**Klotski Project**

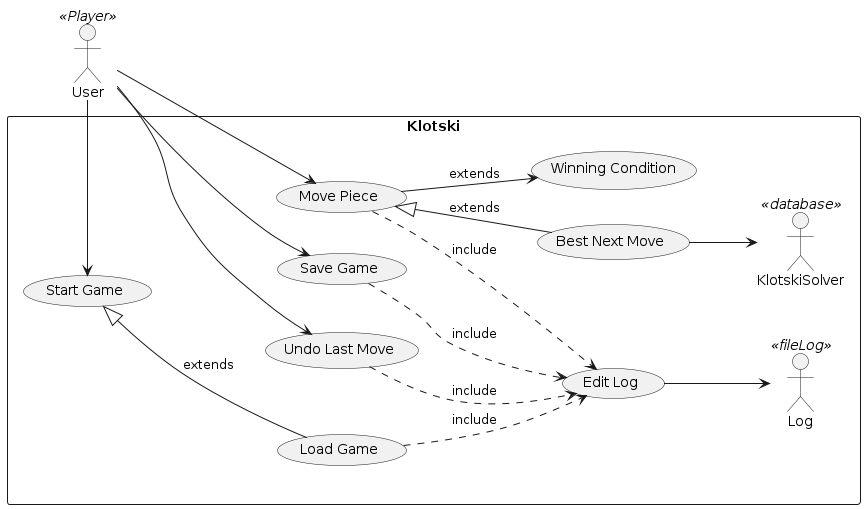
**Documento di Specifiche**

**1) Use Cases Diagrams**

**Nomenclatura**:

* User: l’utente che interagisce con il gioco
* Klotski: Nome del sistema
* KlotskiSolver: il database che, ricevuta una configurazione, cerca una similitudine nel proprio archivio in cerca della mossa migliore per vincere
* Board: tavolo del gioco dove vengono piazzati i blocchi (Pieces)
* Log: documento in cui vengono salvate i movimenti dei Pieces sulla board
* Winning Condition: Il gioco si può considerare concluso solo quando il blocco Rosso (Red Piece) raggiunge la posizione d’uscita

**MADE BY Jonatan Ademi**



**2) Use Cases Diagrams**

Le tabelle seguono il seguente Template:

1. UC name: nome della use case
2. UC Overview: Descrizione della use case
3. Actors: gli attori principali coinvolti nella use case
4. Flow of Events: Presenta la linea di eventi generali che potrebbe seguire la use case
5. Basic Flow: Descrive il comportamento ideale seguito dal sistema
6. Alternative Flow: Descrive invece altri possibili comportamenti, solitamente quando un valore non è valido o disponibile **MADE BY Jonatan Ademi**
7. Data: I dati o le informazioni che vengono utilizzati per l’esecuzione della use case
8. Precondition: Lo stato in cui il sistema si deve trovare prima che parta la use case
9. Post conditions: La lista dei possibili stati in cui si può trovare il sistema
10. Input: Le azioni che provocano l’inizio della use case, solitamente quando lo User preme il pulsante equivalente
11. Output: Ciò che viene visualizzato al termine della use case, solitamente nella Board

Alcune caselle rimangono vuote poiché la use case non prevede quel tipo di informazione

|  |  |
| --- | --- |
| **Use-case name** | **UC1 – Start New Game** |
| **Use case Overview** | The User starts to play on one pre-determined configuration |
| **Actors** | User, Log |
| **Flow of Events** | The User may select one from 4 pre-determined configurations to play on, after which the game generates a Log file in which it would write down future commands. |
| **Basic Flow** | The User determines the configuration on which to play, the Board gets set and the Log is generated |
| **Alternative Flow** |  |
| **Data** | Config presets |
| **Precondition** | The game is displayed |
| **Post Conditions** | The Board is initialized |
| **Input** | The User starts the game, The User press a “Configuration” button |
| **Output** | The Board displays the chosen configuration |
| **Extension Point** | Load Game (described below) |

|  |  |
| --- | --- |
| **Use-case name** | **UC2 – Load Old Game** |
| **Use case Overview** | The User continues an old game he had previously started and saved |
| **Actors** | User, Log |
| **Flow of Events** | The User who wants to continue an old game inserts the Log’s name into the searchbar and the system will search if there is any Log with that name |
| **Basic Flow** | The User inserts the Log, after which the selected Log will be loaded and its corresponding Board will be viewed. |
| **Alternative Flow** | There isn’t a log with that name, the User is notifed |
| **Data** | Log’s Data |
| **Precondition** | At least one previous log |
| **Post Conditions** | The Board is initialized |
| **Input** | The User press the “load Game” button, after which selects a Log |
| **Output** | The generation of the updated Board’s view |

