SDD
System Design Document
DASHING CUBE

Riferimento	Nc11_dashingcube-sdd.docx
Versione	0.4
Data	08/01/2025
Destinatario	Prof. Carmine Gravino
Presentato da	Vincenzo Beniamino Fresa, Francesco Botta
Approvato da	

Revision History

Data	Versione	Descrizione	Autori
03/12/2024	0.1	Stesura dell'introduzione	Francesco Botta Vincenzo Beniamino Fresa
06/12/2024	0.2	Sottosistemi e architetture	Francesco Botta Vincenzo Beniamino Fresa
07/12/2024	0.3	Servizi Sottosistemi	Francesco Botta Vincenzo Beniamino Fresa
08/01/2025	0.4	Prima Revisione	Francesco Botta

Sommario

1.1 Obiettivo del Sistema

Un importante investitore italiano della città di Salerno è interessato ad entrare nel mondo videoludico. Per questo motivo ha indetto una gara per individuare i migliori sviluppatori capaci di realizzare il suo sogno.

Da qui nasce l'idea di Dashing Cube.

L'obiettivo del sistema di questo progetto è quello di fornire un gioco in grado di sfidare gli utenti in diversi livelli di difficoltà in modo da fornire un alto livello di rigiocabilità per portare gli utenti stessi a voler migliorare le loro abilità così da poter raggiungere nuovi record nel gioco.

Inoltre, il sistema fornirà un servizio di acquisto di prodotti virtuali in modo da poter tenere incollati i giocatori completisti, incentivando questo tipo di utenza a rimanere attivi nel gioco.

Il sistema fornirà diversi livelli di difficoltà per poter soddisfare un più vasto bacino di utenza e di mercato.

1.2 Design Goals

Il sistema si focalizzerà su particolari punti di design, seguendo determinate categorie:

- **Performance**: includono i requisiti di spazio e velocità imposti sul sistema.
- **Dependability**: determinano lo sforzo del sistema in caso di fallimento di esso cercando di ridimensionare gli eventuali danni (crash, falle di sicurezza) e le loro conseguenze.
- **Maintenance**: determina quanto sforzo è necessario per modificare il sistema dopo il rilascio.
- **End User**: includono qualità che sono desiderabili dal punto di vista dell'utente, a cui però non è stata data priorità rispetto ai criteri di Performance e Dependability.

N.B:

Parametri utilizzati per la descrizione dei design goals:

- **Rank**, che ne specifica un valore di priorità compreso tra 1 e il numero di design goals individuati.
- **ID**, un identificatore univoco e un nome esplicativo.
- **Descrizione,** una descrizione del design goal.
- **Categoria**, ovvero la categoria di appartenenza del design goal.
- **RNF,** che specifica il requisito che lo ha generato.

Design Goals

Rank	ID	Descrizione	Categoria	RNF
6	DG_1 Facilità d'uso	Il sistema deve risultare comprensibile attraverso l'uso delle "8 regole d'oro di Shneirderman" per il design delle interfacce grafiche.	End User	RNF_U_1
5	DG_2 Feedback immediato	Il sistema deve restituire un feedback chiaro ad ogni interazione dell'utente, in modo che questi possa comprendere facilmente il funzionamento del sistema.	End User	RNF_U_2
3	DG_3 Affidabilità delle operazioni	Il sistema deve garantire che tutte le operazioni avvengano con successo.	Dependability	RNF_A_1
4	DG_4 Gestione dei fallimenti	Il sistema deve essere resiliente ad errori e fallimenti che possono verificarsi durante l'utilizzo da parte dell'utente, notificando	Dependability	RNF_A_2

		quest'ultimo tramite dei messaggi.		
1	DG_5 Tempi di risposta	Il sistema deve garantire tempi di risposta di non oltre 1 decimo di secondo.	Performance	RNF_P_1 RNF_P_4
8	DG_6 Disponibilità delle informazioni	Il sistema deve garantire la fruizione di tutte le informazioni dove richiesto.	Performance	RNF_P_2
9	DG_7 Responsiveness	Il sistema deve mostrare un'interfaccia in grado di potersi adattare ad ogni tipo di schermo.	Performance	RNF_P_3
10	DG_8 Facilità di estensione	Il sistema deve essere sviluppato in modo tale da poter accogliere eventuali tecnologie future, seguendo gli standard IEEE e ISO, in modo da poter estendere le sue funzionalità.	Maintenance	RNF_S_1 RNF_IM_3

