



Università degli Studi di Messina

Dipartimento di Scienze Matematiche, Informatiche, Fisiche e Scienze della Terra
Corso di Laurea Triennale in Informatica

Relazione di Ingegneria del Software
A.A. 2021-2022

“ESCAPE IN THE SPACE”

Prof. Distefano Salvatore

Buemi Emanuele - Giambò Filippo

1. Introduzione	5
1.1. Scopo	5
1.2. Descrizione del progetto	5
1.3. Escape in the space – Storia	6
1.4. Tecnologie utilizzate	6
2. Modello a prototipi	7
2.1. Definizioni	7
2.1.1. Throw-away	8
2.1.2. Evolutivo	8
2.2. Metodologia di sviluppo utilizzata	8
3. Pianificazione iniziale	9
3.1. Studio di fattibilità	9
3.2. Analisi dei rischi	9
3.3. Pianificazione delle iterazioni	9
4. Primo prototipo	10
4.1. Analisi dei requisiti	10
4.1.1. Glossario	10
4.1.2. Requisiti	10
4.1.2.1. Requisiti funzionali	11
4.1.2.2. Requisiti non funzionali	12
4.2. Design	12
4.2.1. UML	12
4.2.2. Use case diagram	13
4.2.2.1. Use case Utente – 1° prototipo	13
4.2.3. Activity Diagram	17
4.2.3.1. Activity Diagram Login/Registrazione Utente – 1° prototipo	18
4.2.3.2. Activity Diagram Account Utente – 1° prototipo	18
4.2.3.3. Activity Diagram Gioco Utente – 1° prototipo	19
4.2.4. Sequence Diagram	20
4.2.4.1. Sequence Diagram Login/Registrazione Utente – 1° prototipo	21
4.3. Implementazione	22
4.3.1. Classe FinestraBenvenuto	22
4.3.2. Classe FinestraConLogin	22
4.3.3. Classe FinestraConRegistrazione	23
4.3.4. Classe FinestraBottoni	23
4.3.5. Classe Spazio	24
4.3.6. Classe RiproduciSuono	24
4.3.7. Classe Oggetto	25
4.4. Testing	25
4.4.1. Verifica	26
4.4.2. Validazione	26
4.4.3. Valutazione 1° prototipo con feedback del client	28

5. Secondo prototipo	29
5.1. Analisi dei requisiti	29
5.1.1. Glossario	29
5.1.2. Requisiti	29
5.1.2.1. Requisiti funzionali	29
5.1.2.2. Requisiti non funzionali	31
5.2. Design	31
5.2.1. Use case diagram	31
5.2.1.1. Use case Giocatore – 2° prototipo	32
5.2.1.2. Use case Admin – 2° prototipo	35
5.2.2. Activity Diagram	38
5.2.2.1. Activity Diagram Login/Registrazione – 2° prototipo	38
5.2.2.2. Activity Diagram Gioco – 2° prototipo	39
5.2.2.3. Activity Diagram Account Admin – 2° prototipo	40
5.2.3. Sequence Diagram	40
5.2.3.1. Sequence Diagram Login Admin – 2° prototipo	40
5.3. Implementazione	41
5.3.1. Classe FinestraConLogin	41
5.3.2. Classe FinestraAdminHome	41
5.3.3. Classe Utenti	41
5.3.4. Classe FinestraBottoni	42
5.3.5. Classe Spazio	42
5.3.6. Classe Oggetto	42
5.4. Testing	42
5.4.1. Verifica	43
5.4.2. Validazione	44
5.4.3. Valutazione 2° prototipo con feedback del client	46
6. Terzo prototipo	47
6.1. Analisi dei requisiti	47
6.1.1. Glossario	47
6.1.2. Requisiti	47
6.1.2.1. Requisiti funzionali	47
6.1.2.2. Requisiti non funzionali	49
6.2. Design	49
6.2.1. Use case diagram	49
6.2.1.1. Use case Utente Ospite – 3° prototipo	50
6.2.1.2. Use case Modalità Gioco – 3° prototipo	51
6.2.1.3. Use case Carriera – 3° prototipo	53
6.2.1.4. Use case Multiplayer – 3° prototipo	55
6.2.2. Activity Diagram	56
6.2.2.1. Activity Diagram Recupera password – 3° prototipo	56
6.2.2.2. Activity Diagram Modalità Gioco – 3° prototipo	57
6.2.3. Sequence Diagram	58
6.2.3.1. Sequence Diagram Recupera password – 3° prototipo	58
6.2.3.2. Sequence Diagram Partita multiplayer - 3°prototipo	59
6.3. Implementazione	59
6.3.1. Classe FinestraBenvenuto	60
6.3.2. Classe FinestraOspite	60
6.3.3. Classe FinestraConLogin	60

