile.php**

La funzione **showStats()** restituisce le statistiche dell’utente autenticato in formato JSON, solo se la richiesta è una chiamata AJAX (GET con header X-Requested-With: XMLHttpRequest). Verifica che l’utente sia loggato e, tramite la classe UserMapper, richiama il metodo showStatsModel() passando il nome utente dalla sessione. I dati vengono poi inviati al client con intestazione Content-Type: application/json.

public function showStats(){

    if ($\_SERVER['REQUEST\_METHOD'] === 'GET' && isset($\_SERVER['HTTP\_X\_REQUESTED\_WITH']) &&

        strtolower($\_SERVER['HTTP\_X\_REQUESTED\_WITH']) === 'xmlhttprequest'){

        if($\_SESSION['logged']){

            require 'application/models/UserMapper.php';

            if(isset($\_SESSION['username'])){

                $username = $\_SESSION['username'];

            }

            $userMapper = new UserMapper();

            $stats = $userMapper->showStatsModel($username);

            header('Content-Type: application/json');

            echo json\_encode($stats);

            exit();

        }

    }

}

**Models\UserMapper.php**

La funzione **showStatsModel()** recupera e restituisce le statistiche aggregate di un utente dal database. Vengono calcolati i valori minimi, massimi e medi della velocità, accuratezza e durata (tempo) dei turni registrati nella tabella turno\_storico, filtrati per nome utente (utente\_username). I dati sono organizzati in due gruppi:

datiEterni: statistiche calcolate su tutta la cronologia dell’utente.

datiGiornalieri: (attualmente identici ai dati eterni, suggerendo una futura estensione per filtro per giorno).

Restituisce un array associativo contenente entrambe le sezioni.

public function showStatsModel($username){

    $velMin = $this->connection->prepare('SELECT MIN(velocita) FROM turno\_storico WHERE utente\_username = ?;');

    $velMin->bindParam(1, $username);$velMin->execute();

    $velMinValue = $velMin->fetchColumn();

    $velMinDay = $this->connection->prepare('SELECT MIN(velocita) FROM turno\_storico WHERE utente\_username = ?;');

    $velMinDay->bindParam(1, $username);$velMinDay->execute();

    $velMinValueDay = $velMinDay->fetchColumn();

    $velMax = $this->connection->prepare('SELECT MAX(velocita) FROM turno\_storico WHERE utente\_username = ?;');

    $velMax->bindParam(1, $username);$velMax->execute();

    $velMaxValue = $velMax->fetchColumn();

    $velMaxDay = $this->connection->prepare('SELECT MAX(velocita) FROM turno\_storico WHERE utente\_username = ?;');

    $velMaxDay->bindParam(1, $username);$velMaxDay->execute();

    $velMaxValueDay = $velMaxDay->fetchColumn();

    $velAvg = $this->connection->prepare('SELECT AVG(velocita) FROM turno\_storico WHERE utente\_username = ?;');

    $velAvg->bindParam(1, $username);$velAvg->execute();

    $velAvgValue = $velAvg->fetchColumn();

    $velAvgDay = $this->connection->prepare('SELECT AVG(velocita) FROM turno\_storico WHERE utente\_username = ?;');

    $velAvgDay->bindParam(1, $username);$velAvgDay->execute();

    $velAvgValueDay = $velAvgDay->fetchColumn();

    $accuratezzaAvg = $this->connection->prepare('SELECT AVG(accuratezza) FROM turno\_storico WHERE utente\_username = ?;');

    $accuratezzaAvg->bindParam(1, $username);$accuratezzaAvg->execute();

    $accuratezzaAvgValue = $accuratezzaAvg->fetchColumn();

    $accuratezzaAvgDay = $this->connection->prepare('SELECT AVG(accuratezza) FROM turno\_storico WHERE utente\_username = ?;');

    $accuratezzaAvgDay->bindParam(1, $username);$accuratezzaAvgDay->execute();

    $accuratezzaAvgValueDay = $accuratezzaAvgDay->fetchColumn();

