Progetto Ingegneria del Software

Anno Accademico 2019-2020
Sessione estiva

Make it True

Daniele Mariella
261064

Docente: Prof. Saverio Delpriori

Indice

1. Specifica del Software

Il gioco consiste in una serie di quesiti inerenti due argomenti: ***matematica*** e ***logica***.

Scopo del gioco è raggiungere il numero più alto di risposte esatte ad ogni livello.
Entrambi gli argomenti sono suddivisi su *3 livelli*:

1) *Facile*
2) *Medio*
3) *Difficile*

Mentre per i livelli 1 e 2 le domande saranno 7, per il livello 3 ce ne saranno 5.

Il giocatore ha a sua disposizione un **suggerimento** ad ogni domanda che indirizzerà il ragionamento sulla strada giusta, quest'ultimo non sarà disponibile nel livello difficile.

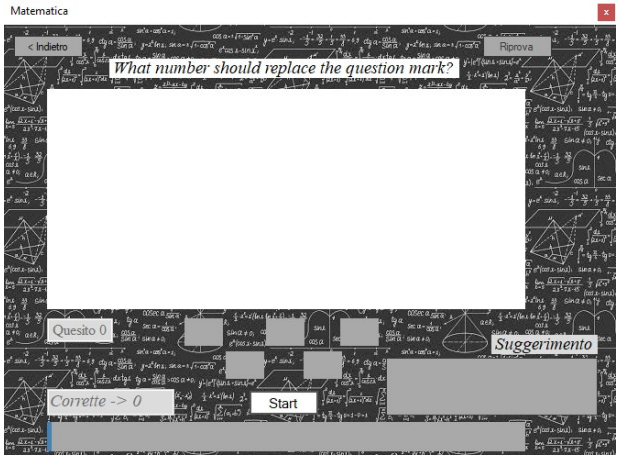
In caso di risposta sbagliata ad una domanda, verrà comunque data la possibilità di ragionare sulle risposte rimanenti fino a quando non si otterrà quella corretta. Ovviamente in questo caso il quesito verrà considerato come errato e non se ne terrà conto nel risultato finale.

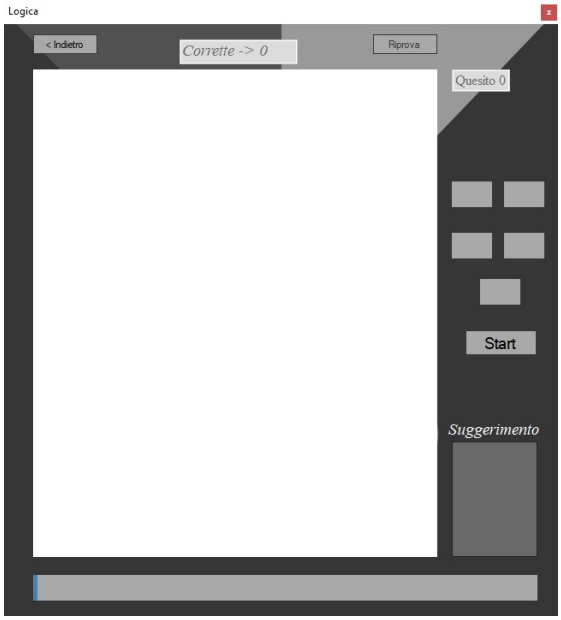
Nei livelli di logica nei quali la scelta di risposta deve esser effettuata tra A, B, C, D, E, può in modo casuale accadere che l'ordine delle lettere non sia in ordine, questo per inserire un'ulteriore difficoltà.

Un **timer**, scandirà l'inizio e la fine del livello portando il giocatore a pensare più velocemente e quindi adattarsi ad un certo limite di tempo.
Nel caso in cui il livello affrontato sarà quello difficile, il tempo ad ogni risposta errata (anche all'interno della stessa domanda) subirà *un aumento automatico* di alcuni secondi per rendere più complicato il gioco.
Se il tempo scadrà prima di aver dato una risposta corretta a tutti i quesiti, la prova verrà considerata non completa.