**MADE BY Jonatan Ademi**

|  |  |
| --- | --- |
| **Use-case name** | **UC3 – Make A Movement** |
| **Use case Overview** | The User moves a Piece in a direction |
| **Actors** | User, Log |
| **Flow of Events** | The User may select a Piece from the Board, after which they input the wished direction. If the move is valid, it will be recorded in the Log |
| **Basic Flow** | The selected Piece is moved onto the new position and the movement is saved in the Log |
| **Alternative Flow** | The movement isn’t valid (either position occupied or the Piece is too large to fit in), therefore the Piece will not be moved |
| **Data** | None |
| **Precondition** | A selected Piece |
| **Post Conditions** | The Board gets updated |
| **Input** | The User selects a piece, then they press a “Direction” button |
| **Output** | If the entered movement is permitted, the Board’s view will be updated and the movement will be added to the Log |

|  |  |
| --- | --- |
| **Use-case name** | **UC4 – Best Next Move** |
| **Use case Overview** | The User requests help from the Solver’s Database, which woud give the solution, if available |
| **Actors** | User, Klotski Solver |
| **Flow of Events** | The User may request help from the Game, which may presents the best next move from the current board configuration. |
| **Basic Flow** | The Solver, upon User’s request, will move a Piece into the next position following the “Best Next Move” contained in its database |
| **Alternative Flow** | The database doesn’t find a comparison for the Board’s configuration, therefore a notification will be sent to the User |
| **Data** | Solver’s Database |
| **Precondition** | The Database has some configurations to compare with |
| **Post Conditions** | The updated Board has subsequent equivalences with the Database |
| **Input** | The User press the “BNM” button |
| **Output** | After confronting the current Board’s configuration with those inside its database, if there is a possible “best next move”, the Board’s view will be updated and the movement will be added to the Log |

**MADE BY Jonatan Ademi**

|  |  |
| --- | --- |
| **Use-case name** | **UC5 – Undo Last Move** |
| **Use case Overview** | The User who wants to undo his last movement, may press the corresponding button |
| **Actors** | User, Log |
| **Flow of Events** | The User who wants to undo his last movement, may press the corresponding button |
| **Basic Flow** | Last movement made by the User is canceled as well as the corresponding line in the Log. The Board will show the configuration previous to that move |
| **Alternative Flow** | There is no movements to undo |
| **Data** | Log’s Data |
| **Precondition** | At least one movement registered in the log |
| **Post Conditions** | The Board returns to its previous state |
| **Input** | The User press the “Undo Game” button |
| **Output** | The Board returns to one step back and the undone movement would be deleted from the Log |

|  |  |
| --- | --- |
| **Use-case name** | **UC6 – Winning Condition** |
| **Use case Overview** | When the “Red Piece” has arrived to the goal, the game is won |
| **Actors** | Klotski |
| **Flow of Events** | After every movement (except Undo) the game will check if the winning requirement is met |
| **Basic Flow** | The Red Piece is on the exit, the User has Won |
| **Alternative Flow** | The Red Piece didn’t reach the exit yet, keep playing! |
| **Data** | “Red Piece” ’s position |
| **Precondition** | The “Red Piece” must be at the exit |
| **Post Conditions** | The Game has ended |
| **Input** | A piece has been moved |
| **Output** | Returns a winning screen if the requirement is met |

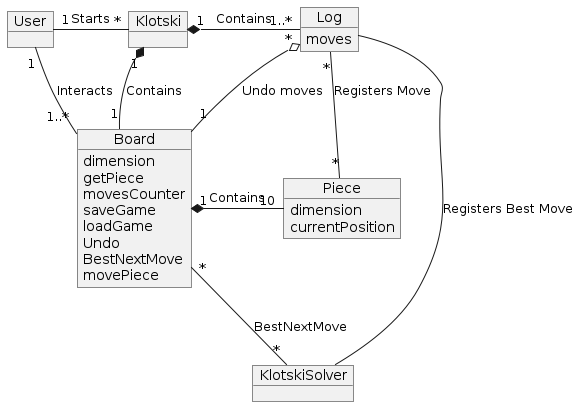
**MADE BY Jonatan Ademi**

|  |  |
| --- | --- |
| **Use-case name** | **UC7 – Save Game** |
| **Use case Overview** | The User saves the current game configuration |
| **Actors** | User, Log |
| **Flow of Events** | The User saves the game’s log under a new name, so it can be recovered |
| **Basic Flow** | The User inserts the Log, after which the selected Log will be loaded and its corresponding Board will be viewed. |
| **Alternative Flow** | None |
| **Data** | Log’s Data |
| **Precondition** | There must be a configuration on the Board |
| **Post Conditions** | A new Log has been generated |
| **Input** | The User press the “Save Game” button, after they name the Log |
| **Output** | The Log is saved in a specific folder under the new name |