    $durataAvg = $this->connection->prepare('SELECT AVG(tempo) FROM turno\_storico WHERE utente\_username = ?;');

    $durataAvg->bindParam(1, $username);$durataAvg->execute();

    $durataAvgValue = $durataAvg->fetchColumn();

    $durataAvgDay = $this->connection->prepare('SELECT AVG(tempo) FROM turno\_storico WHERE utente\_username = ?;');

$durataAvgDay->bindParam(1, $username);$durataAvgDay->execute();

    $durataAvgValueDay = $durataAvgDay->fetchColumn();

    $datiEterni = [

        'velMin' => $velMinValue,'velMax' => $velMaxValue,'velAvg' => $velAvgValue,'accuratezzaAvg' => $accuratezzaAvgValue,'durataAvg' => $durataAvgValue];$datiGiornalieri = ['velMin' => $velMinValueDay,'velMax' => $velMaxValueDay,'velAvg' => $velAvgValueDay,'accuratezzaAvg' => $accuratezzaAvgValueDay,'durataAvg' => $durataAvgValueDay];$dati = ['datiEterni' => $datiEterni,'datiGiornalieri' =>$datiGiornalieri];return $dati;}

## Audio

**Views\homeNotLogged\index.php**

La funzione **updateAudio()** Gestisce la riproduzione dell’audio in base al radiobutton selezionato.

Se "On" è attivo, riprende la riproduzione dall’inizio o dall’ultimo punto salvato; se "Off", mette in pausa l’audio.

function updateAudio() {

        // Se "on" è selezionato, avvia la riproduzione dell'audio

        if (document.getElementById("radio-on").checked === true) {

            if (localStorage.getItem("audioTime") !== null){

                const savedTime = localStorage.getItem("audioTime");

                localStorage.removeItem("audioTime");

                audio.currentTime = parseFloat(savedTime);

                audio.play().catch(error => console.error("Errore durante la riproduzione:", error));

            }

            else {

                audio.play()

                    .catch(error => console.error("Errore durante la riproduzione:", error));

            }

        } else {

            audio.pause();

        }

    }

La funzione **saveAudioProgress()** salva nel localStorage lo stato selezionato dei radiobutton ("on" o "off") e il tempo corrente dell’audio (solo se "on"). Serve per ripristinare la situazione alla prossima visita della pagina.

function saveAudioProgress() {

        // Recupera il valore del radiobutton selezionato

        const audioStatus = document.querySelector('input[name="audio"]:checked').value;

        // Verifica se l'audio è su "On" e salva i progressi

        if (audioStatus === "on") {

            // viene usato JSON.stringify per salvare il valore booleano come stringa

            localStorage.setItem("radio-on", JSON.stringify(true));

            localStorage.setItem("radio-off", JSON.stringify(false));

            localStorage.setItem("audioTime", audio.currentTime); // Salva la posizione attuale

            console.log("Progressi audio salvati");

        } else {

            localStorage.setItem("radio-off", JSON.stringify(true));

            localStorage.setItem("radio-on", JSON.stringify(false));

            console.log("L'audio è spento");

        }

    }

Gli eventi descritti fanno si che all'avvio della pagina carica lo stato salvato dei radiobutton, richiama **updateAudio()** per gestire la riproduzione Aggiunge un evento click unico per abilitare l'audio (richiesto da molti browser per motivi di sicurezza).

const radioButtons = document.querySelectorAll('input[name="audio"]'); // Seleziona tutti i radiobutton con name="audio"

    radioButtons.forEach(radio => {

        radio.addEventListener("change", updateAudio);

    });

    window.onload = () => {console.log(localStorage.length);

        if (localStorage.length !== 0) {

            const on = JSON.parse(localStorage.getItem("radio-on"));

            const off = JSON.parse(localStorage.getItem("radio-off"));

            document.getElementById("radio-on").checked = on;

            document.getElementById("radio-off").checked = off;

