

DattiloKing

Titolo del progetto: DattiloKing
Alunno/a: Amos Haefliger, Leonardo Sciara, Nemanja Zecevic, Robin Sartore
Classe: Info I3BB/I3AA
Anno scolastico: 2024/2025
Docente responsabile: Michel Palucci

1	Introduzione	4
1.1	Informazioni sul progetto	4
1.2	Scopo	4
2	Analisi	4
2.1	Analisi del dominio	4
2.2	Analisi e specifica dei requisiti	5
2.3	Use case	10
2.4	Pianificazione	11
2.4.1	Sprint 1	12
2.4.2	Sprint 2	12
2.4.3	Sprint 3	13
2.4.4	Sprint 4	13
2.4.5	Sprint 5	13
2.4.6	Sprint 6	14
2.4.7	Consegna	14
2.5	Analisi dei mezzi	15
2.5.1	Software	15
2.5.2	Hardware	15
3	Progettazione	16
3.1	Design dell'architettura del sistema	16
3.2	Design dei dati e database	17
3.3	Design delle interfacce	18
3.4	Design procedurale	25
3.4.1	Diagramma delle classi	25
3.4.2	Diagramma di flusso	26
4	Implementazione	27
4.1	Struttura e ambiente del progetto	27
4.2	Generazione frasi	28
4.2.1	Python per generare le frasi	28
4.2.2	Formattazione SQL	29
4.3	Implementazione Database	30
4.4	Implementazione SignUp	31
4.5	Implementazione validazione SignUp	33
4.6	Implementazione SignIn	34
4.7	Validazione SignIn	35
4.8	Implementazione recupero frase	36
4.9	Gestione degli eventi	37
4.10	Implementazione Stampa del testo	38
4.11	Implementazione salvataggio statistiche	39
4.12	Implementazione visualizzazione statistiche	41
4.13	Audio	43
4.14	Delete User	45
4.15	Implementazione generazione codice stanza	46
4.16	Mostrare giocatori nella room	47
5	Test	48
5.1	Protocollo di test	48
5.2	Risultati test	52
5.3	Mancanze/limitazioni conosciute	52
6	Consuntivo	53
7	Conclusioni	58
7.1	Sviluppi futuri	58
7.2	Considerazioni personali	59
7.2.1	Amos Haefliger	59
7.2.2	Nemanja Zecevic	59
7.2.3	Robin Sartore	59
7.2.4	Leonardo Sciara	60
8	Indice delle Figure	61
9	Glossario	62

10	Sitografia	63
11	Allegati	63

1 Introduzione

1.1 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

1.2 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.

2 Analisi

2.1 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.

2.2 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 necessita 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 necessita 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 necessita 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 necessita 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 necessita 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 necessita 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 necessita 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 necessita 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 necessita del DB

2.3 Use case

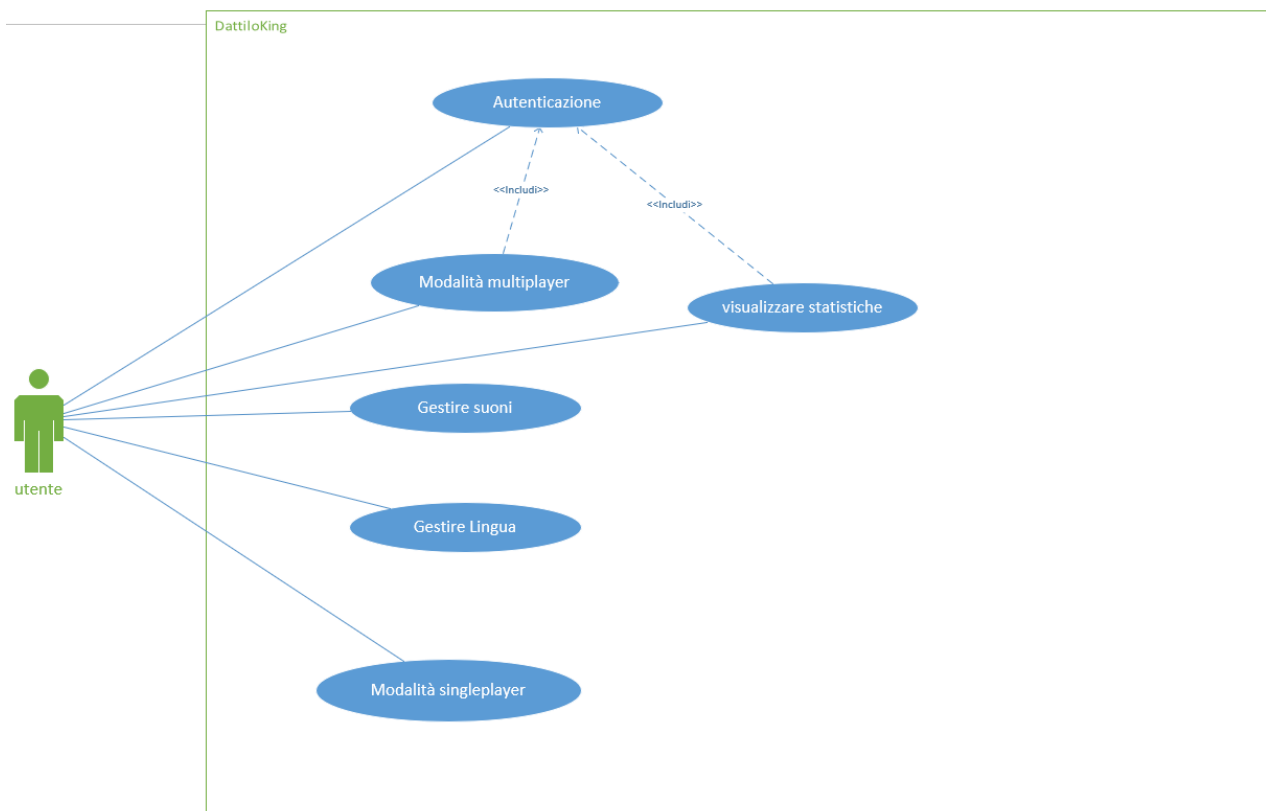


Figura 1 Schema Use Case

Questo è lo Use Case del progetto DattiloKing. L'attore principale è l'utente, che può interagire con diverse funzionalità del sistema. Alcune funzioni come modalità multiplayer e visualizzazione delle statistiche sono collegate alla funzionalità di autenticazione: ciò significa che l'utente deve essere autenticato per accedervi. Altre funzionalità accessibili direttamente includono: l'autenticazione, che l'utente può eseguire in qualsiasi momento, modalità singleplayer, gestione dei suoni e selezione della lingua. Il diagramma offre una panoramica chiara delle azioni disponibili per l'utente e delle dipendenze.

2.4 Pianificazione

In allegato si trova il Gantt completo: **Gantt\Preventivo\Gantt.mpp**

★	▾ DattiloKing	90 h	mer 29.01.25	mer 28.05.25
☞	▾ Teoria Agile	3 h	mer 29.01.25	mer 29.01.25
☞	Concetti Base	3 h	mer 29.01.25	mer 29.01.25
☞	▾ Progettazione	3 h	mer 29.01.25	mer 29.01.25
☞	Creazione git	0.5 h	mer 29.01.25	mer 29.01.25
☞	Creazione Trello	0.5 h	mer 29.01.25	mer 29.01.25
☞	Setup Trello	0.5 h	mer 29.01.25	mer 29.01.25
☞	Requisiti	1 h	mer 29.01.25	mer 29.01.25
☞	Gantt	0.5 h	mer 29.01.25	mer 29.01.25
★	▾ Implementazione	84 h	mer 05.02.25	mer 28.05.25
★	▷ Sprint 1	12 h	mer 05.02.25	mer 19.02.25
★	▷ Sprint 2	12 h	mer 19.02.25	mer 12.03.25
★	▷ Sprint 3	16 h	mer 12.03.25	mer 09.04.25
☞	<u>Milestone 1</u>	1 h	mer 09.04.25	mer 09.04.25
☞	▷ Sprint 4	12 h	mer 05.02.25	mer 16.04.25
☞	▷ Sprint 5	12 h	mer 05.02.25	mer 14.05.25
☞	Milestone 2	1 h	mer 14.05.25	mer 14.05.25
☞	▷ Sprint 6	6 h	mer 14.05.25	mer 21.05.25
☞	Consegna	12 h	mer 21.05.25	mer 28.05.25

Figura 2 Gantt Preventivo

Questo è il Gantt Preventivo che abbiamo fatto per la pianificazione del nostro progetto, utilizziamo la metodologia Agile usando il framework Scrumban. Ecco l'analisi per ogni sprint:

2.4.1 Sprint 1

▲ Sprint 1	12 h
Schema E-R	1 h
Creazione Autenticazione	1 h
Activity Diagram	1 h
Diagramma delle attività	1 h
Template Pagina HTML	1 h
Design delle interfacce	1 h
Design dell'architettura di sistema	1 h
Configurazione ambiente di sviluppo	1 h
Analisi dei mezzi	1 h
Creazione DB	2 h
Use Case	1 h

Figura 3 Sprint No. 1 Preventivo

Secondo noi, queste 12 ore sarebbero state dedicate alla creazione delle fondamenta del progetto, mettendoci queste ore perché ne avevamo già avuto a che fare lo scorso progetto.