2	DG_9 Utilizzo hardware e software	Il sistema deve essere sviluppato come un applicazione indipendente e deve funzionare su hardware già disponibile al momento della progettazione dello stesso.	Maintenance	RNF_IM_1 RNF_IM_2
7	DG_10 Portabilità	Il sistema deve permettere l'utilizzo in più sistemi hardware possibili, non rendendo difficile la portabilità in sistemi nuovi	Maintenance	RNF_PA_1

Trade-off

Trade-off	Descrizione
Design semplice vs commerciabilità	La creazione di un design semplice ed intuitivo è sicuramente più economico e gestibile da realizzare. Il problema è che una fetta di utenti potrebbe considerarlo troppo semplice e quindi noioso.
Gestione dei fallimenti vs tempi di risposta	Se il sistema non soddisfa i requisiti di tempo di risposta o di velocità, è possibile utilizzare più memoria per accelerare il software. Se il software non soddisfa i vincoli di spazio di memoria, i dati possono essere compressi a discapito della velocità.
Riservatezza vs condivisione dei dati	Effettuare una netta distinzione tra le operazioni permesse a ciascun utente, può rendere più difficile la comunicazione e la condivisione dei dati tra gli utenti del sistema.

1.3 Definizioni, acronimi e abbreviazioni

Vengono riportati di seguito alcune definizioni presenti nel documento corrente:

- **Sottosistema:** un sottoinsieme dei servizi del dominio applicativo, formato da servizi legati da una relazione funzionale.
- **Design Goal:** le qualità sulle quali il sistema deve essere focalizzato.
- **Dati Persistenti**: dati che sopravvivono all'esecuzione del programma che li ha creati e che dunque vengono salvati.
- **Mapping Hardware/Software**: studio della connessione tra parti fisiche e logiche di cui si compongono il sistema.
- **SDD:** System Design Document.
- **RAD:** Requirements Analysis Document.
- **DG**: Design Goal
- **UC_BC**: Use Case Boundary Condition

1.4 Organizzazione del documento

- **Introduzione**: Viene descritto in generale lo scopo del sistema e gli obiettivi di design che il sistema propone di raggiungere.
- **Architettura software corrente**: Viene descritto lo stato attuale dell'architettura del software già presente.
- **Architettura software proposta**: Viene descritto come il sistema sarà definito e partizionato in sottosistemi, il loro mapping Hardware/Software, la gestione dei dati persistenti. Verranno poi presentate la struttura dei singoli sottosistemi e le boundary conditions riguardanti l'intero sistema.
- **Glossario:** Contiene la lista dei termini usati nel documento con annessa spiegazione.

2 Architettura del Sistema Corrente

Nel mercato videoludico attuale, l'esplosione dei videogiochi mobile e la conseguente presenza di un enorme numero di clienti di essi ha permesso molte compagnie di concentrarsi esclusivamente in questo tipo di mercato.

Quasi nessuno, però, ha avuto la visione di combinare l'idea dei videogiochi mobile in grado di poter catturare frange di popolazione altrimenti difficili da

attirare e i normali mezzi del medium videoludico (come il computer, le console home, ecc.). Giochi che si avvicinano a questa idea da noi proposta sono Geometry Dash, The Impossibile Game e Super Meat Boy.

3.1 Panoramica

L'architettura del sistema è di tipo **Three-layer**. Essendo un programma indipendente sviluppato internamente da un engine dedicato, abbiamo ritenuto ottimale svilupparlo nella maniera più semplice possibile ponendo dare risalto a tutte le meccaniche del sistema proposto.