Si avrà poi immediatamente la possibilità di *giocare di nuovo* la stessa partita e di portarla al termine nel tempo limite oppure tornare al menù principale per scegliere un altro livello/argomento.

2. Studio del problema





I punti critici.

- ✗ Come dividere l'argomento matematica dalla logica avendo quesiti con risposte completamente diverse. Mentre per il primo le risposte sono basate su dei numeri, il secondo ha solo le lettere A, B, C, D, E, come possibili soluzioni.
- ✗ Aumentare la difficoltà di gioco di ogni livello.
- ✗ Gestire i problemi di casualità delle risposte dei quesiti evitando che si ripetano, eliminando il caso che venga posta la risposta esatta sempre nella stessa posizione ad ogni nuova domanda.
- ✗ Gestione Timer interrompendo una partita tornando indietro
- ✗ Argomento logica, gestione delle risposte A, B, C, D, E nel caso in cui le risposte estratte non siano esattamente nella lettera corrispondente.
- ✗ Nel caso in cui il giocatore clicca più volte il tasto Riprova, il gioco si blocca.

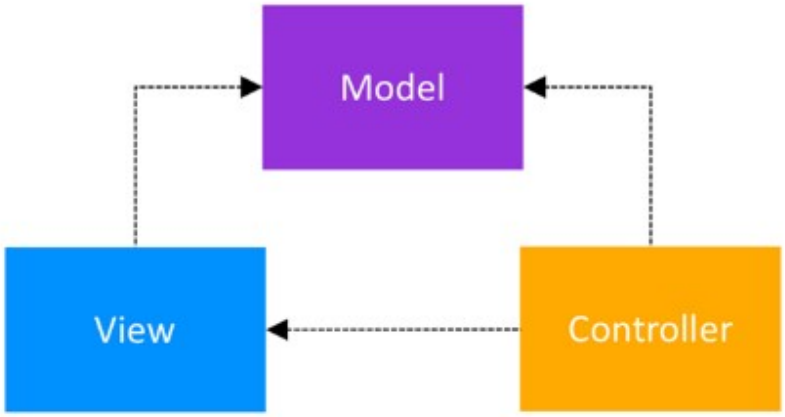
Come si è scelto di affrontarli.

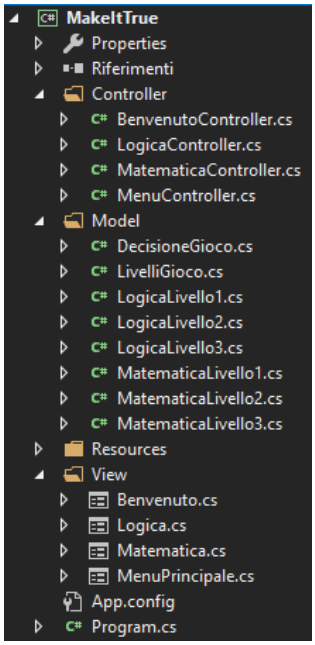
- ✔ Data la diversità delle risposte, si è deciso di creare due Form diversi personalizzati per gli argomenti di matematica e logica evitando confusioni nelle domande e difficoltà nell'impostazione.
- ✔ La difficoltà viene aumentata creando innanzitutto differenti quesiti tra i livelli 1 e 2 mentre per il livello 3 si è deciso di bloccare l'utilizzo del suggerimento e aggiungere una penale in termini di secondi ad ogni risposta errata.
- ✔ Per correggere questo errore che è alla base del gioco, dato che l'utente avrebbe ad ogni domanda già previsto l'esatta posizione della risposta corretta successiva, si è inserito un Thread.sleep() che sospende il thread per un determinato numero di millisecondi.
- ✔ Bloccare il Timer è stato gestito controllando l'eccezione della chiusura del Form, inserendo uno Stop() al relativo Timer.
- ✔ Si è scelto dunque di imporre un ordine iniziale ordinato ad ogni nuovo quesito → A, B, C, D, E. Nel caso in cui la risposta corretta non si trovasse nella posizione ordinata, effettuare uno scambio di lettere creando un ordine casuale ma mantenendo sia l'unicità delle lettere per ogni bottone sia la correttezza della risposta, esempio → C, E, A, D, B (corretta E).
- ✔ La situazione è stata corretta disattivando il bottone Riprova dopo la prima volta che il giocatore decide di rigiocare lo stesso livello e riattivandolo con il nuovo primo quesito.