2.4.2 Sprint 2

▲ Sprint 2	12 h
View	2 h
▲ Impostazioni	6 h
Lingue	1.5 h
Tema	1.5 h
Suoni	1.5 h
Volume	1.5 h
Ricerca Frasi	2 h
Impostazione obbiettivo	2 h

Figura 4 Sprint No. 2 Preventivo

Pensavamo anche qua di metterci 12 ore, perché pensavamo che fossero cose relativamente semplici e con l'aiuto delle intelligenze artificiali per le view ci avremo messo ancora di meno.

2.4.3 Sprint 3

▲ Sprint 3	16 h
▲ statistiche	2 h
Prendere le varie statistiche	1 h
Salvare le stative	1 h
▲ Modalita partita	14 h
Caricamento pagine	3 h
Visualizzare accurateza	3 h
Vedere velocita di scrittura	2 h
Animazioni	2 h
Mostrare input tastiera	4 h

Figura 5 Sprint No. 3 Preventivo

Per questo sprint invece, qua si parla del gioco del singleplayer e abbiamo progettato di metterci due ore per quanto riguarda le statistiche e 14 per il resto, perché secondo noi, date le nostre conoscenze di php, sembravano cose che avevamo già visto, e anche essendo molte cose da fare, secondo ci avremo messo poco.

2.4.4 Sprint 4

▲ Sprint 4	12 h
▲ Multiplayer	12 h
Possibilità di entrare nella stanza	6 h
Creazione stanza	6 h

Figura 6 Sprint No. 4 Preventivo

Abbiamo progettato di metterci così tanto solo per entrare e creare la stanza, perché essendo che non sapevamo come implementare il multiplayer, sarebbe stata una fase difficile perché oltre all'implementazione ci sarebbe dovuto essere anche una parte di studio.

2.4.5 Sprint 5

▲ Sprint 5	12 h
▲ Multiplayer	12 h
Funzionamento della partita multiplayer	6 h
Salvataggio dati Partite	6 h

Figura 7 Sprint No. 5 Preventivo

Per questo sprint, abbiamo progettato di metterci così poco per tutto il multiplayer perché essendo che la parte di studio l'avevamo già fatta nello sprint precedente, sarebbe stata una cosa facile da attuare.

2.4.6 Sprint 6

Sprint 6	6 h
Funzionalità finali	3 h
Correzione vari bug	3 h

Figura 8 Sprint No. 6 Preventivo

L'ultimo sprint, abbiamo progettato di non metterci molto perché ci sarebbero stati solo i test finali e solo piccoli errori in tutto l'applicativo.

2.4.7 Consegna

Consegna	12 h
-----------------	-------------

Figura 9 Consegna Preventivo

Per la consegna del progetto abbiamo progettato 12 ore per finire i risultati dei test e fare l'ultima parte della documentazione.

2.5 Analisi dei mezzi

2.5.1 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 Chrome: 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

2.5.2 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

3 Progettazione

3.1 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.

Diagramma
dell'architettura

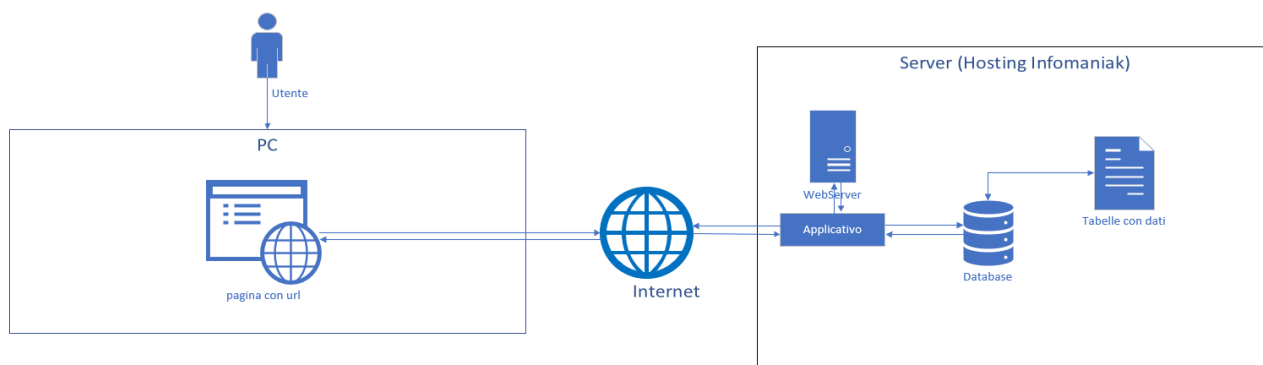


Figura 10 Schema delle strutture di sistema

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 registrati.

3.2 Design dei dati e database

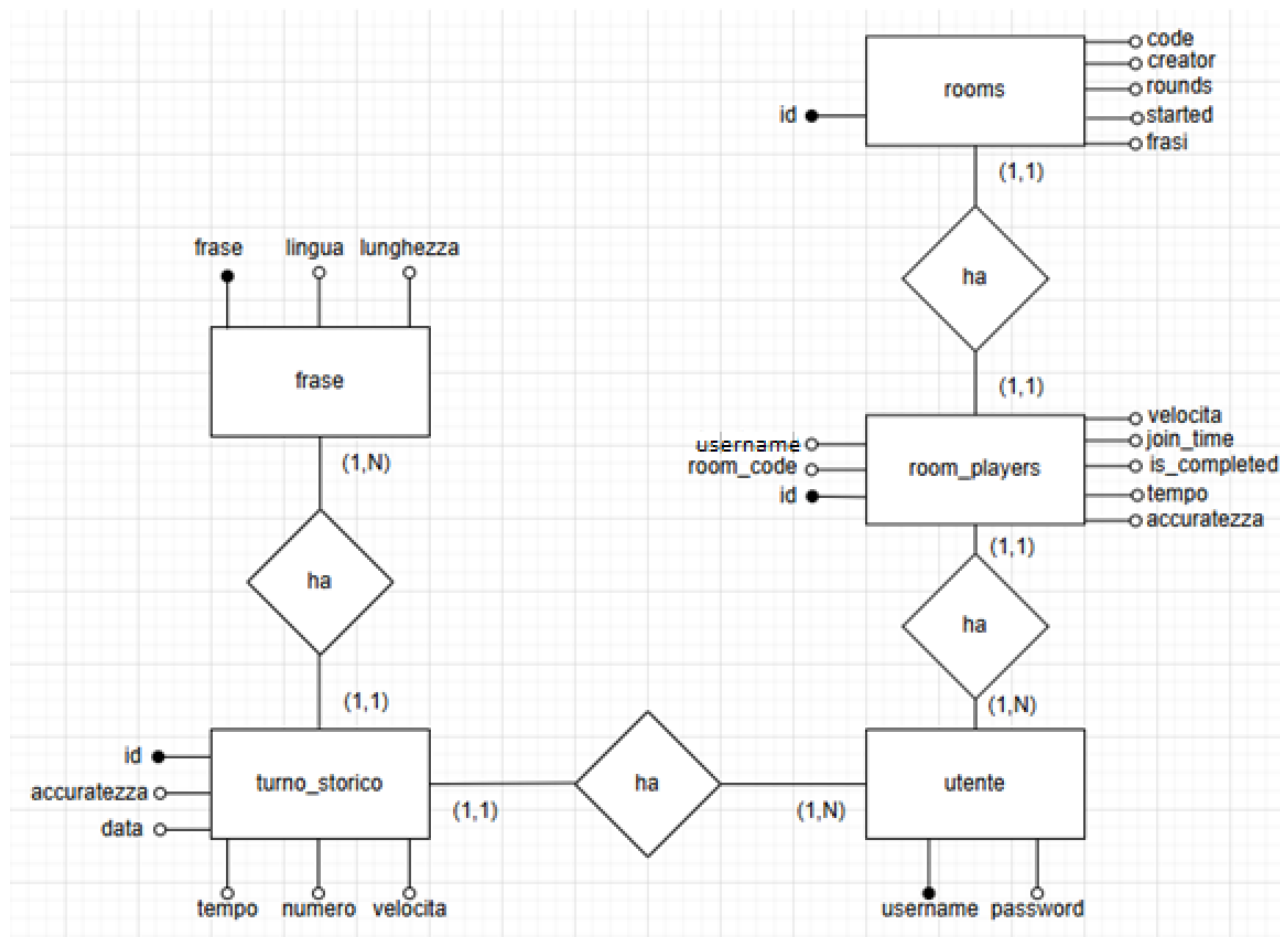


Figura 11 Schema E-R

Ho creato il database di nome dattiloking con quattro tabelle al suo interno, una tabella **“rooms”**, la quale ha l'id come identificatore, **“code”** che è il codice della stanza, **“creator”** che è lo username dell'utente che ha creato la stanza, **“round”** che è il numero di round della partita della stanza, **“started”** che indica se la partita è iniziata e **“frasi”** che contiene tutte le frasi utilizzate nei vari round della stanza.