        }updateAudio();

        window.addEventListener('click', () => {

            if (document.getElementById("radio-on").checked) {

                audio.play().catch(error => console.error("Errore durante la riproduzione:", error));

            }

        }, { once: true });};

## Delete User

**Models\UserMapper.php**

La funzione **deleteUserModel()** elimina l'utente attualmente loggato dal database, rimuovendo prima i dati correlati nella tabella turno\_storico e poi i dati dell'utente stesso dalla tabella utente.

public function deleteUserModel() {

        $deleteTurnoStorico = $this->connection->prepare("DELETE FROM turno\_storico WHERE utente\_username = ?");

        $deleteTurnoStorico->bindParam(1, $\_SESSION['username']);

        $deleteTurnoStorico->execute();

        $delete = $this->connection->prepare("DELETE FROM utente WHERE username = ?");

        $delete->bindParam(1, $\_SESSION['username']);

        $delete->execute();

        if ($delete->rowCount() > 0) {

            unset($\_SESSION['username']);

            unset($\_SESSION['lingua']);

            return true;

        } else {

            return false;}

**Controller\delete.php**

La funzione deleteUser() controlla il flusso dell'eliminazione dell'utente lato controller e restituisce una risposta JSON in base all'esito, poi reindirizza l’utente.

public function deleteUser(){

        $userMapper = new UserMapper();

        $result = $userMapper->deleteUserModel();

        if($result){

            echo json\_encode(['status' => 'success', 'message' => 'Utente Salvato con successo!']);

        }else{

            echo json\_encode(['status' => 'error', 'message' => 'Errore nell eliminazione dell utente!']);

        }

        header('location: ' . URL . 'home/notLogged');

    }

# Test

## Protocollo di test

Definire in modo accurato tutti i test che devono essere realizzati per garantire l’adempimento delle richieste formulate nei requisiti. I test fungono da garanzia di qualità del prodotto. Ogni test deve essere ripetibile alle stesse condizioni.