6.3.4.	Classe FinestraRecuperaPassword	60
6.3.5.	Classe FinestraModalitaGioco	61
6.3.6.	Classe FinestraBottoni	61
6.3.7.	Classe Client	62
6.3.8.	Classe ClientHandler	62
6.3.9.	Classe GameServer	62
6.3.10.	Classe PlayerClient	63
6.3.11.	Classe FinestraFineGameMultiplayer	63
6.4.	Component Diagram e Deployment Diagram	64
6.4.1.	Component/Deployment Diagram -Escape in the space	65
6.5.	Testing	65
6.5.1.	Verifica	65
6.5.2.	Validazione	67
6.5.3.	Valutazione 3° prototipo con feedback del client	70
7.	Meccaniche del gioco	71
7.1.	Schermata di benvenuto	71
7.2.	Schermata ospite	72
7.3.	Schermata di login	72
7.4.	Schermata per recuperare la password	73
7.5.	Schermata di registrazione	73
7.6.	Schermata per la scelta della modalità di gioco	74
7.6.1.	Schermata per la partita multiplayer	75
7.7.	Schermata dell'account per la modalità carriera	76
7.7.1.	Schermata per la partita singola	77
7.8.	Schermata dell'account admin	78

1. Introduzione

1.1. Scopo

Questo documento è stato stilato per avere come principale obiettivo quello di essere utilizzato per definire gli strumenti e le metodologie per la pianificazione, l'analisi, la progettazione, lo sviluppo e il testing del software.

Queste fasi vanno a definire il ciclo vita del software.

1.2. Descrizione del progetto

Il progetto prevede lo sviluppo di un software capace di ricreare un gioco ambientato nello spazio nel quale il giocatore dovrà difendersi schivando e colpendo i meteoriti e le navicelle nemiche ottenendo più punti possibili. **Il razzo**, seguendo i movimenti del mouse o delle freccette, si muoverà verso destra e sinistra e sarà compito proprio del giocatore quello di proteggerlo dai meteoriti e dalle navicelle nemiche. Inoltre il giocatore, tramite l'ausilio del click sulla barra spaziatrice (o spazio), ha la possibilità di **sparare dei missili** per contrastare gli oggetti nemici e incrementare il punteggio (per ogni oggetto nemico colpito verrà assegnato un punteggio uguale a 8).

A ogni oggetto nemico verrà assegnato un punteggio. Gli oggetti nemici sono:

- **I meteoriti**, possono essere di tre tipi, i quali indicano la pericolosità:
 - **Marroni** → hanno una dimensione maggiore rispetto agli altri e per questo a essi sarà attribuito un punteggio uguale a 2 quando verranno schivati;
 - **Grigi** → hanno una dimensione intermedia tra i marroni e gli arancioni e per questo a essi sarà attribuito un punteggio uguale a 3 quando verranno schivati;
 - **Arancioni** → hanno una dimensione minore rispetto a tutti gli altri e per questo a essi sarà attribuito un punteggio uguale a 4 quando verranno schivati;
- **Le navicelle**, ai quali verrà assegnato un punteggio uguale a 5 quando verranno schivati.

Il gioco prevede dei livelli di difficoltà crescenti in cui si avrà un aumento della velocità di caduta degli oggetti nemici quando si passerà al livello successivo.

Quando il giocatore esaurisce le vite a sua disposizione il gioco termina.

Scopo del gioco è quindi ottenere il maggior punteggio possibile.

Il programma prevede la possibilità di poter creare i profili dei giocatori, all'interno di un DBMS locale.

Il gioco concepirà due tipologie di utenti:

- i giocatori: che possono scegliere se entrare come ospiti, visionando immagini inerenti al gioco e vedere delle recensioni, o giocare creando un proprio profilo che verrà memorizzato all'interno del DBMS MySQL locale.
Il giocatore può scegliere di poter giocare:
 - in modalità carriera (giocando partite singole cercando di superare i punteggi degli altri utenti e ottenere il miglior risultato);
 - in modalità multiplayer (provando a sfidare in contemporanea un altro utente).
- l'admin: che avrà il ruolo manageriale di gestione degli utenti e dei commenti inseriti dagli utenti giocatori.

1.3. Escape in the space – Storia

Escape in the space è un gioco ambientato nello spazio ispiratosi al famosissimo gioco Space Invaders. Esso prevede di indossare i panni di un pilota aerospaziale e di difendere il razzo dai meteoriti e dalle navicelle che infestano la galassia. Il suo scopo è quello di stimolare la fantasia del giocatore proiettandolo in un mondo in 2 dimensioni.