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.

Infine esiste la tabella **“room_players”** correlata alla tabella **“rooms”**, la quale serve ad identificare quali utenti hanno fatto parte alle partite e le varie statistiche.

3.3 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.

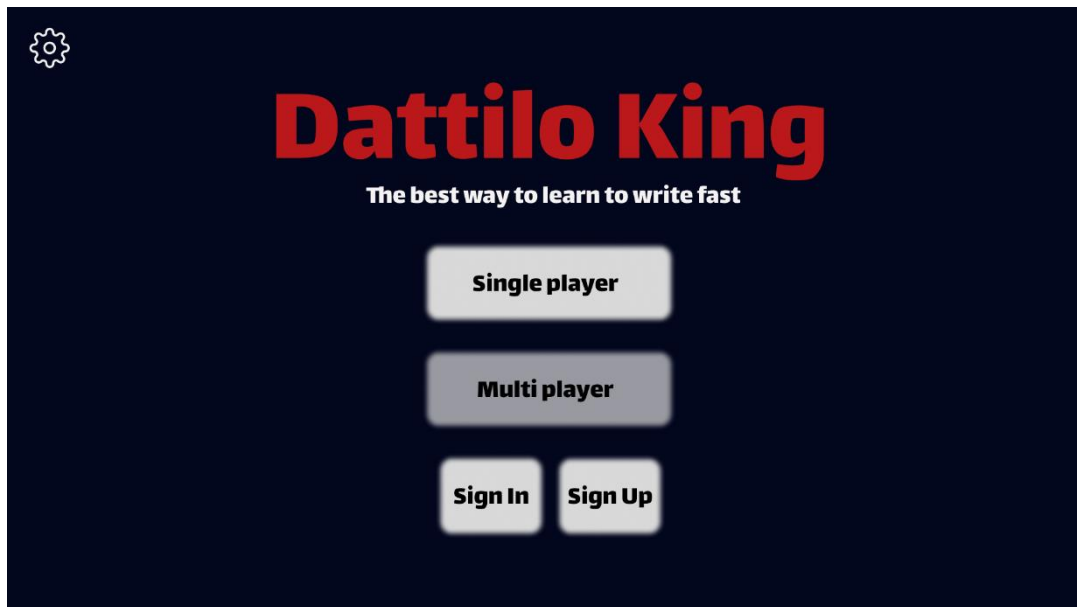


Figura 12 Pagina home senza essere loggati

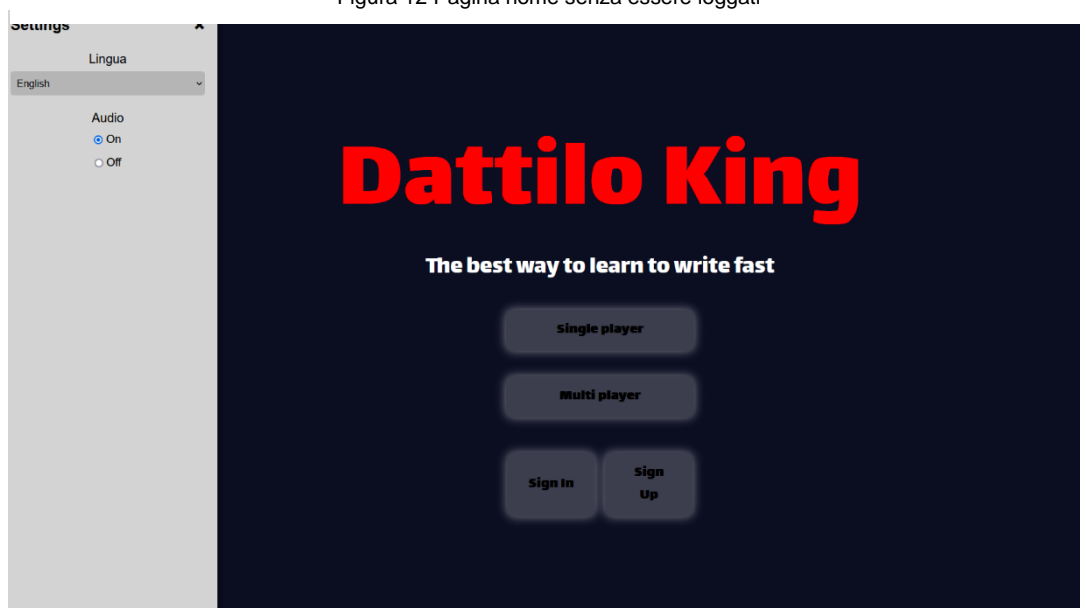
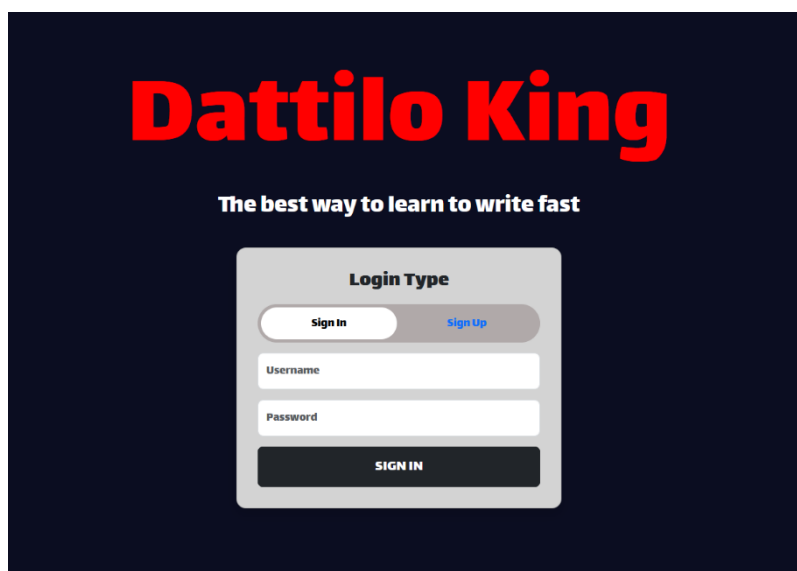


Figura 13 Impostazioni

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.



Dattilo King

The best way to learn to write fast

Login Type


[sign in](#) [sign up](#)

Username

Password

SIGN IN

Figura 14 Form di Login



Dattilo King

The best way to learn to write fast

Login Type

[sign in](#) [sign up](#)

Username

Password

Confirm Password

SIGN UP

Figura 15 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 giornaliero 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.