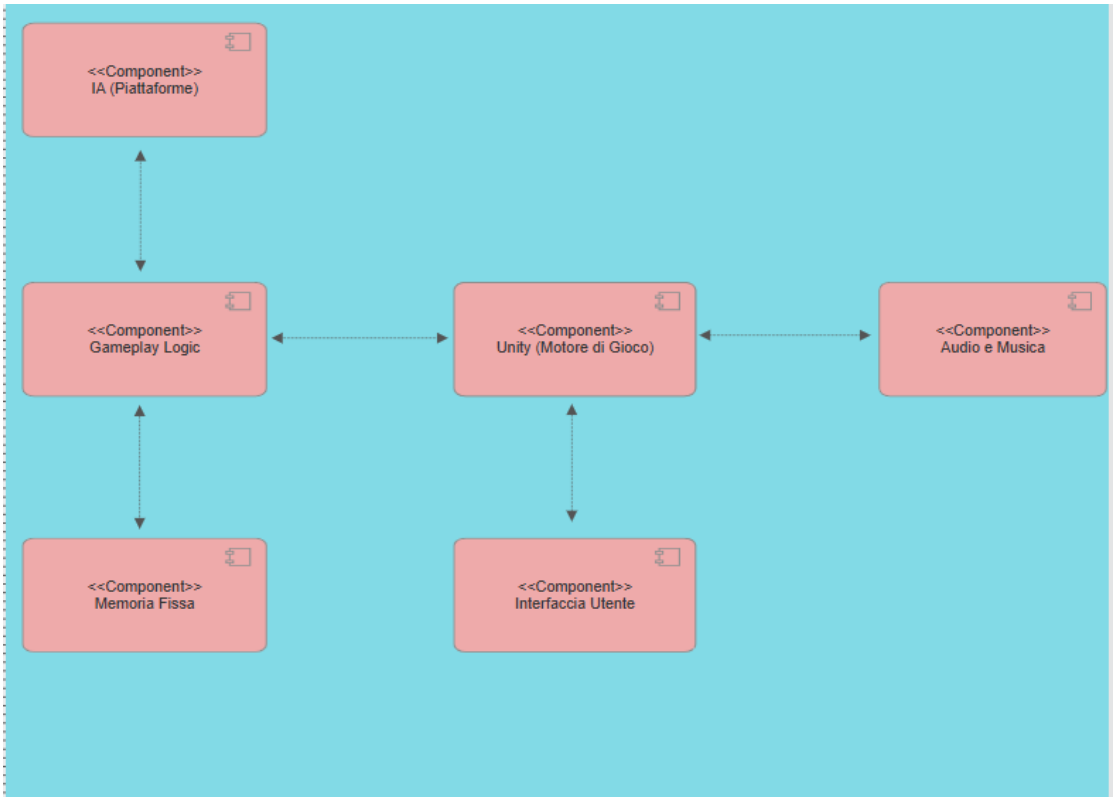
Per lo sviluppo di questo sistema verranno usate le seguenti tecnologie:

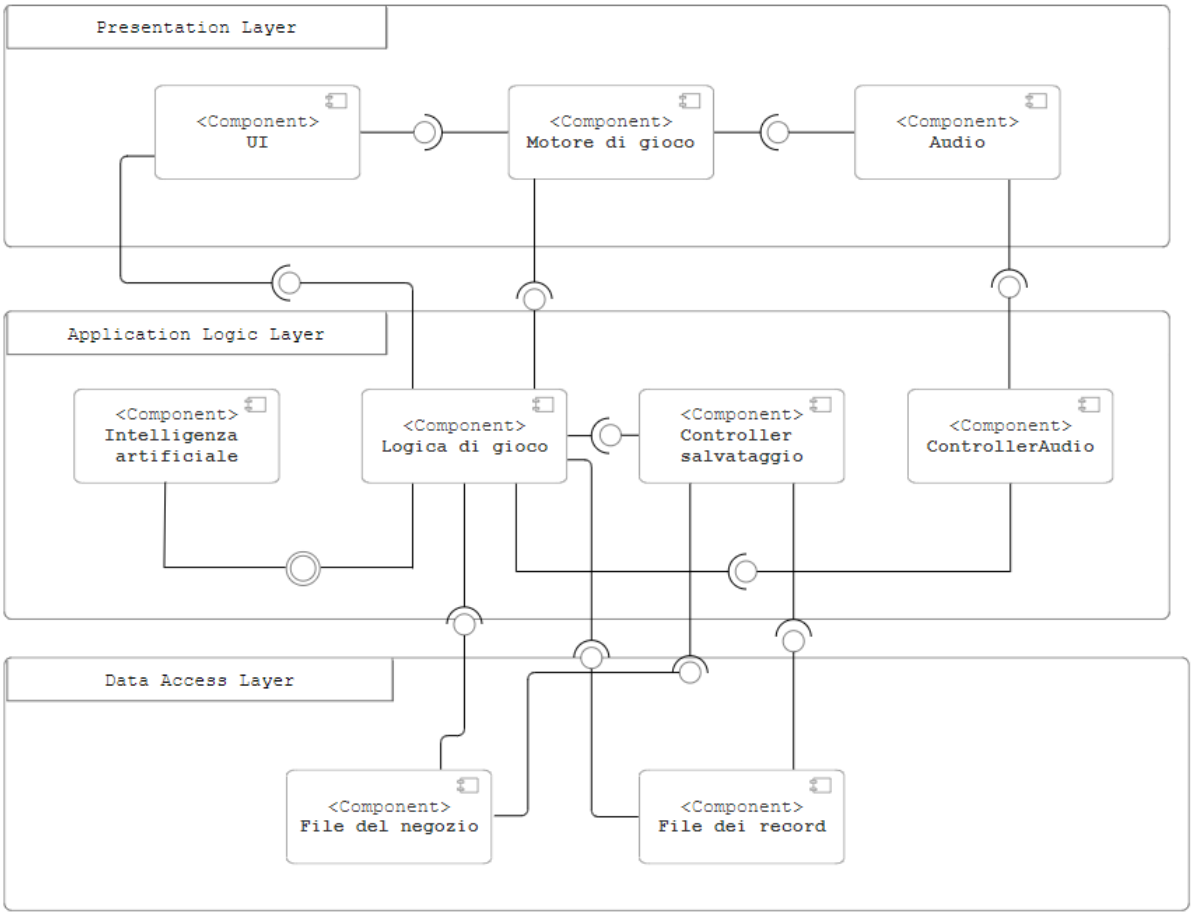
- **UNITY**
- **LIBRERIE AGGIUNTIVE**

3.2 Decomposizione in sottosistemi

Questa è la seguente suddivisione in sottosistemi:

- **Motore di Gioco (Unity):** Gestisce le funzionalità di base come rendering grafico, fisica, animazioni e input dell'utente. Include un sistema di gestione delle risorse come sprite, scripts, ecc.
- **Logica di Gioco (Gameplay Logic):** Contiene le regole del gioco, la gestione degli stati del gioco, e la logica degli eventi. Include sistemi per gestire il punteggio e i livelli.
- **Intelligenza Artificiale:** Responsabile della gestione del movimento e della creazione delle piattaforme nei vari livelli.
- **Interfaccia Utente:** Gestisce la presentazione delle informazioni al giocatore quali i pulsanti di menù, le monete, i record e i cosmetici acquistati.
- **Audio e Musica:** Si occupa della gestione degli effetti sonori, della musica di sottofondo. Include moduli per il controllo del volume, la sincronizzazione audio con eventi di gioco e l'audio spaziale.
- **Memoria fissa:** Gestisce la memorizzazione dei vari record nei livelli, delle monete ottenute e dei cosmetici acquistati che potranno essere visualizzati in seguito.