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-001  REQ-001 | **Nome:** | Visualizzazione testo |
| **Descrizione:** | Visualizzazione della frase nel gioco Singleplayer e Multiplayer | | |
| **Prerequisiti:** | * Si necessita dell’interfaccia di gioco * Si necessita del DB | | |
| **Procedura:** | Una volta entrato dentro la modalità Singleplayer o aver avviato il Multiplayer appare un testo | | |
| **Risultati attesi:** | Quando si entra nella modalità Singleplayer o si avvia la Multiplayer appare il testo | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-002  REQ-002 | **Nome:** | Visualizzazione classifica Multiplayer |
| **Descrizione:** | Visualizzazione della classifica e punteggi dei vari utenti | | |
| **Prerequisiti:** | * Si necessita dell’interfaccia di gioco * Si necessita del DB | | |
| **Procedura:** | Dopo aver finito i round per la modalità Multiplayer, appare la classfica | | |
| **Risultati attesi:** | Quando sono finiti I round appare la classfica di tutti i giocatori | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-003  REQ-003 | **Nome:** | Percentuale correttezza |
| **Descrizione:** | Calcolo della percentuale delle lettere scritte correttamente | | |
| **Prerequisiti:** | * Si necessita dell’interfaccia di gioco * Si necessita del DB | | |
| **Procedura:** | Per il Singleplayer, alla fine della frase vengono illustrati la percentuali di errori che si ha comesso durante la frase. Per il Multiplayer invece, viene mostrata la media della percentuali degli errori commessi durante I vari round. | | |
| **Risultati attesi:** | Quando si finisce una frase nel Singleplayer viene mostrata la percentuale degli errori commessi e nel Multiplayer, una volta finiti i round viene mostrata la media degli errori per ogni utente. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-004  REQ-004 | **Nome:** | Velocità scrittura |
| **Descrizione:** | Calcolo della velocità media in cui si scrive una frase | | |
| **Prerequisiti:** | * Si necessita dell’interfaccia di gioco * Si necessita del DB | | |
| **Procedura:** | Nel Singleplayer, alla fine della frase viene illustrato il WPM (parole al minuto) in base al tempo che si ha impiegato per finire la frase e ai caratteri presenti nella frase. Nel Multiplayer viene mostrata la media del WPM durante i vari round. | | |
| **Risultati attesi:** | Quando si finisce una frase nel Singleplayer viene mostrato il WPM e nel Multiplayer, una volta finiti i round viene mostrata la media del WPM per ogni utente. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-005  REQ-005 | **Nome:** | Tempo scrittura |
| **Descrizione:** | Calcolo del tempo in cui viene finita una frase | | |
| **Prerequisiti:** | * Si necessita dell’interfaccia di gioco * Si necessita del DB | | |
| **Procedura:** | Nel Singleplayer, alla fine della frase viene illustrato a schermo il tempo che si ha impiegato per finirla. Per il Multiplayer invece, viene mostrata la media del tempo durante i vari round | | |
| **Risultati attesi:** | Quando si finisce una frase nel Singleplayer viene mostrato il tempo che si ha impiegato e nel Multiplayer, una volta finiti i round viene mostrata la media del tempo per ogni utente. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-006  REQ-006 | **Nome:** | Visualizzazione e animazione grafica della scrittura |
| **Descrizione:** | Mostra la tastiera e l’animazione degli input dell’utente | | |
| **Prerequisiti:** | * Si necessita dell’interfaccia di gioco | | |
| **Procedura:** | Sia per il Singleplayer che per il Multiplayer, una volta avviati, apparirà una tastiera che una volta premuto un qualsiasi tasto presente su di essa, ci sarà un’animazione sull’immagine del tasto premuto. | | |
| **Risultati attesi:** | Dopo essere entrati nelle due modalità ci sarà la tastiera con le animazioni per I tasti. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-007  REQ-007 | **Nome:** | Selezione della lingua |
| **Descrizione:** | L’utente ha la possibilità di scegliere la lingua | | |
| **Prerequisiti:** | * Si necessita dell’interfaccia di gioco * Si necessita del DB | | |
| **Procedura:** | All’interno della pagina home, sia per utenti loggati che non, premendo sul logo delle impostazioni, si avrà la possibilità di scegliere la lingua con cui ci si voglia esercitare. Una volta avviata la modalità Singleplayer si potrà giocare con la lingua selezionata in precedenza | | |