3. Scelte architetturali

Design Pattern

Con l'idea di sviluppare un gioco che utilizzi un'interfaccia grafica si è scelto di utilizzare il pattern di sviluppo MVC (Model-View-Controller). Questo schema architetturale consente di realizzare la separazione delle competenze ridimensionando il gioco in termini di complessità, semplificando il debug, consentendo una riusabilità del codice e per un'eventuale aggiunta di codice.





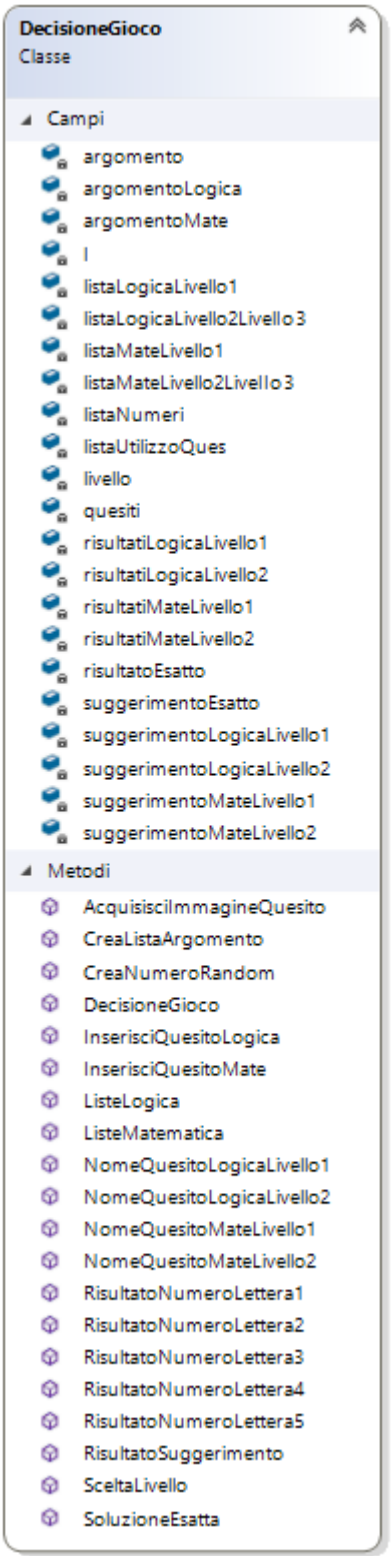
La visualizzazione e il controller dipendono dal modello. Il modello non dipende né dalla visualizzazione né dal controller.

Modelli utilizzati:

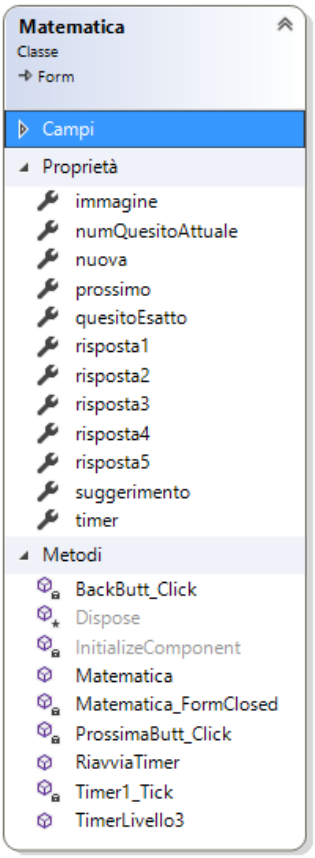
i. Modello: rappresenta lo stato dell'applicazione e le operazioni che deve seguire.
ii. Vista: presenta il contenuto tramite interfaccia utente.
iii. Controller: gestisce e risponde all'input e all'interazione dell'utente.