<<Component>>
IA (Piattaforme)
<<Component>>
Gameplay Logic
<<Component>>
Unity (Motore di Gioco)
<<Component>>
Audio e Musica
<<Component>>
Memoria Fissa
<<Component>>
Interfaccia Utente


Presentation Layer
<Component>
UI
<Component>
Motore di gioco
<Component>
Audio
Application Logic Layer
<Component>
Intelligenza artificiale
<Component>
Logica di gioco
<Component>
Controller salvataggio
<Component>
ControllerAudio
Data Access Layer
<Component>
File del negozio
<Component>
File dei record

Con l'uso dell'architettura **Three-layer** abbiamo suddiviso i sottosistemi in modo tale che questi possano interagire tra di loro in maniera efficiente, rispettando le regole imposte dall'architettura usata.

Nel livello **Presentazione** abbiamo inserito il motore di gioco, perno centrale dell'esperienza del prodotto e scheletro fondante di esso dove creerà interfacciandosi con l'UI, la parte video e quella audio, il sistema in cui il giocatore potrà interagire.

Nel livello **Applicazione** abbiamo la logica di gioco gestita dai vari scripts creati all'interno del motore di gioco stesso (oggetti in grado di gestire le varie meccaniche di gioco). Inoltre, abbiamo una piccola componente di Intelligenza Artificiale che controlla il movimento e la creazione delle piattaforme su cui il giocatore si dovrà muovere.

Nel Livello **Data Access** abbiamo 2 componenti: il file di acquisto e memorizzazione dei cosmetici e delle monete raccolte e il file dei record.

3.3 Mapping Hardware/Software

Il gioco sviluppato sarà disponibile attraverso download. L'applicazione è pensata per essere supportata ed eseguita inizialmente su PC attraverso i porting fornity da Unity. Un futuro aggiornamento potrebbe portare a porting per altri dispositivi. La volontà di rilasciarlo soltanto per PC inizialmente è stata dettata da una maggiore performance in termini di prestazioni, tempi di risposta e velocità di caricamenti.