Lo spazio infinito, i pianeti e le galassie hanno sempre occupato un posto speciale nel cuore degli sviluppatori e degli amanti dei videogiochi. Basti pensare che sono passati quasi 50 anni dalla nascita di “Spacewar”, titolo sperimentale creato dai ricercatori del MIT e riconosciuto come il primo simulatore di volo e combattimento spaziale. E da quel giorno il legame tra gaming e spazio non si è più sciolto ma, al contrario, si è evoluto fino a dare vita a vere e proprie opere d’arte video-ludiche.

La cultura popolare vuole che il primo space gaming della storia sia “Space Invaders” del 1978. In realtà è più opportuno affermare che Invaders sia il primo gioco entrato nell’immaginario collettivo ma non il primo in assoluto sul tema. Abbiamo già parlato di “Spacewar” del 1962. Negli anni ’70 arrivarono nelle sale anche “Starship 1”, “Lunar Lander”, gioco del 1973 basato sull’allunaggio, la serie “Galaxian/Galaga” e “Asteroids”. Leggermente successivo “Defender” del 1981, uno dei primi titoli a mettere in scena battaglie tra astronavi aliene e navicelle umane.

Nel periodo successivo si fece strada anche l’idea di esplorazione umana delle galassie. In questo filone rientrano alcuni dei più famosi spara-tutto in prima persona: “Doom” del 1993, ambientato su Marte come i successivi “Red Faction: Armageddon” del 2011 e “Mass Effect 3” del 2012 e “Duke Nukem 3D”, grande classico con ambientazione lunare.

1.4. Tecnologie utilizzate

Le tecnologie che sono state utilizzate durante il ciclo di vita del software sono molteplici, volte ad un’ottima implementazione dello stesso.

Java → La struttura del progetto che si è scelto di creare si basa molto su una programmazione ad oggetti modulare, per questo motivo si è scelto di implementare il software utilizzando il linguaggio Java che tra gli innumerevoli vantaggi ha quello più importante che permette l’indipendenza del progetto dalla piattaforma su cui è stato sviluppato, ma dato che è un linguaggio orientato agli oggetti, permette di riutilizzare il codice e di gestire progetti anche di grandi dimensioni.

NetBeans → IDE per l’ottimizzazione del linguaggio Java;

Lucid.app → Per la creazione dei grafici UML. Questi diagrammi UML vengono utilizzati perché, UML oltre ad essere uno standard ha anche i seguenti vantaggi:

- Un sistema software grazie al linguaggio UML viene disegnato professionalmente e documentato ancor prima che ne venga scritto il relativo codice dagli sviluppatori.
- Poiché la fase di disegno del sistema precede la fase di scrittura del codice, ne consegue che la scrittura del codice stessa è resa più agevole ed efficiente.
- È più facile prevedere e anticipare eventuali “bug nel sistema”.
- L'utilizzo dei diagrammi UML permette di avere un'idea chiara a chiunque sia coinvolto nello sviluppo di tutto l'insieme che costituisce il sistema.
- Grazie alla documentazione del linguaggio UML diviene ancora più facile effettuare eventuali modifiche future al codice (a beneficio dei costi di mantenimento del sistema).

MySQL DB → Per il salvataggio di dati relativi al gioco.

MySQL è un database che ha una perfetta associazione con il linguaggio Java grazie all'utilizzo del driver JDBC “*Java Database Connectivity*”.

Java Sound → Una API low-level che include il supporto per l'audio digitale. In particolare viene utilizzato il pacchetto `javax.sound.sampled`, che specifica le interfacce per l'acquisizione e la riproduzione di audio digitale;

Swing → Framework per Java, appartenente alle Java Foundation Classes e orientato allo sviluppo di interfacce grafiche. Parte delle classi del framework Swing sono implementazioni di widget (oggetti grafici) come caselle di testo, pulsanti, pannelli e tabelle;

Awt → Libreria Java contenente le classi e le interfacce fondamentali per il rendering grafico. Queste classi consentono di realizzare interfacce utente complesse e di definire l'interazione attraverso la specifica di elementi e di gestori degli stessi;

Word → Per la stesura della documentazione necessaria.

2. Modello a prototipi

2.1. Definizioni

La prototipazione software è l'attività di creazione di prototipi di applicazioni software, ovvero versioni incomplete del programma software in fase di sviluppo che, anche se in qualche modo limitate, possono essere utilizzate a scopo di valutazione.