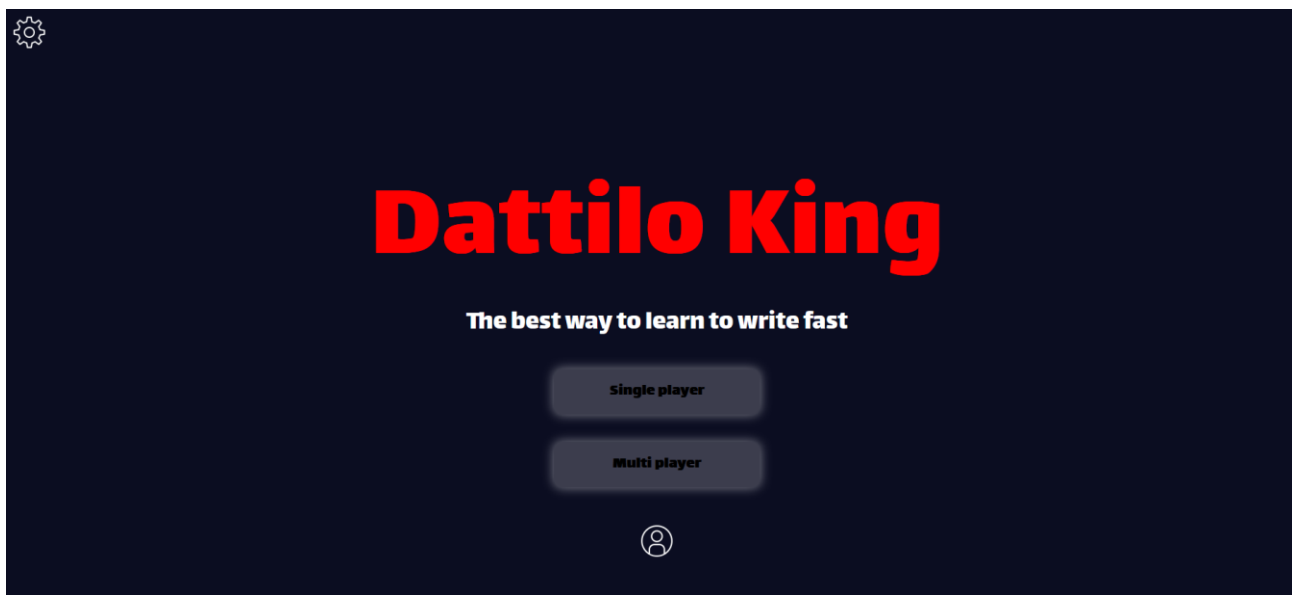


Figura 16 Pagina home da loggati

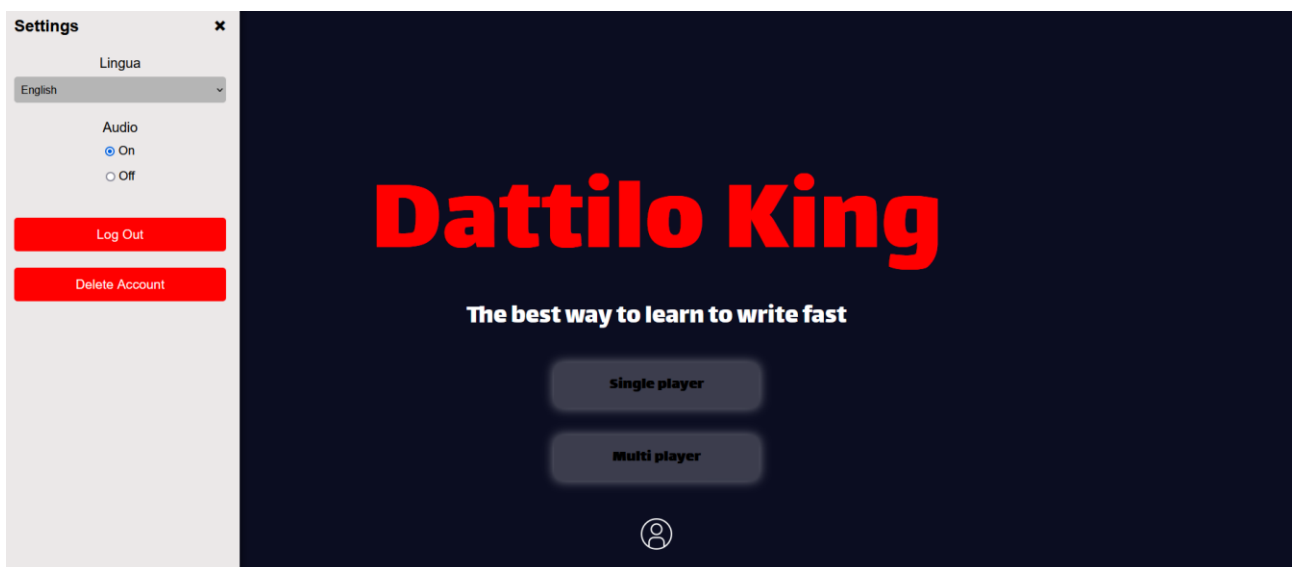


Figura 17 Impostazioni da loggato

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 18 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.

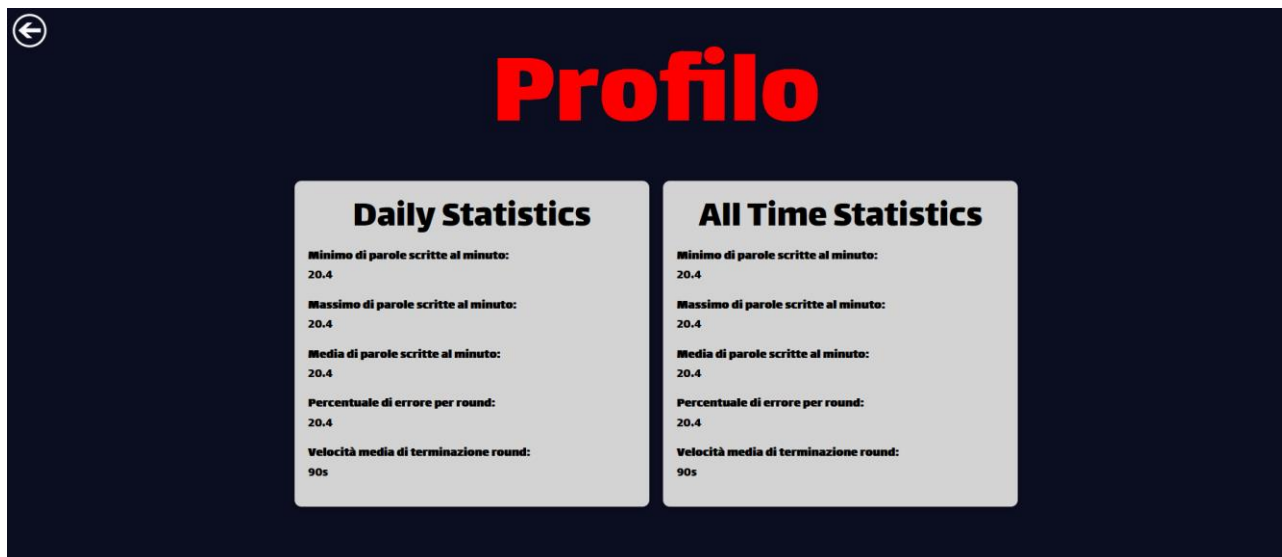


Figura 19 Statistiche personali

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.

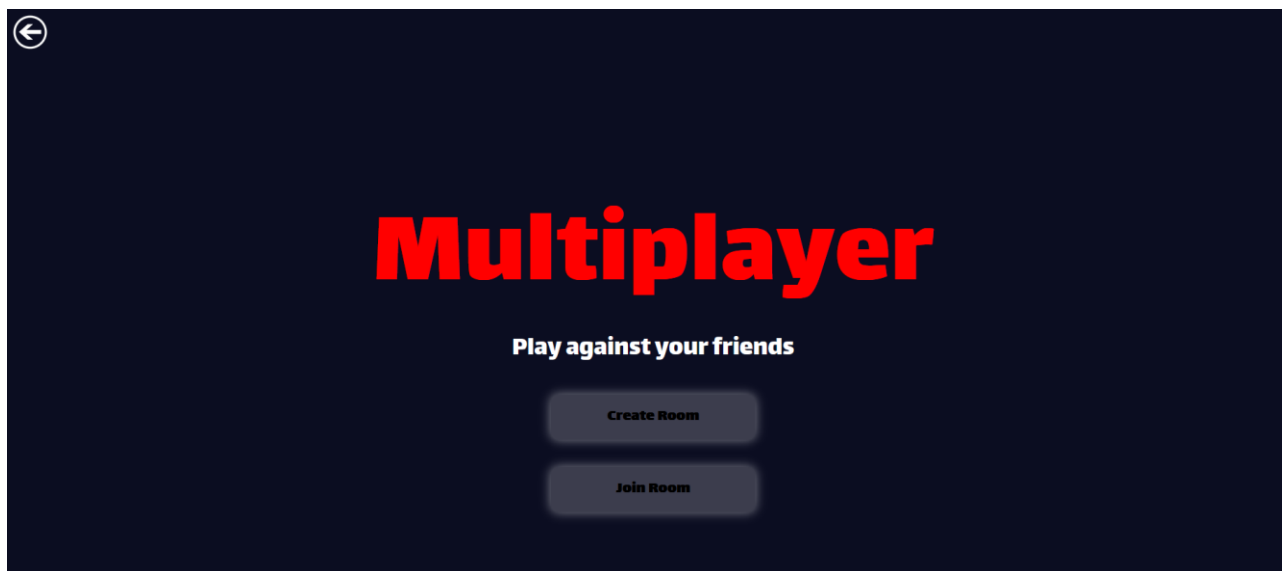


Figura 20 Multiplayer

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 21 Creazione stanza

Figura 22 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 23 Keyboard di gioco multiplayer

L'ultima interfaccia è la seguente, essa rappresenta la classifica della 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 24 Leaderboard multiplayer

3.4 Design procedurale

Descrive i concetti dettagliati dello sviluppo:

- Diagramma delle classi
- Diagrammi di flusso

3.4.1 Diagramma delle classi

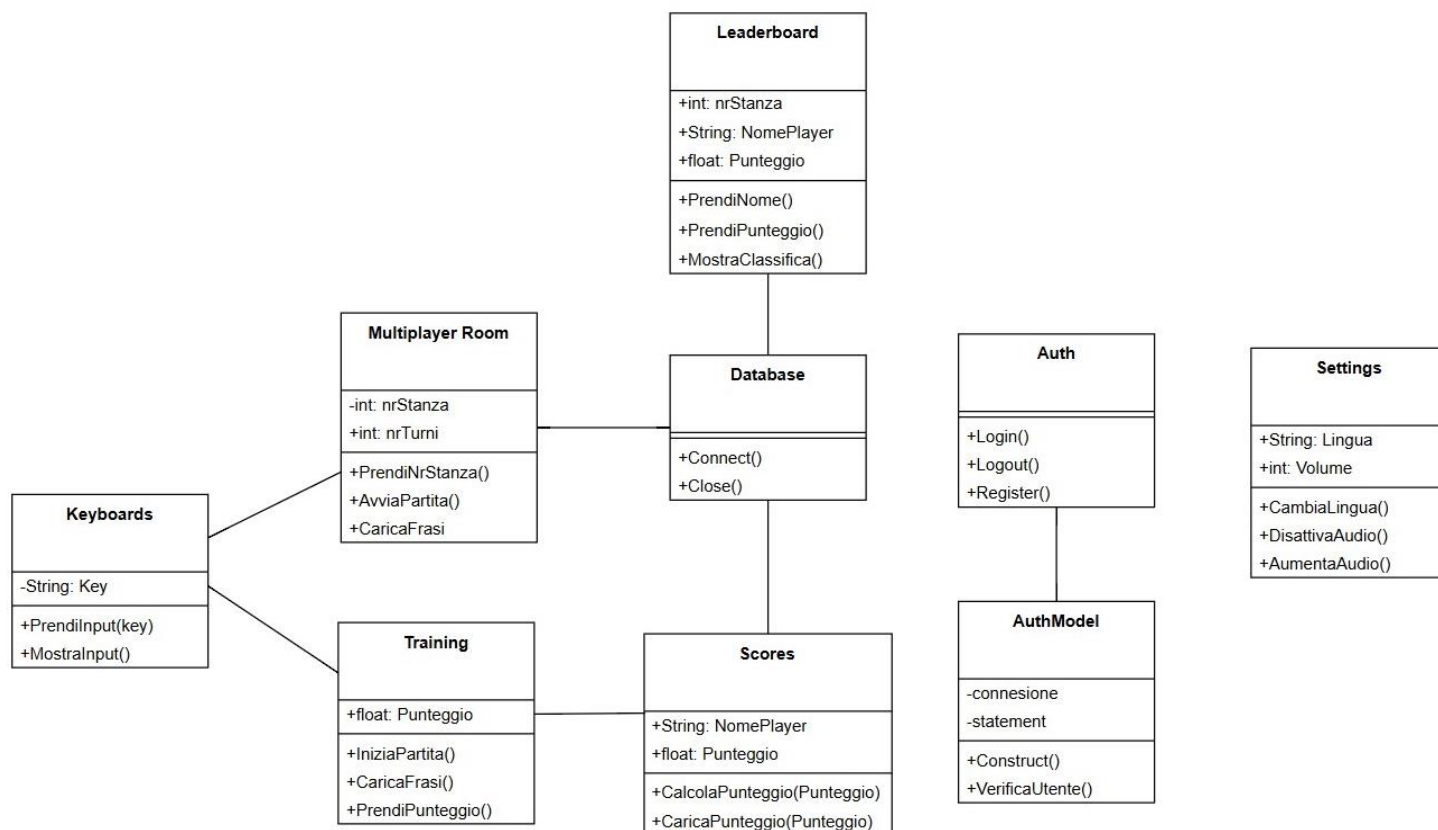


Figura 25 Diagramma delle classi

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, Leaderboard 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.

3.4.2 Diagramma di flusso

3.4.2.1 Generale

In allegato si trova lo schema completo: **ActivityDiagram\ADDattiloKing.vsd**

Il 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 caricata 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

3.4.2.2 Multiplayer

In allegato si trova lo schema completo: **ActivityDiagram\ADDattiloKingMultiplayer.vsd**

Questo 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 nuovo 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
- 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"**

4 Implementazione

4.1 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

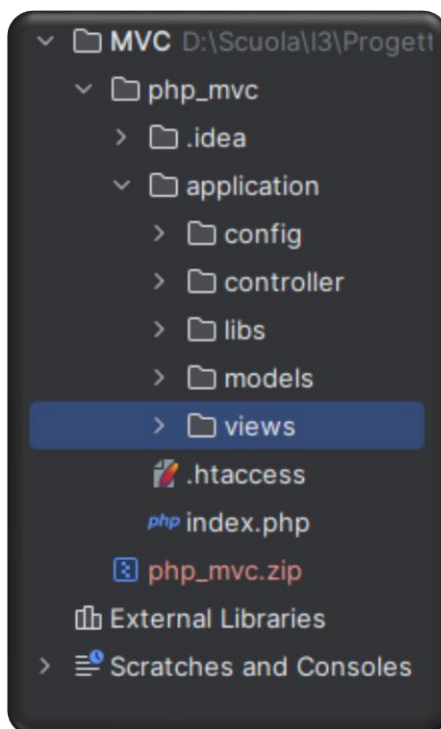


Figura 26 Struttura cartelle

4.2 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.

4.2.1 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 sfiorare il limite di 200 caratteri se si 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(frase_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.")
```

4.2.2 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}")
```

4.3 Implementazione Database

Dopo aver creato il Database 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 installato il servizio di MYSQL.

```
sudo apt update
```

```
sudo apt install mysql-server -y
```

Dopo abbiamo fatto una copia 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:/home/user
```

```
scp D:/dattiloking.sql user@10.100.1.17:/home/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_frasei.sql user@10.100.1.17:/home/user
```

```
SOURCE /home/user/insert_frasei.sql;
```

4.4 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;
    }
}
```


4.5 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;
}
```

4.6 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;
}
```

4.7 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;
}
```

4.8 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";}}
```

4.9 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;
});
```

4.10 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));}
```

4.11 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']);
    }
}
```


4.12 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;}
```

4.13 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 sì 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 });};
```

4.14 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');
}
```

4.15 Implementazione generazione codice stanza

Quando si apre la view della stanza multiplayer si genererà un codice univoco alfanumerico di 6 caratteri. Per farlo la funzione mescola una stringa contenente tutte le lettere e le cifre crea una stringa di 6 cifre e la mette in maiuscolo per sicurezza.

Dopo ciò verrà creata la stanza nel DB passandogli codice generato DA FINIREEEEEEEEEEEEEEE

```
<?php
if (session_status() == PHP_SESSION_NONE) {
    session_start();
}require_once 'application/models/RoomMapper.php';

// Funzione per generare un codice stanza univoco
require_once 'application/models/RoomMapper.php';

if ($_SERVER["REQUEST_METHOD"] == "POST" && isset($_POST["code"])) {
    $codice = $_POST["code"];
    $roomMapper = new RoomMapper();

    if ($roomMapper->roomExists($codice)) {
        $roomMapper->aggiungiPartecipante($codice, $_SESSION['username']);
        $_SESSION['code'] = $codice;
        $_SESSION['creatore'] = false;
        header("Location: " . URL . "play/multiPlayerRoomList");
        exit;
    } else {
        echo "La stanza non esiste.";
    }
}
$_SESSION['code'] = $codice;
$_SESSION['creatore'] = true;
?>
```

4.16 Mostrare giocatori nella room

Quando si entra c'è uno script che viene richiamato ogni due secondi che prende il tuo nome per sapere chi sei e poi fa una chiamata al controller che prende tutti i player nella stessa room, e li aggiunge uno a uno in un div aggiungendo "(tu)" al tuo nome e colora di giallo il nome del creatore visto che il fetch restituisce anche quello. Così ogni giocatore potrà vedere i vari player e capire chi è lui e chi è il creatore.

```
<script>
    // Passa il valore di $_SESSION['username'] a una variabile JavaScript
    const currentUsername = '<?php echo $_SESSION['username']; ?>';
    setInterval(() => {
        fetch('<?php echo URL ?>play/getRoomPlayers?code=<?php echo $codice
?>')
            .then(res => res.json())
            .then(data => {
                const container = document.getElementById('playerContainer');
                container.innerHTML = '';
                const creator = data.creator; // Recupera il nome del creatore
                data.players.forEach(player => {
                    const playerDiv = document.createElement('div');
                    playerDiv.className = 'player';

                    // Confronta la variabile JavaScript currentUsername
                    if (player === currentUsername) {
                        playerDiv.textContent = player + " (TU)";
                    } else {
                        playerDiv.textContent = player;
                    }

                    // Evidenzia il nome del creatore in giallo
                    if (player === creator) {
                        playerDiv.className += ' creator';
                    }

                    container.appendChild(playerDiv);
                });
            });
    }, 2000);
</script>
```

