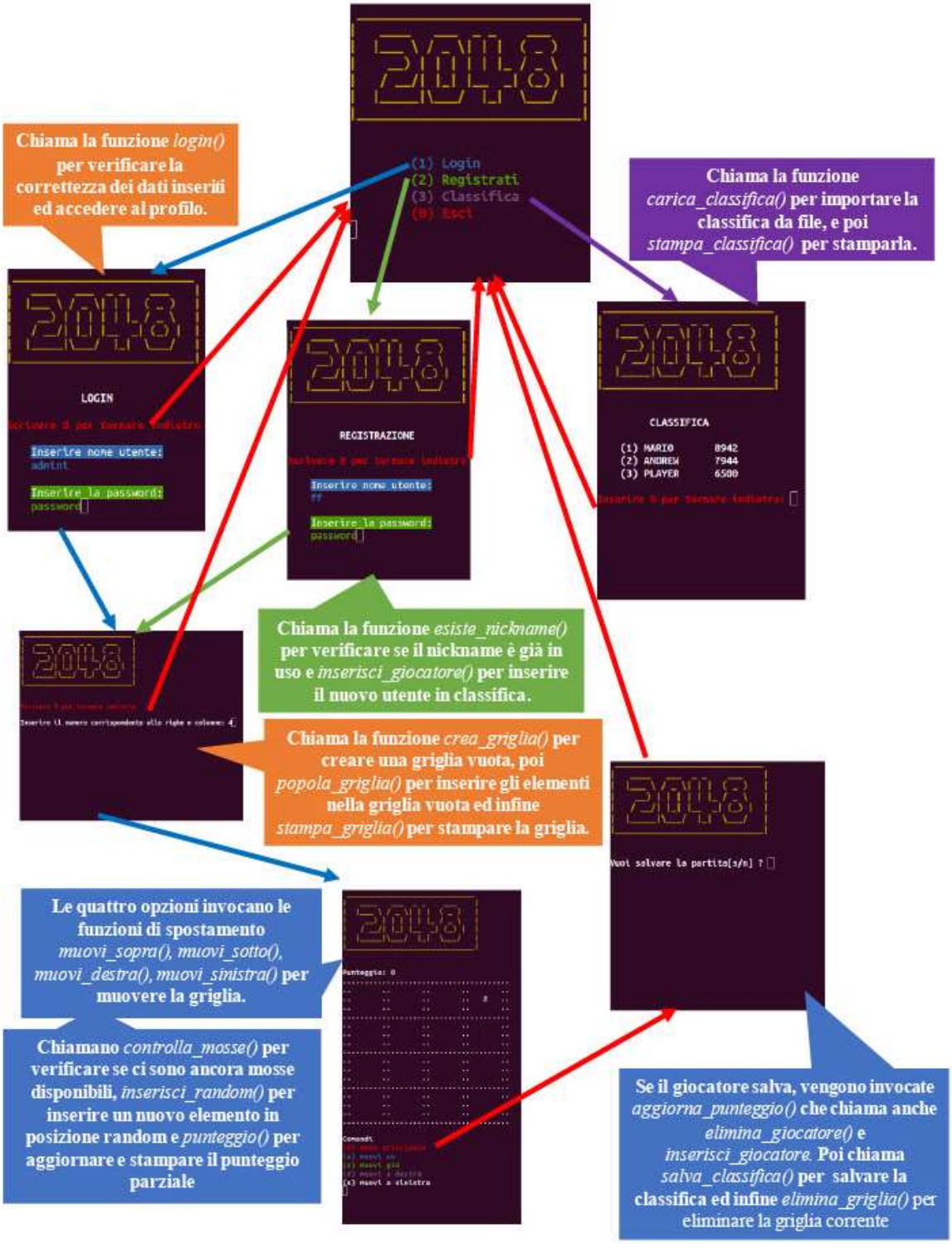
# Documento di progetto

## Parte non tecnica

- **Nome e Cognome:** Giovanni Lombardo
- **N. matricola:** 516130
- **Titolo:** 2048
- **Descrizione:** A 2048 si gioca su una semplice griglia in cui scorrono caselle con numeri diversi (tutti i numeri sono potenze di 2), senza intralci quando un giocatore le muove. Il gioco usa i tasti w,a,s,d per spostare tutte le caselle a sinistra o a destra oppure in alto o in basso. Se due caselle contenenti lo stesso numero si scontrano mentre si muovono, si fondono in un'unica casella che avrà come numero la somma delle due tessere che si sono scontrate. Ad ogni turno, una nuova tessera con il valore di 2 o 4 apparirà in modo casuale in un punto vuoto sulla griglia. Inoltre, un tabellone tiene traccia del punteggio dell'utente. Il punteggio dell'utente inizia da zero e viene incrementato ogni volta che due tessere si combinano, con il valore della nuova casella. Questa versione prevede la possibilità di scegliere la dimensione della griglia e di salvare i migliori punteggi in un file locale. Ogni giocatore per poter iniziare una partita dovrà registrarsi, o eventualmente effettuare il login. Il punteggio del giocatore viene sovrascritto solamente se il nuovo punteggio è migliore del precedente.

# Parte tecnica

## Interfaccia utente



# File griglia.h

```c
#ifndef GRIGLIA_H
#define GRIGLIA_H

#include <stdio.h>
#include "linea.h"
#include "classifica.h"


//griglia
struct nodo_griglia{
        linea linea;
        struct nodo_griglia* next;
        struct nodo_griglia* prec;
};

struct griglia{
        int parz;
        giocatore player;
        struct nodo_griglia* head;
        struct nodo_griglia* tail;