Il modello a prototipi è una tipologia di modello di sviluppo software che appartiene al gruppo dei **modelli evolutivi**, ovvero quelle metodologie che cercano di superare i limiti principali del modello a cascata, come la vincolabilità alla documentazione.

Nel modello a prototipo, il progettista e lo sviluppatore del software possono ottenere un feedback prezioso dagli utenti nelle prime fasi del progetto e questo permetterà non solo, di perfezionare i requisiti, ma soprattutto di ridurre i costi, poiché apportare modifiche nelle prime fasi del ciclo di vita dello sviluppo diventa estremamente conveniente.

A questo punto il cliente e lo sviluppatore si confronteranno per verificare se il software realizzato corrisponde alle specifiche, in base alle quali è costruito il programma software.

Grazie a questi feedback il software, quindi, evolve per versioni successive, dette release, e ogni nuova release costituisce un miglioramento rispetto a quella precedente, cioè avviene:

- La correzione di errori;
- Il perfezionamento delle funzionalità esistenti;
- L'introduzione di nuove funzionalità;
- La compatibilità con ambiente più evoluto.

Con la prototipazione software possiamo avere due approcci principali:

- **Throw-away**,
- **Evolutivo**.

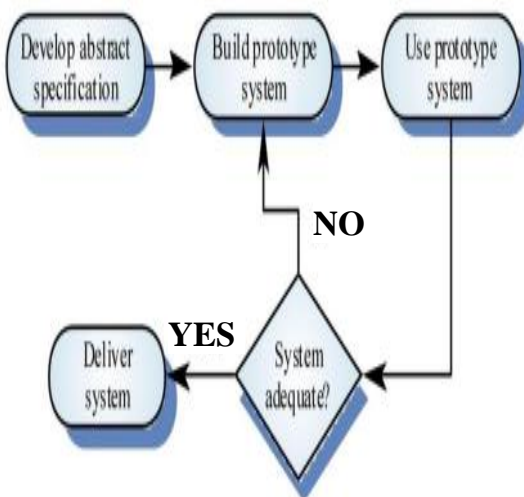
2.1.1. Throw-away

La prototipazione **Throw-away (usa e getta)** si riferisce alla creazione di un modello che alla fine verrà scartato piuttosto che diventare parte del software finale consegnato. Una volta completata la raccolta dei requisiti preliminari, viene costruito un semplice modello di lavoro del sistema per mostrare visivamente agli utenti come possono apparire i loro requisiti quando vengono implementati in un sistema finito. La ragione più ovvia per utilizzare la prototipazione usa e getta è che può essere eseguita rapidamente. La velocità è fondamentale nell'implementazione di un prototipo usa e getta, poiché con un budget limitato di tempo e denaro si può risparmiare per un prototipo che verrà scartato.

Nonostante questi aspetti positivi, si predilige adottare l'approccio evolutivo perché, con l'approccio Throw-away abbiamo che la struttura del prototipo si degrada a causa di rapidi cambiamenti. Inoltre i prototipi sono normalmente privi di documentazione e questo rende impossibile regolare il sistema per soddisfare requisiti non funzionali.

2.1.2. Evolutivo

La prototipazione **evolutiva** (nota anche come prototipazione breadboard) è molto diversa dalla prototipazione usa e getta. L'obiettivo principale quando si utilizza la prototipazione evolutiva è costruire un prototipo molto robusto in modo strutturato e perfezionarlo costantemente. La ragione di questo approccio è che il prototipo evolutivo, una volta costruito, costituisce il cuore del nuovo sistema che viene continuamente perfezionato e ricostruito apportando i miglioramenti e definendo nuovi requisiti. Vediamo come si svolge lo sviluppo di un prototipo evolutivo.



Nella **1° fase** → Il cliente valuta il prototipo e conferma se esso è sviluppato secondo la sua richiesta o necessita di modifiche.

Nella **2° fase** → Se la valutazione data dal cliente è negativa, lo sviluppatore considera il feedback fornito dal cliente e modifica il prototipo in base alla sua richiesta. Queste iterazioni (tra sviluppatore e cliente) continuano fino a quando il cliente non è soddisfatto del prototipo.

Nella **3° fase** → Una volta che il cliente è soddisfatto, cioè tutte le sue richieste sono state applicate correttamente il software verrà consegnato al cliente.

I benefici di questo approccio sono molteplici, come migliorare la qualità e la completezza delle specifiche del progetto, permettere di incorporare nel progetto i feedback degli utenti, migliore usabilità del sistema, ridurre gli sforzi durante lo sviluppo ecc...