5 Test

5.1 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:	TC-001	Nome:	Visualizzazione testo
Riferimento:	REQ-001		
Descrizione:	Visualizzazione della frase nel gioco Singleplayer e Multiplayer		
Prerequisiti:	<ul style="list-style-type: none"> - 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:	TC-002	Nome:	Visualizzazione classifica Multiplayer
Riferimento:	REQ-002		
Descrizione:	Visualizzazione della classifica e punteggi dei vari utenti		
Prerequisiti:	<ul style="list-style-type: none"> - Si necessita dell'interfaccia di gioco - Si necessita del DB 		
Procedura:	Dopo aver finito i round per la modalità Multiplayer, appare la classifica		
Risultati attesi:	Quando sono finiti i round appare la classifica di tutti i giocatori		

Test Case:	TC-003	Nome:	Percentuale correttezza
Riferimento:	REQ-003		
Descrizione:	Calcolo della percentuale delle lettere scritte correttamente		
Prerequisiti:	<ul style="list-style-type: none"> - 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 commesso 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:	TC-004	Nome:	Velocità scrittura
Riferimento:	REQ-004		
Descrizione:	Calcolo della velocità media in cui si scrive una frase		
Prerequisiti:	<ul style="list-style-type: none"> - 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:	TC-005	Nome:	Tempo scrittura
Riferimento:	REQ-005		
Descrizione:	Calcolo del tempo in cui viene finita una frase		
Prerequisiti:	<ul style="list-style-type: none"> - 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:	TC-006	Nome:	Visualizzazione e animazione grafica della scrittura
Riferimento:	REQ-006		
Descrizione:	Mostra la tastiera e l'animazione degli input dell'utente		
Prerequisiti:	<ul style="list-style-type: none"> - 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:	TC-007	Nome:	Selezione della lingua
Riferimento:	REQ-007		
Descrizione:	L'utente ha la possibilità di scegliere la lingua		
Prerequisiti:	<ul style="list-style-type: none"> - 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 nella lingua che si ha selezionato.		

Test Case:	TC-008	Nome:	Seleziona tema di sfondo
Riferimento:	REQ-008		
Descrizione:	L'utente ha la possibilità di selezionare un tema di sfondo		
Prerequisiti:	<ul style="list-style-type: none"> - 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:	TC-009	Nome:	Multiplayer
Riferimento:	REQ-009		
Descrizione:	Permette di giocare da più dispositivi simultaneamente		
Prerequisiti:	<ul style="list-style-type: none"> - 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:	TC-010	Nome:	Gestione audio
Riferimento:	REQ-010		
Descrizione:	L'utente ha la possibilità di abilitare o disattivare l'audio		
Prerequisiti:	<ul style="list-style-type: none"> - 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:	TC-011	Nome:	Visualizzazione punteggi Singleplayer
Riferimento:	REQ-011		
Descrizione:	Visualizza i punteggi giornalieri e di tutti i tempi dell'utente		
Prerequisiti:	<ul style="list-style-type: none"> - 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:	TC-012	Nome:	Scelta turni Multiplayer
Riferimento:	REQ-012		
Descrizione:	Creatore della stanza multiplayer sceglie il numero di round		
Prerequisiti:	<ul style="list-style-type: none"> - 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 utenti della stanza parteciperanno al numero di round scelti dal creatore		

Test Case:	TC-013	Nome:	Cancellazione utente
Riferimento:	REQ-013		
Descrizione:	L'utente ha la possibilità di cancellare l'utente loggato		
Prerequisiti:	<ul style="list-style-type: none"> - 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:	TC-014	Nome:	Controllo battitura
Riferimento:	REQ-014		
Descrizione:	Esegue controlli nella battitura		
Prerequisiti:	<ul style="list-style-type: none"> - 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.		

5.2 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
TC-006	Passato
TC-007	Passato
TC-008	Fallito
TC-009	Passato
TC-010	Passato
TC-011	Parzialmente Passato
TC-012	Passato
TC-013	Passato
TC-014	Passato

5.3 Mancanze/limitazioni conosciute

Durante lo sviluppo di DattiloKing sono emerse alcune limitazioni e funzionalità mancanti, dovute principalmente alla complessità tecnica, alle tempistiche strette e a problemi tecnici esterni (come l'indisponibilità della rete nera il giorno della consegna finale). I principali elementi mancanti sono le seguenti:

- **Classifica Multiplayer assente:** non è stato possibile completare l'implementazione della leaderboard per la modalità multiplayer. I risultati di fine partita non vengono memorizzati né confrontati tra i giocatori.
- **Audio non implementato in multiplayer:** sebbene l'audio sia presente nella modalità singleplayer, non è stato integrato nel multiplayer, a causa dell'indisponibilità della rete nera.
- **Crash del gioco in multiplayer:** durante il multiplayer, dopo l'avvio della partita, il gioco si blocca dopo poco tempo. Questo è dovuto a un numero eccessivo di query inviate al server in breve tempo, causate dal frequente utilizzo del metodo fetch() e dalla mancanza di un sistema più efficiente di gestione della comunicazione. Il problema è stato accentuato dalla mancata implementazione dei WebSocket, che avrebbero permesso un'interazione in tempo reale più stabile e performante, che però non abbiamo implementato perché sarebbe stato richiesto una fase di studio in più che non abbiamo avuto modo di fare a causa del tempo ristretto.
- **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.

6 Consuntivo

▣ DattiloKing	177.39 h	mer 29.01.25	mer 28.05.25
▣ Teoria Agile	3 h	mer 29.01.25	mer 29.01.25
Concetti Base	3 h	mer 29.01.25	mer 29.01.25
▣ Progettazione	4 h	mer 29.01.25	mer 12.02.25
Creazione git	0.5 h	mer 29.01.25	mer 29.01.25
Creazione Trello	0.5 h	mer 29.01.25	mer 29.01.25
Setup Trello	0.5 h	mer 29.01.25	mer 29.01.25
Requisiti	2 h	mer 29.01.25	mer 12.02.25
Gannt	0.5 h	mer 29.01.25	mer 29.01.25

Figura 27 Fase di teoria e progettazione Consuntivo

Nella Teoria Agile nessuna differenza, il tempo dedicato è stato esattamente come previsto.

Nella fase di progettazione un leggero aumento delle ore (+1) dovuto alla maggiore attenzione nei dettagli in fase di progettazione

Implementazione	170.39 h	mer 05.02.25	mer 28.05.25
Sprint 1	31.63 h	mer 05.02.25	mer 26.02.25
Schema E-R	3 h	mer 05.02.25	mer 05.02.25
Creazione Autenticazione	2.25 h	mer 19.02.25	mer 19.02.25
Activity Diagram	3 h	mer 05.02.25	mer 12.02.25
Class Diagram	3 h	mer 12.02.25	mer 12.02.25
Template Pagina HTML	1.5 h	mer 05.02.25	mer 05.02.25
Design delle interfacce	3 h	mer 12.02.25	mer 12.02.25
Design dell'architettura di sistema	0.75 h	mer 05.02.25	mer 05.02.25
Configurazione ambiente di sviluppo	0.75 h	mer 05.02.25	mer 05.02.25
Analisi dei mezzi	1.13 h	mer 05.02.25	mer 05.02.25
Creazione pagina per singleplayer	2 h	mer 19.02.25	mer 19.02.25
Tastiera a schermo	2.25 h	mer 26.02.25	mer 26.02.25
Creazione DB	1.5 h	mer 19.02.25	mer 19.02.25
Use Case	0.75 h	mer 05.02.25	mer 19.02.25
Generazione frasi	3 h	mer 19.02.25	mer 19.02.25
Accesso al db per le frasi	2.25 h	mer 26.02.25	mer 26.02.25
Implementazione backend/frontend	0.75 h	mer 19.02.25	mer 19.02.25
Prima versione logout/delete	0.75 h	mer 26.02.25	mer 26.02.25

Figura 28 Sprint No. 1 Consuntivo

Lo Sprint 1 ha richiesto significativamente più tempo del previsto, con un aumento di quasi 20 ore. Le attività pianificate sono state completate, ma sono state aggiunte ulteriori task non previste inizialmente, come il design delle interfacce e dell'architettura di sistema, la configurazione dell'ambiente di sviluppo, e altre attività di implementazione.