};

typedef struct griglia* griglia;




griglia crea_griglia();

int popola_griglia(griglia my_griglia,int n,giocatore player);

void stampa_griglia(griglia my_griglia,int n,int punteggio);

int griglia_piena(griglia my_griglia, int n);

void inserisci_random_griglia(griglia my_griglia,int n);

void muovi_destra_griglia(griglia my_griglia, int n, classifica new_classifica);

void muovi_sinistra_griglia(griglia my_griglia,int n,  classifica new_classifica);

void muovi_sopra(griglia my_griglia,int n, classifica new_classifica);

void muovi_sotto(griglia my_griglia,int n,  classifica new_classifica);

int controlla_colonne(griglia my_griglia);

int controlla_mosse(griglia my_griglia,int n);

void elimina_griglia(griglia my_griglia);

void game_over(griglia my_griglia,giocatore player, int punteggio, classifica new_classifica);

void salva(giocatore player,int punteggio, classifica new_classifica,griglia my_griglia);


#endif
```

# File linea.h

```c
#ifndef LINEA_H
#define LINEA_H



//linee
struct nodo_linea{
        int data;
        struct nodo_linea* next;
        struct nodo_linea* prec;
};

struct linea{
        struct nodo_linea* head;
        struct nodo_linea* tail;
};

typedef struct linea* linea;



linea crea_linea();

int inserisci_cella(linea my_linea,int elem);

void inserisci_random(linea my_linea,int n);

int confronta_celle(struct nodo_linea prec, struct nodo_linea next);

int controlla_linea(linea my_linea);

int somma_celle(struct nodo_linea prec,struct nodo_linea next,int* parz);

int linea_piena(linea my_linea);

void muovi_destra(linea my_linea,int* parz, int n);

void muovi_sinistra(linea my_linea,int* parz, int n);

void elimina_linea(linea my_linea);

#endif
```