| **Risultati attesi:** | Dopo aver selezionato la lingua, entrando nella modalità Singleplayer, il testo da scrivere sarà nelle lingua che si ha selezionato. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-008  REQ-008 | **Nome:** | Seleziona tema di sfondo |
| **Descrizione:** | L’utente ha la possibilità di selezionare un tema di sfondo | | |
| **Prerequisiti:** | * Si necessita dell’interfaccia di gioco | | |
| **Procedura:** | All’interno della pagina home, sia per utenti loggati che non, premendo sul logo delle impostazioni, si avrà la possibilità di scegliere un tema con cui ci si voglia giocare. | | |
| **Risultati attesi:** | Dopo aver selezionato il tema, tutto l’applicativo sarà con quel formato. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-009  REQ-009 | **Nome:** | Multiplayer |
| **Descrizione:** | Permette di giocare da più dispositivi simultaneamente | | |
| **Prerequisiti:** | * Si necessita dell’interfaccia di gioco * Si necessita del DB | | |
| **Procedura:** | Una volta loggati, si avrà la possibilità di giocare la modalità Multiplayer, creare una stanza o entrarci. Il creatore avrà un codice per entrare nella stanza che fornirà ai vari giocatori. Per ogni round, gli utenti presenti nella stanza dovranno scrivere una frase che sarà uguale per tutti. | | |
| **Risultati attesi:** | Durante la creazione della stanza, il creatore sceglierà il numero di round e gli utenti si uniranno alla stanza tramite il codice fornito dal creatore. All’avvio, ogni utente dovrà scrivere la frase che è uguale per tutti | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-010  REQ-010 | **Nome:** | Gestione audio |
| **Descrizione:** | L’utente ha la possibilità di abilitare o disattivare l’audio | | |
| **Prerequisiti:** | * Si necessita dell’interfaccia di gioco | | |
| **Procedura:** | All’interno della pagina home, sia per utenti loggati che non, premendo sul logo delle impostazioni, si avrà la possibilità di attivare o disattivare l’audio del gioco con cui si voglia giocare. | | |
| **Risultati attesi:** | Dopo aver attivato l’audio, si potrà ascoltare la musica del gioco, oppure toglierla. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-011  REQ-011 | **Nome:** | Visualizzazione punteggi Singleplayer |
| **Descrizione:** | Visualizza i punteggi giornalieri e di tutti i tempi dell’utente | | |
| **Prerequisiti:** | * Si necessita dell’interfaccia di gioco * Si necessita del DB | | |
| **Procedura:** | Per gli utenti loggati, all’interno della pagina home, premendo sul logo della persona, si avrà la possibilità di osservare tutte le statistiche giornaliere e di tutti i tempi totalizzati da quell’utente. | | |
| **Risultati attesi:** | Dopo essere entrati nelle statistiche dell’utente loggato, si potranno osservare tutte le statistiche giornaliere e di tutti i tempi. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-012  REQ-012 | **Nome:** | Scelta turni Multiplayer |
| **Descrizione:** | Creatore della stanza multiplayer sceglie il numero di round | | |
| **Prerequisiti:** | * Si necessita dell’interfaccia di gioco | | |
| **Procedura:** | Durante la creazione della stanza del Multiplayer, il creatore sceglie il numero di round che avrà quella stanza. Dopo l’avvio, ogni utente presente nella stanza parteciperà al numero di round scelti dal creatore. | | |
| **Risultati attesi:** | Quando il creatore avrà avviato il Multiplayer, tutti gli utente della stanza parteciperanno al numero di round scelti dal creatore | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-013  REQ-013 | **Nome:** | Cancellazione utente |
| **Descrizione:** | L’utente ha la possibilità di cancellare l’utente loggato | | |
| **Prerequisiti:** | * Si necessita dell’interfaccia di gioco * Si necessita del DB | | |
| **Procedura:** | Per gli utenti loggati, all’interno della pagina home, premendo sul logo delle impostazioni, si avrà la possibilità di cancellare l’utente loggato. | | |
| **Risultati attesi:** | Quando quell’utente verrà eliminato, non si potrà più fare l’accesso direttamente. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-014  REQ-014 | **Nome:** | Controllo battitura |
| **Descrizione:** | Esegue controlli nella battitura | | |
| **Prerequisiti:** | * Si necessita dell’interfaccia di gioco * Si necessita del DB | | |
| **Procedura:** | Durante la scrittura di una frase, per ogni carattere scritto in modo sbagliato, l’utente non potrà andare avanti con la frase finché non digiterà correttamente il carattere. | | |
| **Risultati attesi:** | Quando l’utente sbaglia a scrivere un carattere, il puntatore rimane fermo finché non lo scrive correttamente. | | |
|  |  | | |