**Documento di Design**

**1) Domain Model**

**MADE BY Jonatan Ademi**

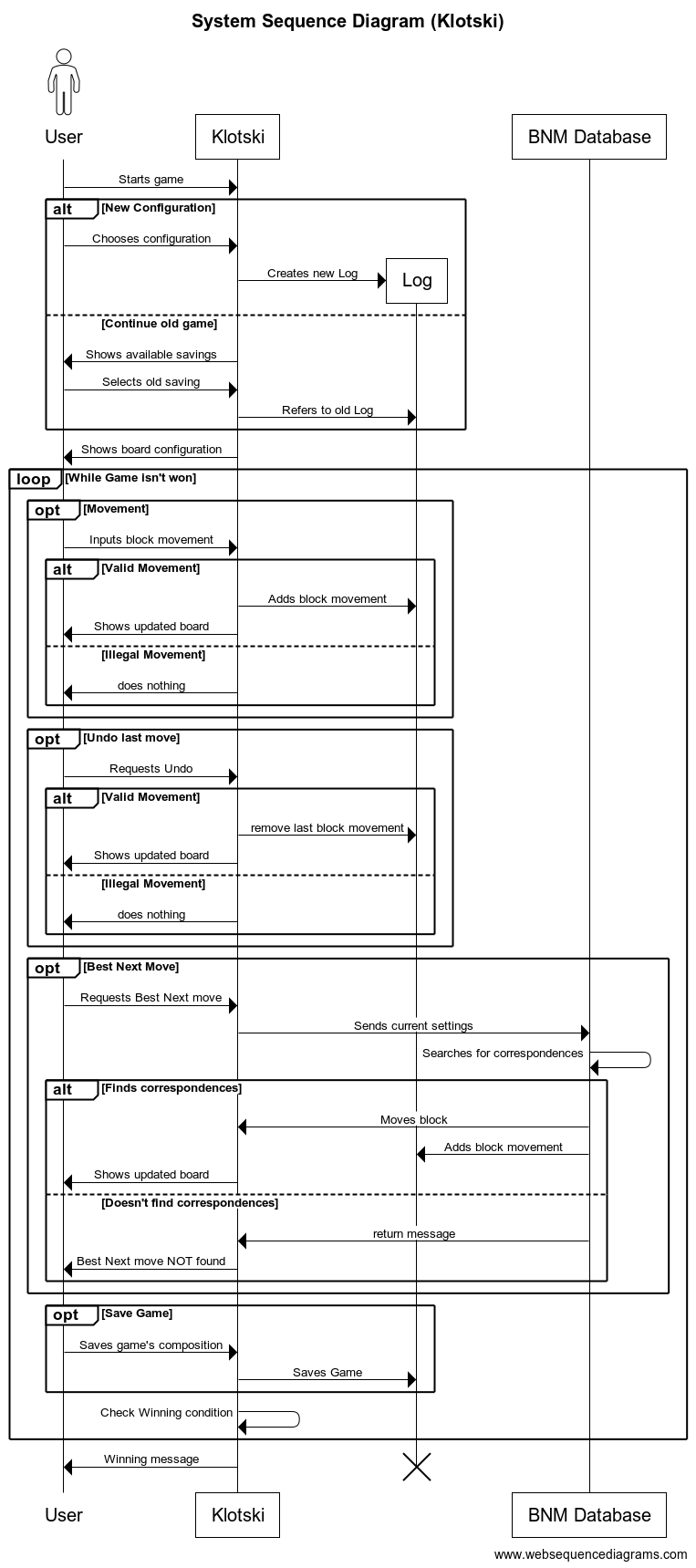


Descrizioni:

* Uno User può iniziare una o più partite a Klotski
* Quando viene iniziata una partita di Klotski, vengono generate 1 board e 1 Log (Sempre nella stessa partita si possono generare più log, ma la board rimane la stessa, cambia solo la configurazione dei Pieces)
* Quando la board viene generata, vengono creati 10 pezzi di varie dimensioni, tra cui il Red Piece (necessario per vincere)
* Uno User può interagire con una o più Board e tramite di essa può:
  + Muovere i Pieces, la modifica del log avviene dalla classe Piece
  + Richiedere la migliore mossa successiva (Best Next Move or BNM), la modifica del log avviene dalla classe KlotskiSolver
  + Annullare l’ultima mossa eseguita, la modifica del log avviene dalla classe Board