2.2. Metodologia di sviluppo utilizzata

All'interno di questo progetto la metodologia di sviluppo software adoperata dal team è quella del **prototype evolutivo** nel quale un prototipo viene costruito, testato e perfezionato secondo le esigenze del cliente fino a ottenere il sistema finale.

3. Pianificazione iniziale

3.1. Studio di fattibilità

Questo studio si propone di valutare oltre i possibili risultati anche la necessità di rispettare le scadenze previste ed anche i costi necessari per l'implementazione. Dopo un'accurata analisi dei requisiti il progetto risulta fattibile per le seguenti motivazioni:

- Il dominio applicativo è sicuramente codificabile in termini di algoritmi e strutture dati ed è matematicamente ben studiato, e ciò aiuta lo sviluppo dell'applicazione.
- Gli strumenti a disposizione per gli sviluppatori sono sufficienti alla realizzazione del progetto.
- La scelta di utilizzare un'organizzazione modulare garantisce la piena collaborazione del team.
- I tempi richiesti di realizzazione delle specifiche sono più che sufficienti.
- I costi richiesti per l'implementazione sono più che sostenibili.

3.2. Analisi dei rischi

I rischi connessi al progetto software in oggetto sono principalmente legati alla saturazione del mercato in merito ai giochi inerenti la battaglia nello spazio e quindi riguardano la possibilità di non riuscire a catturare un target di pubblico sufficientemente vasto per il software sviluppato. Per tale ragione è necessario insistere su un mercato che sappia catturare un target che sia il più ampio possibile, il che significa insistere su un prodotto con caratteristiche di immediatezza e user-friendliness, e che sappia accontentare giocatori principianti ed esperti, senza allontanare ma anzi facilitando i primi e permettendo ai secondi di fare pratica in modo consono alle proprie capacità.

3.3. Pianificazione delle iterazioni

Il seguente documento prevede 3 iterazioni del software, coerenti con la metodologia utilizzata per descrivere il ciclo di vita del software, ovvero la prototipizzazione evolutiva:

- Nella **prima iterazione** abbiamo: implementato le meccaniche più semplici che permettono a un utente registrato e loggato di giocare, di vedere le istruzioni relative al gioco e di attivare l'audio.
- Nella **seconda iterazione** abbiamo: implementato la parte relativa all'admin del gioco che può visualizzare gli utenti registrati e eliminare un utente o un commento, aggiunto funzionalità in più al gioco come la possibilità di mettere il gioco in pausa, la funzionalità di potersi difendere sparando dei missili contro gli oggetti nemici, giocare una partita con un livello di difficoltà crescente, visualizzare il migliore punteggio raggiunto dall'utente e aggiungere delle recensioni sul gioco.
- Nella **terza ed ultima iterazione** abbiamo: migliorato la grafica in tutte le parti del gioco, implementato la parte relativa all'utente ospite il quale può visualizzare immagini e commenti sul gioco, aggiunto le funzionalità di visualizzare la classifica dei migliori 5 punteggi, recuperare la password e la possibilità di scegliere se giocare in modalità carriera o multiplayer.

4. Primo prototipo

In questo primo prototipo il team ha deciso di implementare una prima versione base del gioco “**Escape in the space**”. L’obiettivo, come già detto, è quello di avere un primo feedback sul software, così da poterlo migliorare successivamente.

4.1. Analisi dei requisiti

L’analisi dei requisiti è un’attività preliminare allo sviluppo (o alla modifica) di un sistema software, il cui scopo è quello di definire le funzionalità in modo univoco che il nuovo prodotto (o il prodotto modificato) deve offrire, ovvero i requisiti che devono essere soddisfatti dal software sviluppato.

4.1.1. Glossario

Adesso andiamo a definire alcuni termini che vedremo più avanti nel progetto:

- **Login:** termine utilizzato per l’inserimento delle credenziali dell’utente;
- **Registrazione:** termine utilizzato per la creazione di un profilo utente;
- **Partita:** cioè il tempo che intercorre tra l’inizio e la fine del gioco;
- **Giocatore:** colui che gioca la partita;
- **Punteggio:** indica il punteggio ottenuto dal giocatore durante la partita;
- **Vita:** indica la salute del giocatore;
- **Razzo:** indica l’oggetto che viene mosso dal giocatore;
- **Movimento:** indica lo spostamento del razzo che il giocatore può effettuare tramite il mouse o le freccette direzionali;
- **Meteoriti e navicelle nemiche:** indicano gli oggetti dal quale il giocatore deve difendersi;
- **Collisione:** è quell’evento che si manifesta nel momento in cui il razzo viene a contatto con un meteorite o navicella;
- **Game Over:** è quella condizione che si verifica quando la ‘Vita’ raggiunge lo zero;
- **Database:** archivio dati in cui sono memorizzate e gestite le informazioni;

4.1.2. Requisiti

Il primo passo da fare è individuare i requisiti del cliente. In pratica, una volta che si ha tutto il materiale in cui è descritto ciò che viene richiesto dal cliente, bisogna individuare precisamente quali sono gli obiettivi che il programma deve raggiungere, quali sono quelli più prioritari e quelli meno (utilizzando la notazione MoSCoW) dividendoli in due macro categorie: **requisiti funzionali** e **requisiti non funzionali**.