## 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.

|  |  |
| --- | --- |
| **TC-001** | Passato |
| **TC-002** | Passato |
| **TC-003** | Passato |
| **TC-004** | Passato |
| **TC-005** | Passato |

## Mancanze/limitazioni conosciute

Tra gli elementi mancanti di questo progetto possiamo concentrarci sui seguenti:

Tema di sfondo: non è stato implementato il sistema per la scelta del tema, a causa del tempo limitato a disposizione.

Generazione delle frasi: non siamo riusciti ad ottenere frasi con un numero di caratteri predefinito. Le intelligenze artificiali utilizzate non restituivano risultati precisi in termini di lunghezza, rendendole difficili da integrare correttamente.

Utilizzo dei WebSocket: non abbiamo utilizzato i WebSocket a causa della scarsa conoscenza del loro funzionamento. Questo avrebbe richiesto uno studio in più che non abbiamo avuto modo di fare, e di conseguenza abbiamo fatto ampio uso di fetch. Ciò ha portato a un codice meno ottimizzato e più lento nelle prestazioni.

# Consuntivo

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

# Conclusioni

Questo progetto è stato realizzato per le persone che desiderano divertirsi con un gioco di dattilografia, mettendo alla prova la propria velocità e precisione sulla tastiera. La modalità multiplayer permette di sfidare amici in tempo reale, rendendo l’esperienza ancora più divertente e competitiva. Oltre al divertimento e alla semplicità di questo progetto, può essere un ottimo strumento per migliorare le proprie abilità e velocità di scrittura. Questo progetto rappresenta un traguardo importante per noi, essendo il nostro primo progetto creato con il modello Agile.

Esempio CONCLUSIONI:

Questo progetto è stato pensato e realizzato per tutte le persone che desiderano unire l’utile al dilettevole, divertendosi con un gioco di dattilografia in grado di mettere realmente alla prova la propria velocità e precisione sulla tastiera. L’interfaccia semplice e intuitiva lo rende accessibile a chiunque, ma la vera sfida risiede nel migliorare costantemente le proprie performance.

Un elemento chiave che arricchisce l’esperienza è la modalità multiplayer, che consente di sfidare amici o altri utenti in tempo reale. Questo non solo aumenta il coinvolgimento e il divertimento, ma introduce anche una sana componente competitiva che spinge i partecipanti a dare il massimo, rendendo ogni partita unica e stimolante.

Oltre all’aspetto ludico, il progetto ha anche una forte valenza educativa: può essere un ottimo strumento per migliorare le proprie abilità di scrittura, la velocità di battitura e l’accuratezza nella digitazione, qualità sempre più richieste in ambito lavorativo e scolastico.

Per noi, questo progetto rappresenta molto più di un semplice gioco: è un vero e proprio traguardo. È infatti il primo progetto che abbiamo sviluppato seguendo il modello Agile, affrontando tutte le fasi di progettazione, sviluppo e test in modo strutturato e collaborativo. Questo ci ha permesso di acquisire nuove competenze tecniche e organizzative, consolidando la nostra capacità di lavorare in team in un contesto reale di sviluppo software.

## Sviluppi futuri

Migliorie o estensioni che possono essere sviluppate sul prodotto.

## Considerazioni personali

### Amos Haefliger

### Nemanja Zecevic

### Robin Sartore

### Leonardo Sciara

Lavorare a questo progetto è stata un’esperienza estremamente formativa. Creare un gioco di dattilografia con l’implementazione del multiplayer ci ha permesso non solo di applicare le nostre conoscenze tecniche, ma anche di collaborare e imparare a gestire un progetto dall’inizio alla fine. Creare un qualcosa che è diventato un qualcosa di concreto, funzionante e divertente da usare, ci ha dato una grande soddisfazione. Credo che questo progetto rappresenti un primo importate passo nel nostro percorso di crescita, sia come sviluppatori che come team.

# Indice delle Figure

[Figura 1 Schema Use Case 9](#_Toc197517031)

[Figura 2 Schema del Gantt 10](#_Toc197517032)

[Figura 3 Schema delle strutture di sistema 12](#_Toc197517033)

[Figura 4 Schema E-R 13](file:///C:\Users\amos.haefliger\DattiloKing\3_Documentazione\Documentazione%20DattiloKing.docx#_Toc197517034)

[Figura 5 HomePage senza essere loggati 14](file:///C:\Users\amos.haefliger\DattiloKing\3_Documentazione\Documentazione%20DattiloKing.docx#_Toc197517035)

[Figura 6 Settings 14](file:///C:\Users\amos.haefliger\DattiloKing\3_Documentazione\Documentazione%20DattiloKing.docx#_Toc197517036)