▲ Sprint 2	56.88 h	mer 12.03.25	mer 09.04.25
Implementazione template in MVC	7.5 h	mer 12.03.25	mer 26.03.25
Pagina lista multiplayer	0.75 h	mer 26.03.25	mer 26.03.25
Sistemato Login	3 h	mer 12.03.25	mer 12.03.25
UML	2.25 h	mer 12.03.25	mer 12.03.25
Visualizzazione frasi	3 h	mer 12.03.25	mer 12.03.25
LogOut e Delete account	1.5 h	mer 12.03.25	mer 12.03.25
Generazione frasi	2.25 h	mer 12.03.25	mer 26.03.25
Protocollo di test	6.75 h	mer 12.03.25	mer 26.03.25
Scelta lingua	3.75 h	mer 12.03.25	mer 12.03.25
Pagina tutorial	0.75 h	mer 12.03.25	mer 12.03.25
Audio	7.75 h	mer 12.03.25	mer 09.04.25
Generazione file sql	0.75 h	mer 26.03.25	mer 26.03.25
Turno utente	4.5 h	mer 12.03.25	mer 09.04.25
Calcolo/Salvataggio statistiche	6 h	mer 26.03.25	mer 26.03.25
Rifinitura SignIn/SignUp	3 h	mer 02.04.25	mer 02.04.25
Scelta Lingua	2.25 h	mer 02.04.25	mer 02.04.25
Deploy DB	1.13 h	mer 09.04.25	mer 09.04.25

Figura 29 Sprint No. 2 Consuntivo

Lo Sprint 2 ha visto un aumento significativo delle ore di lavoro, con l'aggiunta di molte attività non previste inizialmente. Questo indica che la fase di implementazione ha richiesto più lavoro del previsto, probabilmente a causa di una maggiore complessità o di aggiustamenti continui.

▲ Sprint 3	27 h	mer 09.04.25	mer 30.04.25
Deploy Sito in VM	3 h	mer 16.04.25	mer 16.04.25
Studio multiplayer	1.5 h	mer 09.04.25	mer 09.04.25
Rifinitura applicativo	13.5 h	mer 09.04.25	mer 16.04.25
▲ statistiche	3 h	mer 30.04.25	mer 30.04.25
Visualizzazione punteggi	1.5 h	mer 30.04.25	mer 30.04.25
Prendere le varie statistiche	1.5 h	mer 30.04.25	mer 30.04.25
▲ Modalita partita	6 h	mer 16.04.25	mer 30.04.25
Creazione stanza	3 h	mer 16.04.25	mer 16.04.25
Join stanza	3 h	mer 16.04.25	mer 16.04.25
<u>Milestone 1</u>	0.75 h	mer 09.04.25	mer 09.04.25

Figura 30 Sprint No. 3 Consuntivo

Lo Sprint 3 ha richiesto 11 ore in più rispetto al preventivo. Sono state aggiunte diverse attività non previste inizialmente, come il deploy del sito in una VM, lo studio del multiplayer, e la rifinitura dell'applicativo. Inoltre, alcune attività previste sono state modificate o estese, come la modalità partita che ha incluso la creazione e la gestione delle stanze per il multiplayer.

▲ Sprint 4	15.25 h	mer 30.04.25	mer 07.05.25
Documentazione Socket	4.5 h	mer 30.04.25	mer 07.05.25
Prova implementazione WebSocket	3 h	mer 07.05.25	mer 07.05.25
Rifinitura applicativo	3.75 h	mer 07.05.25	mer 07.05.25
▲ Multiplayer	4 h	mer 30.04.25	mer 07.05.25
Controllo codice stanza	2.25 h	mer 07.05.25	mer 07.05.25
Sistemato join in stanza	0.75 h	mer 07.05.25	mer 07.05.25
Visualizzazione round e giocatori	1 h	mer 07.05.25	mer 07.05.25

Figura 31 Sprint No. 4 Consuntivo

Lo Sprint 4 ha richiesto 3.25 ore in più rispetto al preventivo. Le attività previste per il multiplayer sono state modificate e ridotte in termini di ore, con un focus maggiore su aspetti specifici come il controllo del codice della stanza e la visualizzazione dei round e dei giocatori. Inoltre, sono state aggiunte attività non previste inizialmente, come la documentazione dei socket, la prova di implementazione dei WebSocket, e la rifinitura dell'applicativo.

▲ Sprint 5	30.63 h	mer 07.05.25	mer 21.05.25
▲ Multiplayer	30.63 h	mer 14.05.25	mer 28.05.25
Rifinitura salvataggio frasi in db, stampa di esse	6 h	mer 21.05.25	mer 21.05.25
Cambiamento round per tutti gli utenti della stanza	3.75 h	mer 14.05.25	mer 14.05.25
Debug	1.75 h	mer 14.05.25	mer 21.05.25
Selezione lingua	1.88 h	mer 14.05.25	mer 14.05.25
Eliminazione utente	2.25 h	mer 21.05.25	mer 21.05.25
Gantt	3 h	mer 14.05.25	mer 21.05.25
Sistemato multiplayer	3.5 h	mer 21.05.25	mer 21.05.25
Documentazione	7 h	mer 28.05.25	mer 28.05.25
Implementazione Audio	1.5 h	mer 21.05.25	mer 21.05.25
Milestone 2	0.75 h	mer 14.05.25	mer 14.05.25

Figura 32 Sprint No. 5 Consuntivo

Lo Sprint 5 ha richiesto 18.63 ore in più rispetto al preventivo. Le attività previste per il multiplayer sono state estese e modificate, con un focus maggiore su aspetti specifici come la rifinitura del salvataggio delle frasi nel database, il cambiamento del round per tutti gli utenti, il debug, la selezione della lingua, e l'implementazione audio. Inoltre, sono state aggiunte attività non previste inizialmente, come la documentazione e la gestione del Gantt.

▲ Sprint 6	28 h	mer 21.05.25	mer 28.05.25
Sistemato multiplayer	8 h	mer 21.05.25	mer 21.05.25
Documentazione	20 h	mer 21.05.25	mer 21.05.25
Consegna	0.5 h	mer 28.05.25	mer 28.05.25

Figura 33 Sprint No. 6 Consuntivo

Lo Sprint 6 ha richiesto 22 ore in più rispetto al preventivo. Le attività previste per le funzionalità finali e la correzione dei bug non sono state eseguite come pianificato. Invece, sono state aggiunte attività non previste inizialmente, come la sistemazione del multiplayer e una documentazione estesa.

7 Conclusioni

Questo progetto è stato pensato e realizzato per tutte le persone che vogliono divertirsi con un gioco di dattilografia capace di mettere realmente alla prova la propria velocità e precisione sulla tastiera. L'interfaccia semplice e intuitiva lo rende accessibile anche ai meno esperti, ma la vera sfida sta nel migliorarsi costantemente, battendo i propri record e perfezionando la digitazione.

Uno degli elementi più ambiziosi del progetto è stata la modalità multiplayer, pensata per consentire agli utenti di sfidare amici o altri partecipanti in tempo reale. Sebbene questa funzionalità sia stata implementata e funzionante nelle sue componenti principali, non è stato possibile completare la classifica finale né integrare correttamente l'audio, e il gioco può bloccarsi a causa del numero eccessivo di query inviate al server durante i round. Questo problema è dovuto all'uso intensivo di `fetch()` e alla mancanza dei WebSocket, che non siamo riusciti a introdurre per via dei tempi stretti e della poca conoscenza con la tecnologia.

Nonostante queste difficoltà, il progetto mantiene un forte valore educativo: si è dimostrato utile per migliorare la scrittura a tastiera, stimolando gli utenti a lavorare su velocità, accuratezza e concentrazione, abilità sempre più importanti in ambito professionale.

Per noi, questo progetto rappresenta molto più di un semplice gioco. È il nostro primo lavoro strutturato secondo il modello Agile, affrontato in team dalla progettazione allo sviluppo fino ai test. Abbiamo dovuto superare difficoltà concrete, inclusi imprevisti tecnici come la rete nera non funzionante nel giorno di consegna, ma tutto ciò ha reso l'esperienza ancora più formativa. Abbiamo acquisito nuove competenze pratiche e una maggiore consapevolezza di cosa significhi collaborare in un contesto reale di sviluppo.

7.1 Sviluppi futuri

In futuro, DattiloKing potrebbe essere migliorato con l'integrazione dei WebSocket, che permetterebbero una gestione del multiplayer in tempo reale molto più stabile ed efficiente. Questo sarebbe particolarmente utile per risolvere i problemi attuali legati al blocco della partita durante i round multiplayer, causato dal sovraccarico di richieste al server tramite `fetch()`. Un altro sviluppo importante riguarda la classifica del multiplayer, attualmente assente: implementarla permetterebbe agli utenti di confrontare i propri risultati finali in modo strutturato e competitivo. Anche la gestione dell'audio in multiplayer, al momento non disponibile, potrebbe essere aggiunta per offrire un'esperienza più coerente tra le due modalità di gioco. Per l'interfaccia, si prevede di completare il sistema di selezione del tema di sfondo, già presente tra i requisiti ma non implementato. Ulteriori personalizzazioni come avatar, profili utente modificabili e un sistema di progressione con livelli e statistiche dettagliate aumenterebbe la motivazione degli utenti nel lungo termine. Infine, dal punto di vista dell'accessibilità, sarebbe bello implementare più lingue possibili per coinvolgere giocatori da tutto il mondo.