4.1.2.1. Requisiti funzionali

Sono dei requisiti che ci dicono cosa deve fare il sistema, in particolare descrivono le funzionalità in dettaglio che il sistema dovrebbe offrire.

Sostanzialmente sono dei requisiti dichiarati dall'utente che si possono vedere direttamente nel prodotto finale, a differenza dei requisiti non-funzionali.

I requisiti funzionali espressi come requisiti utente devono essere espressi in linguaggio naturale ed essere facilmente comprensibili.

Per il primo prototipo i requisiti funzionali individuati sono:

ID	Descrizione	Tipo	MoSCoW
RF-01	L'utente deve essere registrato per poter giocare	Funzionale	Must Have
RF-02	L'utente deve poter vedere le istruzioni su come giocare	Funzionale	Should Have
RF-03	L'utente deve poter mettere l'audio del gioco	Funzionale	Should Have
RF-04	L'utente deve poter effettuare il logout in qualsiasi momento	Funzionale	Must Have
RF-05	L'utente deve poter giocare spostandosi sia con le freccette sia con il mouse	Funzionale	Must Have
RF-06	L'utente deve poter visualizzare mentre gioca le vite rimaste e il punteggio accumulato evitando gli oggetti nemici	Funzionale	Should Have
RF-07	Quando il player viene colpito le vite dovranno diminuire	Funzionale	Must Have
RF-08	Il player quando verrà colpito deve rimanere invulnerabile per un periodo uguale a 1.5 secondi	Funzionale	Should Have
RF-09	Il gioco deve terminare non appena le vite rimaste si azzerano (Game Over)	Funzionale	Must Have
RF-10	L'utente deve poter scegliere se ricominciare una nuova partita immediatamente dopo aver perso	Funzionale	Should Have

4.1.2.2. Requisiti non funzionali

Sono dei requisiti che descrivono come il sistema si deve comportare, in particolare specificano delle proprietà e definiscono dei vincoli di qualità che il sistema deve soddisfare in base agli accordi di contratto presi nel progetto.

Sono spesso in numero inferiore rispetto ai requisiti funzionali ma sono più critici. I requisiti non funzionali derivano dalle necessità degli utenti, dai limiti del budget, dalle politiche organizzative e da fattori esterni.

Per il primo prototipo i requisiti non funzionali individuati sono:

ID	Descrizione	Tipo	MoSCoW
RNF-01	La navicella deve muoversi verso destra e sinistra in maniera fluida	Non Funzionale	Must Have
RNF-02	Il sistema non deve presentare bug e non deve bloccarsi (Affidabilità)	Non Funzionale	Must Have
RNF-03	L'applicazione deve risultare performante e rispondere il più velocemente possibile (Performance)	Non Funzionale	Should Have
RNF-04	Il sistema deve essere facilmente espandibile e adattabile alle future esigenze dell'applicazione (Flessibilità)	Non Funzionale	Must Have
RNF-05	Il sistema deve garantire la privacy delle informazioni trattate (Privacy)	Non Funzionale	Should Have
RNF-06	Il gioco deve risultare semplice e intuitivo	Non Funzionale	Should Have

- RF: Requisito Funzionale;
- RNF: Requisito Non Funzionale.

4.2. Design

4.2.1. UML

UML (*Unified Modeling Language*, "linguaggio di modellazione unificato"), è un linguaggio di modellazione e di specifica basato sul paradigma orientato agli oggetti. Definisce un insieme di diagrammi utilizzati per progettare, realizzare e documentare un sistema software complesso attraverso delle notazioni grafiche. Esso è indipendente dall'ambito del progetto, dal processo di sviluppo e dal linguaggio di programmazione.

Adesso andiamo a costruire i diagrammi dei casi d'uso, che appartengono ai diagrammi UML.