[Figura 7 Form di Login 15](file:///C:\Users\amos.haefliger\DattiloKing\3_Documentazione\Documentazione%20DattiloKing.docx#_Toc197517037)

[Figura 8 Form di SignUp 15](file:///C:\Users\amos.haefliger\DattiloKing\3_Documentazione\Documentazione%20DattiloKing.docx#_Toc197517038)

[Figura 9 HomePage da loggati 16](#_Toc197517039)

[Figura 10 Settings da loggato 16](#_Toc197517040)

[Figura 11 Keyboard di gioco singleplayer 17](file:///C:\Users\amos.haefliger\DattiloKing\3_Documentazione\Documentazione%20DattiloKing.docx#_Toc197517041)

[Figura 12 Statistiche personali 18](#_Toc197517042)

[Figura 13 Multiplayer 18](#_Toc197517043)

[Figura 14 Creazione stanza 19](file:///C:\Users\amos.haefliger\DattiloKing\3_Documentazione\Documentazione%20DattiloKing.docx#_Toc197517044)

[Figura 15 Inserimento codice, unione stanza 19](file:///C:\Users\amos.haefliger\DattiloKing\3_Documentazione\Documentazione%20DattiloKing.docx#_Toc197517045)

[Figura 16 Keyboard di gioco multiplayer 20](file:///C:\Users\amos.haefliger\DattiloKing\3_Documentazione\Documentazione%20DattiloKing.docx#_Toc197517046)

[Figura 17 Leaderboard multiplayer 20](file:///C:\Users\amos.haefliger\DattiloKing\3_Documentazione\Documentazione%20DattiloKing.docx#_Toc197517047)

[Figura 18 Diagramma delle classi 21](file:///C:\Users\amos.haefliger\DattiloKing\3_Documentazione\Documentazione%20DattiloKing.docx#_Toc197517048)

[Figura 19 Diagramma di flusso 22](file:///C:\Users\amos.haefliger\DattiloKing\3_Documentazione\Documentazione%20DattiloKing.docx#_Toc197517049)

[Figura 20 Diagramma di flusso 23](file:///C:\Users\amos.haefliger\DattiloKing\3_Documentazione\Documentazione%20DattiloKing.docx#_Toc197517050)

[Figura 21 Struttura cartelle 24](file:///C:\Users\amos.haefliger\DattiloKing\3_Documentazione\Documentazione%20DattiloKing.docx#_Toc197517051)

# 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 | **Asynchronous JavaScript And XML**: una tecnica che permette di eseguire richieste ed ottenere dati da una pagina web in modo asincrono. |
| CSS | **Cascading Style Sheets**: linguaggio che permette di definire il layout e la grafica di una pagina web. |

# Bibliografia

## Bibliografia per articoli di riviste:

1. Cognome e nome (o iniziali) dell’autore o degli autori, o nome dell’organizzazione,
2. Titolo dell’articolo (tra virgolette),
3. Titolo della rivista (in italico),
4. Anno e numero
5. Pagina iniziale dell’articolo.

## Bibliografia per libri

1. Cognome e nome (o iniziali) dell’autore o degli autori, o nome dell’organizzazione,
2. Titolo del libro (in italico),
3. ev. Numero di edizione,
4. Nome dell’editore,
5. Anno di pubblicazione,
6. ISBN.

## Sitografia

1. URL del sito (se troppo lungo solo dominio, evt completo nel diario),
2. Eventuale titolo della pagina (in italico),
3. Data di consultazione (GG-MM-AAAA).

**Esempio:**

* http://standards.ieee.org/guides/style/section7.html, *IEEE Standards Style Manual*, 07-06-2008.

# Allegati

Elenco degli allegati, esempio:

* Diari di lavoro
* Codici sorgente/documentazione macchine virtuali
* Istruzioni di installazione del prodotto (con credenziali di accesso) e/o di eventuali prodotti terzi
* Documentazione di prodotti di terzi
* Eventuali guide utente / Manuali di utilizzo
* Mandato e/o QdC
* Prodotto
* …