# File classifica.h

```c
#ifndef CLASSIFICA_H
#define CLASSIFICA_H
#include <stdio.h>

//classifica
struct giocatore{
        char nickname[10];
        char password[10];
        int punteggio;
        struct giocatore* next;

};


typedef struct giocatore* giocatore;

struct classifica{
        struct giocatore* head;
};

typedef struct classifica* classifica;


classifica carica_classifica();

void stampa_classifica(classifica my_classifica);

int esiste_nickname(classifica my_classifica,char* nickname);

giocatore inserisci_giocatore(classifica my_classifica,char* nickname,char* password,int punteggio);

giocatore login(classifica my_classifica, char* nickname,char* password);

int elimina_giocatore(classifica my_classifica,char* nickname);

void cancella_classifica(classifica my_classifica);

int aggiorna_punteggio(classifica my_classifica, giocatore player,int punteggio);

void salva_classifica(classifica my_classifica);


#endif
```

# Tipi di dato astratto

- **Griglia:** lista doppiamente concatenata di liste. Formata da n (n inserito dall'utente) *nodi_griglia,* il puntatore al giocatore che è entratoe il punteggio parziale associato.
- **Nodo_griglia:** contiene un puntatore ad una *linea,* un puntatore al nodo successivo ed uno al precedente.
- **Linea:** lista doppiamente concatenata. Ogni linea rappresenta una riga ed è formata da n *nodi_linea.*
  **Nodo_linea:** contiene il valore della cella, un puntatore next ed un puntatore prec.
- **Classifica:** lista ordinata contenente *giocatori.*
- **Giocatore:** nodo della lista *Classifica* contente il nickname, il punteggio e un puntatore al successivo.

# Operazioni

### Griglia

- **Crea_griglia():** crea una griglia vuota e restituisce il puntatore.
- **Popola griglia(griglia my_griglia, int n, giocatore player):** chiama *crea_linea() per creare n linee*, inserisce 0 in ogni nodo delle linee tramite la funzione *inserisci_cella,* un 2 in un elemento random e associa un giocatore alla griglia; infine inserisce le linee una ad una nella griglia. Restituisce 0 o 1.
- **Stampa_griglia(griglia my_griglia, int n, int punteggio):** stampa il punteggio del giocatore e la griglia, stampando in ogni cella il valore contenuto in *nodo_linea* e lasciando uno spazio vuoto in caso di 0.
- **Muovi_destra_griglia(griglia my griglia, int n, classifica new_classifica):** richiama la funzione *muovi_destra()* n volte, in modo da agire su ogni linea.
- **Muovi_sinistra_griglia(griglia my_griglia, int n, classifica new_classifica):** richiama la funzione *muovi_sinistra()* n volte, in modo da agire su ogni linea.
- **Muovi_sopra(griglia my griglia, int n, classifica new_classifica):** confronta a coppie i nodi di una lista con i nodi nella stessa posizione della lista successiva tramite *confronta_celle()*, partendo dall'inizio. Se trova elementi uguali li somma tramite *somma_celle()* ed inserisce il nuovo valore nella cella più a sinistra.
- **Muovi_sotto(griglia my griglia, int n, classifica new_classifica):** confronta a coppie i nodi di una lista con i nodi nella stessa posizione della lista successiva tramite *confronta_celle()*, partendo dalla fine. Se trova elementi uguali li somma tramite *somma_celle()* ed inserisce il nuovo valore nella cella più a destra.
- **Inserisci_random_griglia(griglia my_griglia, int n):** estrae una linea random, verifica se non è piena tramite *linea_piena()* e inserisce un numero in una cella random tramite *inserisci_random().*
- **Controlla_colonne(griglia my_griglia):** verifica attraverso *confronta_celle()* se ci sono celle uguali a coppie in verticale che permettono all'utente di effettuare ancora mosse. Viene richiamato da *controlla_mosse().* Restituisce 0 o 1.