7.2 Considerazioni personali

7.2.1 Amos Haefliger

L'esperienza di sviluppo del progetto DattiloKing è stata per me estremamente formativa e stimolante, sia dal punto di vista tecnico che collaborativo. Lavorare in un gruppo coeso e ben organizzato, secondo i principi della metodologia Agile, ci ha permesso di affrontare le varie fasi progettuali con grande flessibilità, migliorando costantemente il prodotto grazie a feedback continui e cicli di sviluppo iterativi. DattiloKing è un'applicazione web innovativa, pensata per potenziare le competenze di dattilografia degli utenti attraverso un'esperienza di gioco coinvolgente, sia in modalità singleplayer che multiplayer. Il nostro obiettivo era quello di colmare una lacuna evidente nel mercato: sebbene esistano altri software per migliorare la velocità di scrittura, pochi offrono la possibilità di sfidare i propri amici online in un ambiente moderno, dinamico e intuitivo. Con DattiloKing, abbiamo creato un'interfaccia pulita e reattiva, accessibile anche ai meno esperti, che rende l'apprendimento più efficace e divertente. Personalmente, ho apprezzato molto l'approccio Agile perché ci ha permesso di gestire efficacemente le priorità, adattarci ai cambiamenti in corsa e valorizzare le competenze individuali all'interno del gruppo. Ogni sprint è stato un'occasione per migliorare, per ascoltare i suggerimenti reciproci e per affrontare le sfide tecniche con spirito collaborativo. Ritengo che questo progetto sia un esempio concreto di come la sinergia tra organizzazione, creatività e tecnologia possa portare a un risultato di qualità, destinato a crescere e a distinguersi sul mercato.

7.2.2 Nemanja Zecevic

Ritengo che lavorare a questo progetto sia stata un'esperienza formativa la quale mi ha permesso anche di avere un'idea più chiara su come possa sembrare una vera esperienza lavorativa. Ho imparato a lavorare meglio in gruppo e gestire possibili conflitti interni che si possono creare durante l'evoluzione di un progetto. Poter mettere in pratica le mie conoscenze assieme a quelle dei compagni e riuscire a gestire un progetto fino ad arrivare ad un risultato concreto reputo sia la cosa più soddisfacente di questa esperienza. Durante il progetto abbiamo avuto diverse volte dei conflitti sulle idee e sul come fare le cose ma siamo sempre riusciti a risolverli trovando compromessi, questo ritengo sia stato molto importante. Spesso abbiamo avuto anche problematiche legate a fattori che non potevamo controllare o software che non sapevamo ancora utilizzare, come può essere GitHub che più di una volta ci ha fatto perdere lavori di documentazione, oppure la rete nera che ha avuto problemi di funzionamento nel momento più importante, ma posso comunque ritenermi soddisfatto del risultato che abbiamo ottenuto, essendo che comunque siamo riusciti a finire praticamente tutto, a parte qualche dettaglio e qualcosa che avremmo potuto implementare meglio per rendere l'applicativo più fluido e meglio gestito. Nel complesso sono contento di questa esperienza, perché mi ha permesso di imparare molto e di lavorare in team.

7.2.3 Robin Sartore

Questo progetto mi ha insegnato tanto, come il lavoro in team che a volte può essere altalenante, ma l'importante è che ho capito come trovare sempre una soluzione comune. È stato interessante capire cosa vuol dire far parte di un team. Il progetto in sé mi è sembrato interessante da subito perché non avrei mai pensato di lavorare a un sito di dattilografia ed è stato interessante vedere le nostre soluzioni. Una cosa che sono sicuro che avremmo potuto migliorare è la gestione del tempo, visto che siamo arrivati quasi alla fine e avevamo poco tempo per fare il multiplayer, adottando così un metodo poco funzionale a livello di prestazioni ma che ci avrebbe fatto risparmiare tempo. Infine sono contento del risultato che abbiamo ottenuto anche se sono sicuro che si sarebbe potuto lavorare ancora per migliorarlo, e magari, in futuro prenderemo in considerazione questa opzione. Spero che questo progetto servirà a qualcuno per poter migliorare la sua velocità di scrittura dato che questo era l'obiettivo del progetto.

Nel progetto abbiamo fatto tanti errori a volte non per colpa nostra; ad esempio GitHub sovrascriveva file su cui lavoravamo insieme e dovevamo ripristinarli, rischiando di perderli; un altro problema è che quando qualcuno finiva quello che doveva fare, molte volte stava fermo senza fare niente, facendo perdere molto tempo essenziale che avremmo potuto usare per implementare le cose non finite, ma andando avanti abbiamo capito come evitare questi problemi e siamo cresciuti insieme al progetto migliorandoci. Questo progetto mi ha preparato molto per un futuro in cui lavorerò in un team di persone perché penso che arriverei più preparato sapendo cosa mi aspetta e non farei errori commessi in questo progetto.

7.2.4 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. Nonostante i vari errori commessi durante il progetto, come l'organizzazione non ottimale, creare un qualcosa da zero 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 importante passo nel nostro percorso di crescita, sia come sviluppatori che come team.

8 Indice delle Figure

Figura 1 Schema Use Case.....	10
Figura 2 Gantt Preventivo.....	11
Figura 3 Sprint No. 1 Preventivo	12
Figura 4 Sprint No. 2 Preventivo	12
Figura 5 Sprint No. 3 Preventivo	13
Figura 6 Sprint No. 4 Preventivo	13
Figura 7 Sprint No. 5 Preventivo	13
Figura 8 Sprint No. 6 Preventivo	14
Figura 9 Consegna Preventivo	14
Figura 10 Schema delle strutture di sistema	16
Figura 11 Schema E-R	17
Figura 13 Pagina home senza essere loggati	18
Figura 14 Impostazioni	18
Figura 15 Form di Login.....	19
Figura 16 Form di SignUp.....	19
Figura 17 Pagina home da loggati.....	20
Figura 18 Impostazioni da loggato	20
Figura 19 Keyboard di gioco singleplayer	21
Figura 20 Statistiche personali	22
Figura 21 Multiplayer	22
Figura 22 Creazione stanza.....	23
Figura 23 Inserimento codice, unione stanza.....	23
Figura 24 Keyboard di gioco multiplayer	24
Figura 25 Leaderboard multiplayer.....	24
Figura 26 Diagramma delle classi	25
Figura 27 Struttura cartelle	27
Figura 28 Fase di teoria e progettazione Consuntivo.....	53
Figura 29 Sprint No. 1 Consuntivo	54
Figura 30 Sprint No. 2 Consuntivo	55
Figura 31 Sprint No. 3 Consuntivo	56
Figura 32 Sprint No. 4 Consuntivo	56
Figura 33 Sprint No. 5 Consuntivo	57
Figura 34 Sprint No. 6 Consuntivo	57

9 Glossario

Termine	Descrizione
AJAX (Asynchronous JavaScript And XML)	Tecnica che permette di eseguire richieste ed ottenere dati da una pagina web in modo asincrono.
Bootstrap	Framework CSS open-source che facilita la creazione di interfacce web responsive e moderne, fornendo componenti già pronti come tasti, form e griglie.
CSS	Linguaggio usato per definire l'aspetto grafico delle pagine HTML.
Fetch	Metodo per eseguire richieste HTTP in JavaScript.
HTML	Linguaggio di markup utilizzato per la struttura delle pagine web.
JS (JavaScript)	Linguaggio di scripting usato per rendere dinamiche le pagine web.
MVC	Architettura software che separa Model, View e Controller per una gestione modulare del codice.
MySQL	Sistema di gestione di database relazionali utilizzato nel progetto.
PHP	Linguaggio di scripting lato server per creare pagine web.
WebSocket	Protocollo per comunicazioni bidirezionali in tempo reale, non implementato nel progetto.

10 Sitografia

1. <https://coolors.co/palette/0b090a-161a1d-660708-a4161a-ba181b-e5383b-b1a7a6-d3d3d3-f5f3f4-ffffff>
2. <https://icon-sets.iconify.design/mdi/>
3. <https://chatgpt.com/>
4. <https://www.deepseek.com/>
5. <https://dev.to/robertobutti/websocket-with-php-4k2c>
6. <https://www.php.net/manual/en/book.sockets.php>
7. <https://stackoverflow.com/questions/941744/keyboard-input-in-php>
8. <https://www.php.net/manual/en/function.key.ph>

11 Allegati

Elenco degli allegati:

- QdC
- Abstract
- Risultati Test
- Applicativo
- Diari
- ActivityDiagram generale
- ActivityDiagram multiplayer
- Audio
- Design
- Gantt Preventivo
- Gantt Consuntivo
- Schema ER
- Schema delle architetture di sistema
- Frasi per database
- UML
- Use Case