**2) System Sequence Diagram (SSD)**

**MADE BY Jonatan Ademi**

**START**

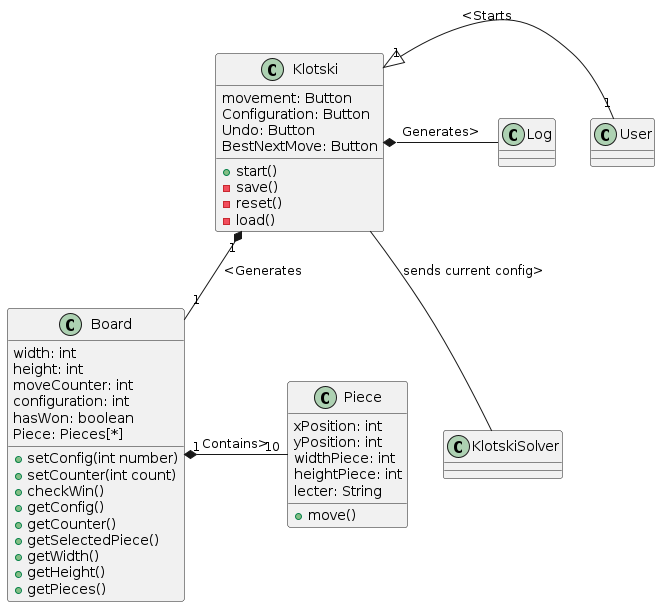
Nel momento in cui lo User avvia il gioco può scegliere se cominciare una nuova configurazione oppure continuare una partita precedente.

Successivamente, finché la condizione di Vittora non viene raggiunta, lo User può muovere i blocchi, annullare l’ultimo spostamento oppure farsi aiutare dal Klotski Solver.

Eventualmente può salvare la partita per continuarla in un altro momento.

Omesso il fatto che in qualunque momento il giocatore può decidere di ricominciare da capo, che sia tramite funzione di reset() oppure selezionando un’altra configurazione (tra i preset o le partite precedenti).

**3) Class Model MADE BY Jonatan Ademi**



1. La classe Klotski è la rappresentazione del *main*, ed è composta da vari pulsanti:

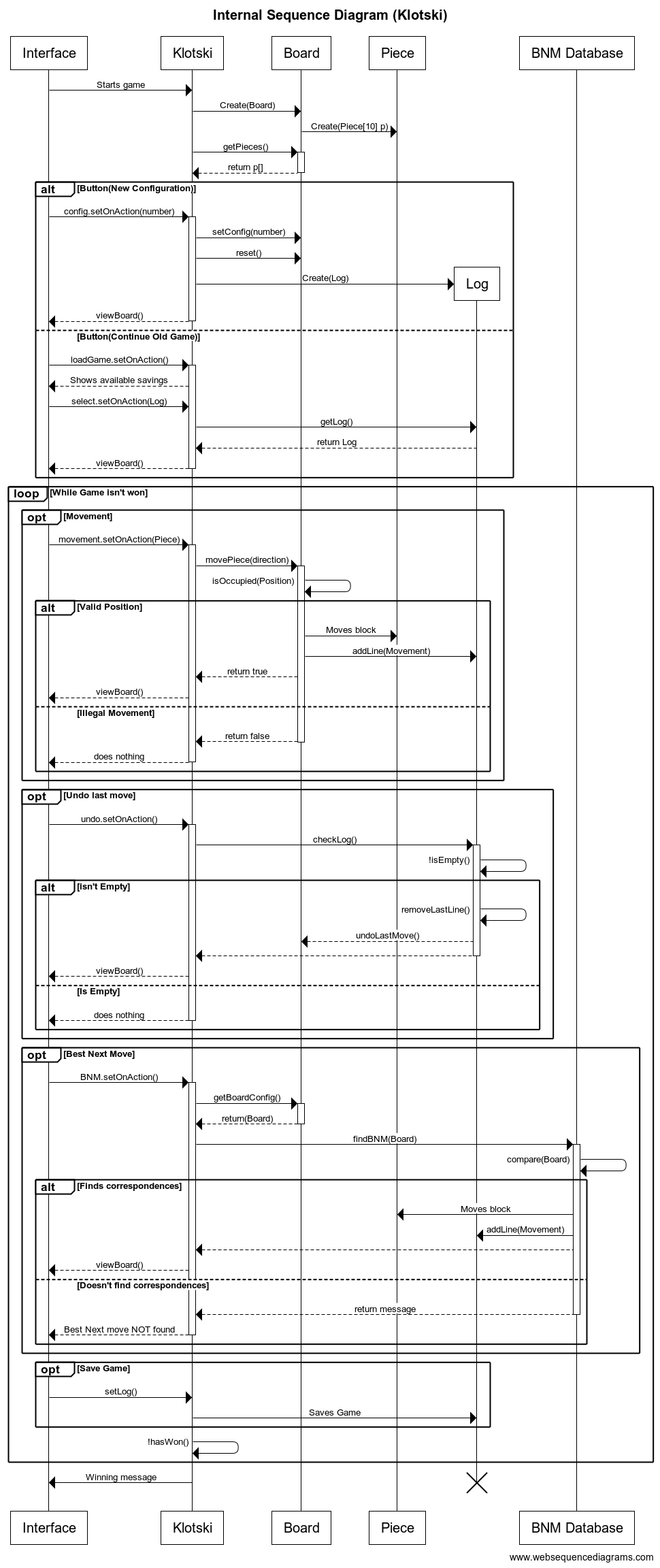
* movement: sono 4 pulsanti direzionali che, selezionato il Piece da muovere, tentano di spostarlo nella posizione decisa dal giocatore (o Klotski Solver)
* Configuration: ci sono 4 configurazioni di default con cui cominciare la partita, ognuna con una disposizione diversa dei vari Piece
* Undo: annulla l’ultimo spostamento effettuato, cancellando la corrispettiva stringa dal Log
* BestNextMove: invia l’attuale configurazione della Board al Klotski Solver e, se all’interno del suo database si trova la medesima configurazione, Klotski Solver esegue la mossa successiva, con corrispondente aggiornamento del Log