- **Controlla_mosse(griglia my_griglia, int n):** richiama *controlla_linea()* per ogni linea e *controlla_colonna()* per ogni linea in verticale. Restituisce 0 o 1.
- **Griglia_piena(griglia my_griglia, int n):** Verifica attraverso *linea_piena()* se la griglia è piena. Restituisce 0 o 1.
- **Salva(giocatore player, int punteggio, classifica new_classifica, griglia my_griglia):** effettua il salvataggio della partita tramite le funzioni *aggiorna_punteggio()* e *salva_classifica().*
- **Game_over(griglia my_griglia, giocatore player, int punteggio, classifica new_classifica):** mostra il messaggio di fine delle mosse disponibili e poi chiama *salva().*
- **Elimina_griglia(griglia my_griglia):** chiama *elimina_linea()* n volte per eliminare tutte le linee e poi distrugge la griglia.

## Linea

- **Crea_linea():** crea una nuova linea vuota e ne restituisce il puntatore.
- **Inserisci_cella(linea my_linea, int elem):** inserisce elem nelle celle di my_linea. Restituisce 0 o 1.
- **Inserisci_random(linea my_linea, int n):** inserisce un elemento tra 2 e 4 in una cella random vuota. Viene eseguita dopo ogni mossa e dopo che *controlla_mosse* ritorna un esito positivo.
- **Confronta_celle(struct nodo_linea prec, struct nodo_linea next):** confronta il valore del nodo prec col valore del nodo next. Restituisce 1 o 0.
- **Controlla_linea(linea my_linea):** verifica attraverso *confronta_celle()* se ci sono celle limitrofe uguali che permettono all'utente di effettuare ancora mosse. Viene richiamato da *controlla_mosse().* Restituisce 0 o 1.
- **Somma_celle(struct nodo_linea prec, struct nodo_linea next, int* parz):** somma il contenuto delle celle prec e next. Restituisce la somma.
- **Muovi_destra(linea my_linea, int* parz, int n):** confronta a coppie i nodi delle n linee tramite *confronta_celle()* partendo dalla fine. Se trova elementi uguali li somma tramite *somma_celle()* ed inserisce il nuovo valore nella cella più a destra.
- **Muovi_sinistra(linea my_linea, int* parz, int n): c**onfronta a coppie i nodi delle n linee tramite *confronta_celle()* partendo dall'inizio. Se trova elementi uguali li somma tramite *somma_celle()* ed inserisce il nuovo valore nella cella più a sinistra.
- **Linea_piena(linea my_linea):** verifica se la linea è piena. Restituisce 0 o 1.
- **Elimina_linea(linea my_linea):** distrugge la linea.

## Classifica

- **Carica_classifica(FILE):** carica da file la classifica. Restituisce la classifica
- **Stampa_classifica(classifica my_classifica):** stampa la classifica a video.
- **Esiste_nickname(classifica my_classifica, char* nickname):** verifica se esiste già un utente con quel nickname. Restituisce 0 o 1.
- **Inserisci_giocatore(classifica my_classifica, char* nickname, char* password, int punteggio):** inserisce ordinatamente un nuovo giocatore con questi prametri. Se l'utente si è appena registrato, l'inserimento avverrà con il punteggio 0. Restituisce il puntatore al giocatore.
- **Login(classifica my_classifica, char* nickname, char* password):** Verifica che esiste un giocatore con questi parametri e ne restituisce il puntatore.

- **Elimina_giocatore(classifica my_classifica, char* nickname):** elimina il giocatore con quel determinato nickname. Restituisce 0 o 1.
- **Aggiorna_punteggio(classifica my_classifica, giocatore player, int punteggio):** Aggiorna il punteggio dell'utente. Visto che dopo la modifica la classifica deve rimanere ordinata, la funzione prima cancella l'utente tramite *elimina_giocatore()* e poi lo reinserisce con il nuovo punteggio tramite *inserisci giocatore*. Restituisce 0 o 1.
- **Salva classifica(classifica my_classifica):** salva la classifica su file.
- **Cancella_classifica(classifica my_classifica):** libera lo spazio allocato per la classifica.