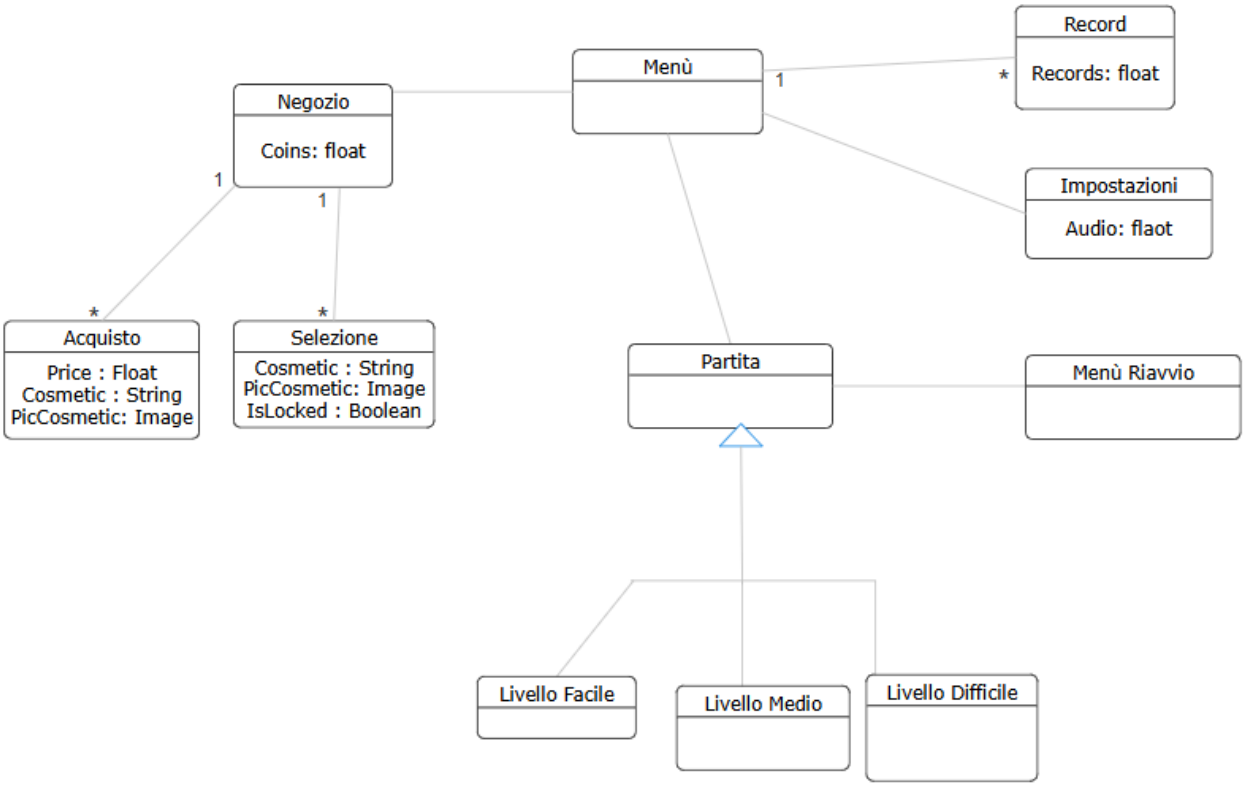
3.4 Gestione dei dati Persistenti

Per la gestione dei dati persistenti del sistema, si è optato di utilizzare un sistema di memorizzazione file multipli in modo da gestire l'accesso ai dati e garantire continuità all'esperienza dell'utente.

Questa scelta è stata presa tenendo conto degli obiettivi prefissati, garantendo una semplicità nella progettazione del sistema, e una semplificazione nel mantenimento di esso.

L'engine UNITY è sufficentemente equipaggiato per la scrittura e la lettura dei file di memorizzazione.

Class Diagram



3.5 Controllo globale del software

Il flusso del sistema fornisce diverse funzionalità che richiedono una continua interazione da parte dell'utente, ragione per cui, il controllo del flusso globale è di tipo **event-driven** ovvero guidato dagli eventi. Quando si verifica un evento, vengono selezionati i sottosistemi che si occupano della logica applicativa e della gestione dei dati persistenti.

3.6 Condizioni Limite

In questo paragrafo osserveremo il comportamento del sistema nei momenti critici, ovvero l'avviamento, la chiusura, il crash e il fallimento del recupero dei dati persistenti.

Avvio del sistema

Identificativo	*UCBC1- Avvio del Sistema*	*Data*	06/12/24
		Vers.	1.0

		Autore	Vincenzo Beniamino Fresa
Descrizione	Avvio del sistema da parte del giocatore		
Attore Principale	Giocatore		
Attori secondari	NA		
Entry Condition	Il Gioco è installato		
Exit condition On success	Il Gioco si avvia correttamente		
Exit condition On failure	Il Gioco non viene avviato correttamente		

FLUSSO DI EVENTI PRINCIPALE		
1	Giocatore	Clicca due volte l'icona del gioco.
2	Sistema	Esegue l'applicazione.

SCENARIO/FLUSSO DI EVENTI ALTERNATIVO		
2.a1	Sistema	Notifica al giocatore che non è stato possibile avviare il gioco.
2.a2	Giocatore	Riparte dal punto 1.

Spegnimento del sistema

Identificativo	*UCBC2-*	*Data*	06/12/24

	Spegnimento del Sistema	*Vers.*	1.0
		Autore	Botta Francesco
Descrizione	Spegnimento del sistema da parte del giocatore		
Attore Principale	Giocatore		
Attori secondari	NA		
Entry Condition	Il gioco è avviato		
Exit condition On success	Il gioco si arresta correttamente		
Exit condition On failure	Il gioco non viene arrestato correttamente		

FLUSSO DI EVENTI PRINCIPALE		
1	Giocatore	Clicca sul pulsante di spegnimento
2	Sistema	Prima salva i dati dell'utente e immediatemnte dopo procede all'arresto del gioco

Crash del sistema

Identificativo	*UCBC3- Crash del sistema*	*Data*	06/12/24
		Vers.	1.0
		Autore	Vincenzo Beniamino Fresa