Alcune classi che compongono il nucleo del programma.

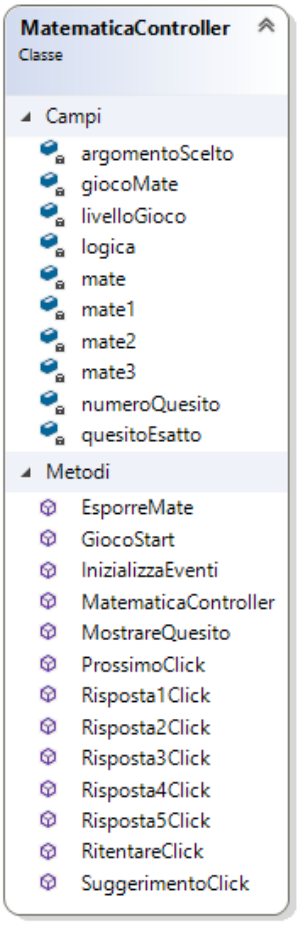
Model

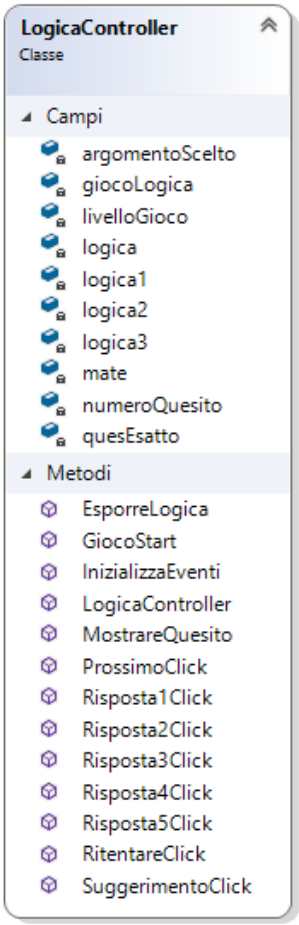


View



Controller



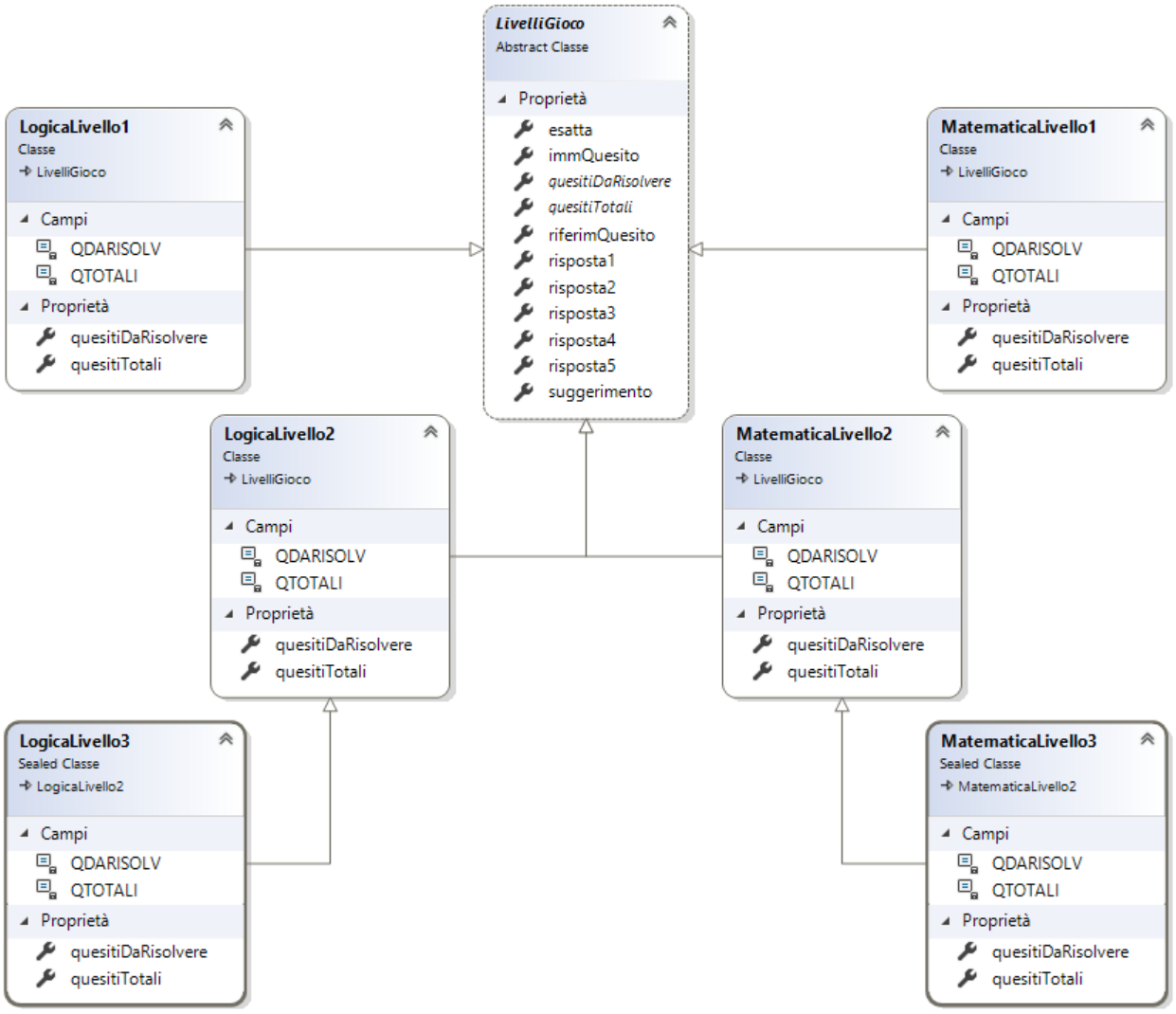


DecisioneGioco

Classe che richiama i metodi e le classi necessarie a creare un'istanza del gioco vero e proprio.

- **AcquisisciImmagineQuesito**:
  metodo per trovare il quesito (immagine).
- **CreaListaArgomento**:
  metodo per la creazione della lista a seconda dell'argomento (mate o logica) e del relativo livello.
- **CreaNumeroRandom**:
  metodo per la creazione di un numero casuale senza commettere l'errore di ripeterlo 2 o più volte.
- **DecisioneGioco**:
  metodo costruttore della classe DecisioneGioco.
- **InserisciQuesitoLogica**:
  metodo per l'inserimento del quesito di logica nella lista opportuna con i relativi dati (risposte, suggerimento, nome immagine e riferimento quesito).
- **InserisciQuesitoMate**:
  metodo per l'inserimento del quesito di matematica nella lista opportuna con i relativi dati (risposte, suggerimento, nome immagine e riferimento quesito).
- **ListeLogica**:
  metodo che contiene tutte le liste inerenti l'argomento logica, nello specifico:
  → i possibili risultati del livello 1, 2 e 3;
  → i possibili suggerimenti del livello 1 e 2.
- **ListeMatematica**:
  metodo che contiene tutte le liste inerenti l'argomento matematica, nello specifico:
  → i possibili risultati del livello 1, 2 e 3;
  → i possibili suggerimenti del livello 1 e 2.
- **NomeQuesitoLogicaLivello1 e NomeQuesitoLogicaLivello2**:
  metodi che collegano il numero random con il nome dell'immagine che conterrà il quesito di logica.
- **NomeQuesitoMateLivello1 e NomeQuesitoMateLivello2**:
  metodi che collegano il numero random con il nome dell'immagine che conterrà il quesito di matematica.
- **RisultatoNumeroLettera1, RisultatoNumeroLettera2, RisultatoNumeroLettera3, RisultatoNumeroLettera4, RisultatoNumeroLettera5**:
  metodi per trovare i risultati da inserire nelle risposte.
- **RisultatoSuggerimento**:
  metodo per trovare il suggerimento esatto per il quesito.
- **SceltaLivello**:
  metodo per decidere il tipo di argomento, logica o matematica e il relativo livello di gioco.
- **SoluzioneEsatta**:
  metodo per trovare la risposta esatta del quesito.