4.2.2. Use case diagram

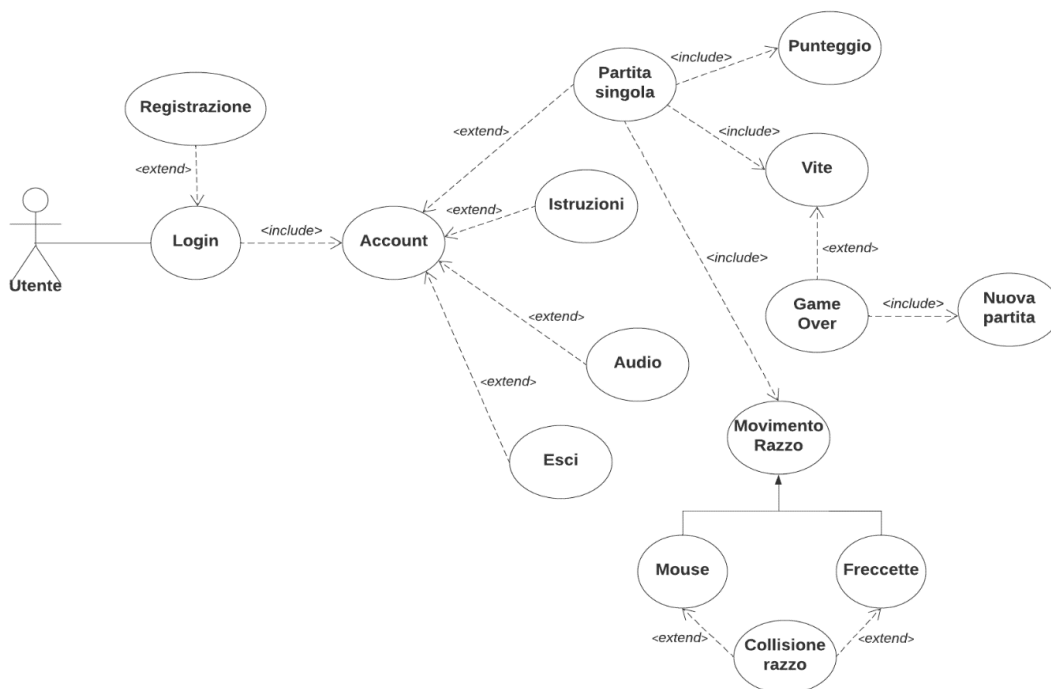
Lo **Use Case Diagram** è il primo diagramma (ad alto livello) di UML che si dovrebbe creare e che permette di definire i requisiti software dal punto di vista dell'utente.

Definisco cosa il sistema offre (servizi per l'utente finale, ...), ma non definisco come il sistema è realizzato (cioè non definiscono la parte d'implementazione, i dettagli interni del sistema, l'architettura, la topologia dell'applicazione).

L'obiettivo è di rendere più comprensibile il sistema, rappresentando il tutto attraverso dei modelli che permettono di semplificare il ragionamento e capire meglio chi fa cosa e le interazioni tra gli attori coinvolti.

Gli Use Case (casi d'uso) sono una tecnica basata su scenari nell'UML che identifica gli attori coinvolti in uno scenario e l'interazione che esiste tra le funzionalità ed essi.

4.2.2.1. Use case Utente – 1° prototipo



Caso d'uso: Login

Autore: Utente

Pre-condizione: Le credenziali di accesso, username e password, devono essere già presenti in archivio

Post-condizione: Se l'autenticazione ha avuto successo, l'utente accede alla finestra relativa al suo account e può usufruire di tutti i servizi forniti

Scenario principale:

- L'utente inserisce le credenziali
- L'utente accede alla schermata di gioco

Scenario alternativo:

- Se le credenziali di accesso inserite dall'utente non sono presenti in archivio, il sistema richiede all'utente di verificare la correttezza delle credenziali, altrimenti richiede all'utente di avviare la procedura di registrazione.

Caso d'uso: Registrazione

Autore: Utente

Pre-condizione: I parametri inseriti, username e password, non devono essere già presenti in archivio

Post-condizione: Se la registrazione ha avuto successo l'utente è stato inserito nell'archivio

Scenario principale:

- L'utente inserisce i dati richiesti per registrarsi
- L'utente viene registrato

Scenario alternativo:

- Se le credenziali inserite sono già presenti nell'archivio, il sistema richiede all'utente di inserire delle credenziali diverse

Caso d'uso: Account

Autore: Utente

Pre-condizione: Le credenziali inserite per effettuare l'accesso, username e password, devono essere corrette

Post-condizione: Se l'utente risulta loggato può usufruire di tutti i servizi forniti (Partita singola, Istruzioni, Audio, Esci)

Scenario principale:

- L'utente loggato potrà scegliere il servizio.