Descrizione	Il sistema smette di funzionare improvvisamente
Attore Principale	Giocatore
Attori secondari	NA
Entry Condition	Il Gioco non risponde ai comandi
Exit condition On success	Il gioco viene riavviato dall'utente
Exit condition On failure	Il gioco non pu essere riavviato

FLUSSO DI EVENTI PRINCIPALE		
1	Sistema	Notifica che c'è un errore
2	Giocatore	Tenta di riavviare il gioco

Errore di accesso ai dati persistenti

Identificativo	*UCBC4- Errore di accesso ai dati persistenti*	*Data*	06/12/24
		Vers.	1.0
		Autore	Botta Francesco
Descrizione	Comportamento dell'utente in caso di errore di accesso ai dati persistenti da parte del gioco		
Attore Principale	Giocatore		
Attori secondari	NA		
Entry Condition	Il gioco non riesce ad accedere ai dati persistenti OR I dati risultano corrotti		

Exit condition On success	Il Gioco riprende il suo normale funzionamento
Exit condition On failure	Il Gioco non riprende il suo normale funzionamento
FLUSSO DI EVENTI PRINCIPALE	
1 Sistema	Restituisce un messaggio di errore e rimanda alla schermata di salvataggio

4 Servizi dei sottosistemi

Motore di gioco:

Servizio	Descrizione	Interfaccia
RenderizzaGioco	Permette il render del sistema, mostrando a schermo la parte UI, audio e video del programma	GameRender

Logica del gioco:

Servizio	Descrizione	Interfaccia
Scelta livello di difficoltà	Gestisce la funzionalità per accedere ad uno dei livelli di difficoltà	DifficultySelection
Controllo giocatore	Gestisce la funzionalità che permette il salto del giocatore all'utente	MenuController
Pausa	Gestisce la funzionalità che permette la pausa durante una sessione.	MenuController
Riavvio sessione	Gestisce la funzionalità di avviare una nuova sessione al termine di una precedente	MenuController

Intelligenza Artificiale:

Servizio	Descrizione	Interfaccia
Movimento piattaforme	Gestisce la funzionalità che permette di muovere e modificare le piattaforme durante la sessione di un livello	GameSession

Interfaccia Utente:

Servizio	Descrizione	Interfaccia
Visualizza monete acquisite	Gestisce la funzionalità di visualizzare le monete acquisite durante le sessioni di gioco	GameSession
Visualizza record	Gestisce la funzionalità che permette di visualizzare i punteggi migliori fatti.	MenuController
Visualizza negozio	Gestisce la funzionalità che permette di accedere al negozio.	MenuController
Visualizza Menu	Permette di mostrare il menu principale.	MenuController

Audio e musica:

Servizio	Descrizione	Interfaccia

Attivazione effetti sonori	Gestisce la funzionalità di eseguire effetti sonori.	AudioController
Esecuzione musica	Permette di far partire la musica all'interno del sistema.	AudioController

Memoria fissa:

Servizio	Descrizione	Interfaccia
Salvataggio monete	Conserva il numero di monete all'interno di una banca dati.	Database
Salvataggio cosmetici	Memorizza quali cosmetici sono stati acquistati dal giocatore.	Database
Salvataggio record	Conserva i 3 migliori punteggi effettuati.	Database

5 Glossario

Termine	Definizione
Record	Elenco dei risultati delle varie sessioni di gioco ottenuto dal giocatore, completo di tutti i dati relativi.
Sprite	Immagine in grafica bidimensionale (2D), che fa parte di una scena più grande (lo "sfondo") e che può essere spostata in maniera indipendente rispetto ad essa
Porting	Processo di trasposizione, a volte anche con modifiche, di un componente software, volto a consentirne l'uso in

	una piattaforma diversa da quella originale
Giocatori Completisti	Giocatori che hanno l'obiettivo di sbloccare tutti i collezionabili di un gioco, in questo caso i possibili cosmetici presenti nel negozio.
Scripts	Programma o sequenza di istruzioni che viene interpretata o portata a termine da un altro programma (in questo caso Unity)