Classi che hanno lo scopo di simulare i livelli di gioco.

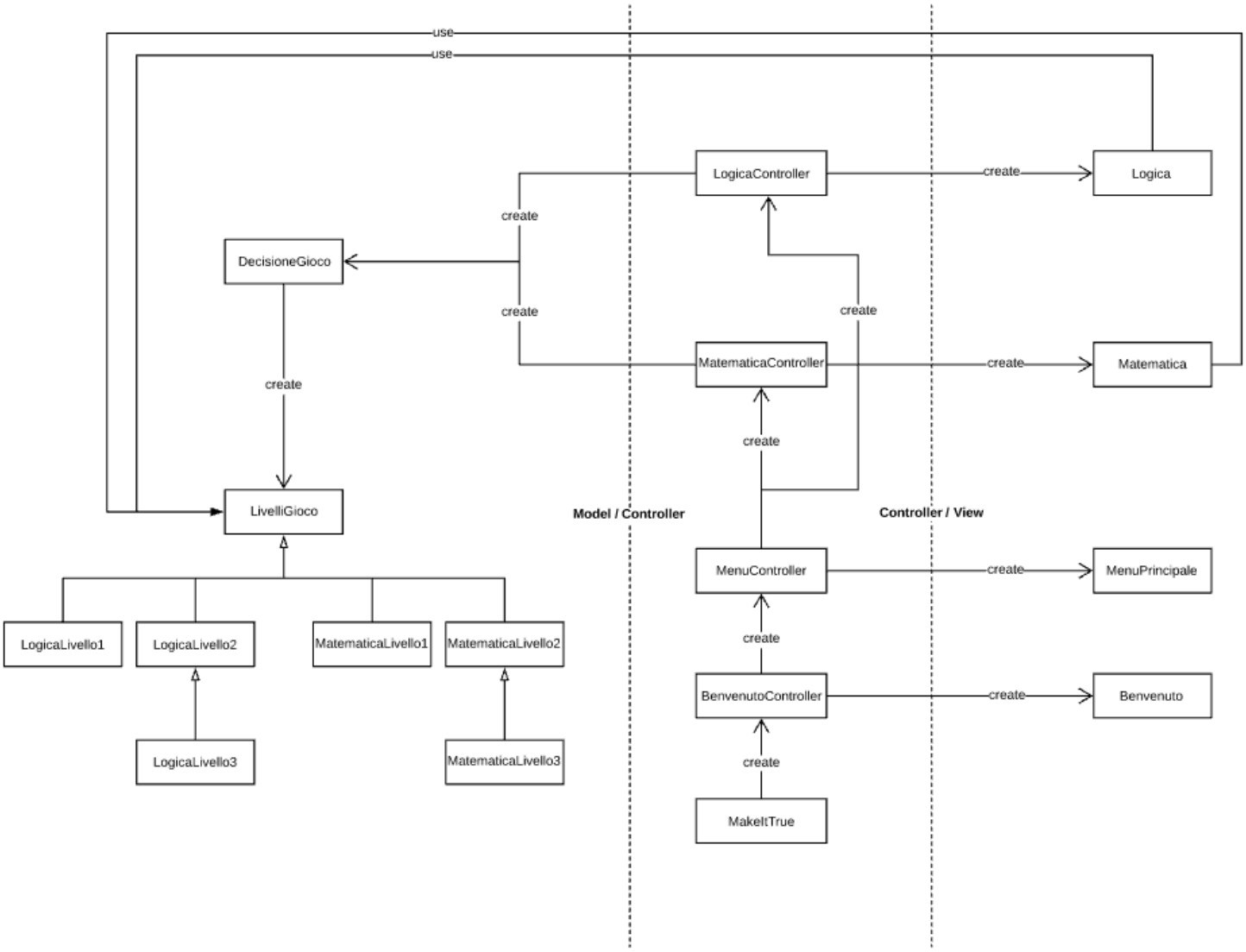


Come descritto dal diagramma, tutte le classi dei livelli sia di matematica che di logica derivano dalla classe padre LivelliGioco.
Si nota poi che i livelli 3 di entrambi gli argomenti ereditano tutte le caratteristiche dei livelli 2.

Infatti il livello 3 è caratterizzato dai quesiti del livello 2 ma come descritto prima, senza suggerimenti e con solo 5 domande, considerando poi la diminuzione di tempo ad ogni risposta errata.

Diagramma delle classi MVC per risaltare le relazioni che ci sono tra: model - controller - view.