1. Inoltre, il main dispone dei pulsanti

* save(): per generare il Log nella directory, cosicché si possa continuare il gioco in un secondo momento
* load(): carica un Log dalla suddetta directory per continuare la partita
* reset(): reimposta alla configurazione di default di partenza (ATTENZIONE: non è possibile eseguire l’undo di un reset())

1. Le classi BoardView(), InsView(), PieceView(), Texts(), ViewPrint() e ViewSettings() non sono state inserite nel diagramma per evitare di renderlo troppo dispersivo e perché gestiscono esclusivamente l’aspetto grafico del gioco.

**4) Internal Sequence Diagram MADE BY Jonatan Ademi**

**START**

In questo caso, l’interfaccia rappresenta ciò che l’utente può vedere e che può usare per interagire col sistema.

Quando viene avviata il gioco, vengono automaticamente generati la Board e i rispettivi Pieces.

La schermata mette a disposizione 4 configurazioni predeterminate. Cliccando una di queste, viene assegnato il corrispettivo numero di configurazione alla classe Board, dopodiché quest’ultima viene resettata (La funzione di reset consulta il numero di configurazione assegnato e posiziona i blocchi secondo tale config.) e viene generato un nuovo log.

Alternativamente, c’è la possibilità di riprendere una partita precedente caricando un log già salvato, questo viene “assegnato” alla partita corrente e la board eredita le posizioni dei Pieces.

Come visto precedentemente, ci sono tre azioni principali che il giocatore può svolgere: Movimento, Annullamento e Suggerimento.

Move(): Una volta selezionato il blocco da spostare e determinata la direzione, l’informazione viene passata alla classe Board() il quale controlla se lo spostamento è possibile. In caso affermativo, il blocco viene spostato, lo spostamento scritto nel log e la Board aggiornata. Altrimenti non succede nulla.

Undo(): Nel caso il giocatore volesse annullare l’ultima mossa effettuata, si controlla il log, si identifica tale mossa, la si rimuove dal log e il blocco spostato corrispondente ritorna alla posizione precedente.

BestNextMove(): Una volta premuto il pulsante, si confronta il Database con le soluzioni e (se presente) viene eseguita la mossa migliore per la vittoria, riportando lo spostamento nel log e aggiornando la Board.

Dopo ogni spostamento, si controlla se le condizioni di vittoria sono state soddisfatte e in caso affermativo viene stampato il messaggio di vittoria e il log della partita viene chiuso (è stato gestito in tale modo poiché abbiamo concordato che avrebbe poco senso conservare il log di una partita conclusa). **MADE BY Jonatan Ademi**