I controller comunicano con la vista e con il modello (LogicaController e MatematicaController). La vista comunica con il modello attraverso le classi Logica e Matematica. Il modello risulta indipendente rispetto l'implementazione delle altre classi.



4. Documentazione sull'utilizzo

Passi da eseguire per la compilazione del gioco:

- Estrarre la cartella MakeItTrue dal file .zip MakeItTrue.zip (Ottenuto da GitHub)
- Doppio click sul file con estensione .sln ed attendere il caricamento del progetto
- Compilare la soluzione

Il file eseguibile dopo aver compilato sarà poi disponibile.

- Percorso → MakeItTrue/bin/Debug
- Doppio click sull'eseguibile MakeItTrue.exe

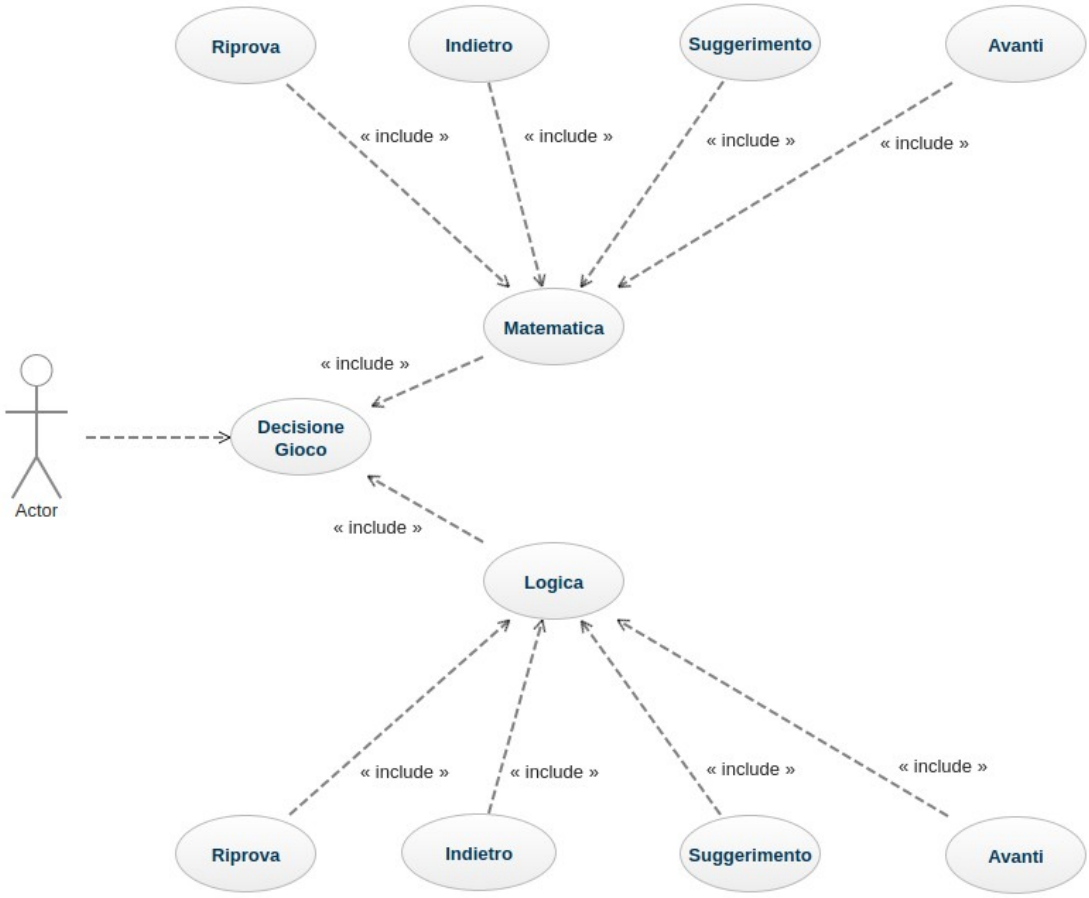
L'applicazione è stata eseguita sulle seguenti macchine:

Sistema Operativo	*RAM*	*CORE*	*Architettura*
Windows 10 Home	4 GB	4 core	64 bit
Windows 10 Pro	8 GB	4 core	64 bit

I test sono stati effettuati anche con Ubuntu 18.04.4 LTS usando Monodevelop.

Sistema Operativo	*RAM*	*CORE*	*Architettura*
Ubuntu	16 GB	8 core	64 bit

5. Use Cases con relativo schema UML



Descrizione dei casi d’uso più significativi

- Il giocatore inizia la partita e dopo il form Benvenuto avrà a disposizione un menù principale dal quale potrà decidere se affrontare l’argomento di matematica o di logica e con quale livello di difficoltà.

Caso d’uso	Inizia nuova partita con quesiti di matematica (livello 1)
ID	02
Attore	Giocatore
Precondizioni	Aver avviato il gioco
Corso d’azione base	Il giocatore sceglie il livello di matematica
Post condizioni	Form di matematica avviato
Percorso Alternativo	Livello 2 o 3 di matematica o cambio di argomento scegliendo logica

Caso d’uso	Inizia nuova partita con quesiti di logica (livello 1)
ID	05
Attore	Giocatore
Precondizioni	Aver avviato il gioco
Corso d’azione base	Il giocatore sceglie il livello di logica
Post condizioni	Il form di logica viene avviato
Percorso Alternativo	Livello 2 o 3 di logica o cambio di argomento scegliendo matematica

- Il giocatore sceglie di riprovare lo stesso livello e quindi si avvia una nuova partita

Caso d’uso	Riprova lo stesso livello di difficoltà che si sta giocando
ID	10
Attore	Giocatore
Precondizioni	Post condizioni 02 o 05 (e le altre post condizioni degli altri livelli)
Corso d’azione base	Il giocatore vuole giocare una nuova partita e clicca sul bottone “Riprova”
Post condizioni	Si resettano i quesiti e le risposte
Percorso Alternativo	Tornare indietro e giocare altri livelli

- Risposte del giocatore al quesito in modo corretto e Timer partita

Caso d’uso	Risposta corretta
ID	13
Attore	Giocatore
Precondizioni	Si è di fronte ad un quesito
Corso d’azione base	Il giocatore sceglie la risposta corretta
Post condizioni	Timer si ferma e viene considerata la domanda come corretta
Percorso Alternativo	Risposta errata, si riprova il test o si torna indietro

Caso d’uso	Tempo finito
ID	15
Attore	Timer del sistema
Precondizioni	Gioco in esecuzione
Corso d’azione base	La barra del timer si riempie totalmente determinando la fine della partita
Post condizioni	Avviso del tempo finito e partita terminata
Percorso Alternativo	Il giocatore raggiunge la fine della partita