Scenario alternativo:

- L'utente, in qualunque momento, è libero di poter uscire dal gioco premendo il pulsante 'x' in alto a destra

Caso d'uso: Partita singola

Autore: Utente

Pre-condizione: L'utente si è loggato e accede alla schermata introduttiva

Post-condizione: L'utente giocatore può iniziare a giocare

Scenario principale:

- L'utente preme sul pulsante "Partita Singola"
- L'utente inizierà a giocare

Scenario alternativo:

- L'utente, in qualunque momento, è libero di poter uscire dal gioco premendo il pulsante 'x' in alto a destra

Caso d'uso: Istruzioni

Autore: Utente

Pre-condizione: L'utente si è loggato e accede alla schermata introduttiva

Post-condizione: L'utente vede i comandi del gioco e saprà come giocare

Scenario principale:

- L'utente preme sul pulsante "Istruzioni"
- L'utente vedrà le istruzioni per giocare

Scenario alternativo:

- L'utente, in qualunque momento, è libero di poter uscire dal pannello istruzioni premendo il pulsante 'x'

Caso d'uso: Audio

Autore: Utente

Pre-condizione: L'utente si è loggato e accede alla schermata introduttiva

Post-condizione: L'utente attiva la colonna sonora del gioco

Scenario principale:

- L'utente preme sul pulsante "Audio"
- L'utente attiverà l'audio per giocare

Caso d'uso: Esci

Autore: Utente

Pre-condizione: L'utente si è loggato e accede alla schermata introduttiva

Post-condizione: La schermata introduttiva viene chiusa e l'utente ritorna alla schermata di login

Scenario principale:

- L'utente preme sul pulsante "Esci"
- L'utente ritornerà alla schermata di login

Scenario principale:

- L'utente non uscirà dal gioco

Caso d'uso: Punteggio

Autore: Utente

Pre-condizione: L'utente si è loggato e accedendo alla schermata introduttiva avvia il gioco

Post-condizione: Il punteggio aumenta non appena il razzo schiva gli oggetti nemici

Scenario principale:

- L'utente avvia il gioco
- Il punteggio aumenterà quando il razzo schiverà i nemici

Scenario alternativo:

- L'utente perderà una vita nel caso in cui il razzo non riuscirà a schivare i nemici

Caso d'uso: Movimento Razzo

Autore: Utente

Pre-condizione: L'utente si è loggato e accedendo alla schermata introduttiva avvia il gioco

Post-condizione: Il razzo si muove in base al movimento del mouse o in base alla freccia direzionale che viene premuta

Scenario principale:

- L'utente avvia il gioco
- Il razzo si muoverà in base alla scelta di movimento che l'utente utilizzerà (mouse o freccette direzionali)

Scenario alternativo:

- Il razzo rimarrà fermo nel caso in cui non verrà mosso

Caso d'uso: Collisione Razzo

Autore: Utente

Pre-condizione: L'utente si è loggato e accedendo alla schermata introduttiva avvia il gioco

Post-condizione: Il razzo mentre si muove viene colpito da un oggetto nemico

Scenario principale:

- L'utente avvia il gioco
- Il razzo verrà colpito dal meteorite o dalla navicella nemica

Scenario alternativo:

- Il razzo non verrà colpito dal meteorite o dalla navicella nemica

Caso d'uso: Vite

Autore: Utente

Pre-condizione: L'utente si è loggato e accedendo alla schermata introduttiva avvia il gioco

Post-condizione: Le vite diminuiscono se il razzo viene colpito

Scenario principale:

- L'utente avvia il gioco
- Le vite diminuiranno se il razzo verrà colpito

Scenario alternativo:

- Le vite rimarranno invariate nel caso in cui il razzo non verrà colpito

Caso d'uso: Game Over

Autore: Utente

Pre-condizione: L'utente si è loggato e accedendo alla schermata introduttiva avvia il gioco

Post-condizione: L'utente ha esaurito le vite e di conseguenza ha perso

Scenario principale:

- L'utente avvia il gioco
- Il razzo viene ripetutamente colpito dagli oggetti nemici fino ad azzerare le vite
- L'utente perde

Scenario alternativo:

- L'utente continua a giocare non avendo perso tutte le vite

Caso d'uso: Nuova Partita

Autore: Utente

Pre-condizione: L'utente ha perso la partita

Post-condizione: L'utente dopo aver perso la partita può decidere se riiniziare una nuova partita

Scenario principale:

- L'utente conclude la partita
- L'utente deciderà di ricominciare una nuova partita

Scenario alternativo:

- L'utente deciderà di non ricominciare una nuova partita

4.2.3. Activity Diagram

L'**Activity Diagram** è un diagramma UML che è molto simile ai diagrammi di flusso. Esso mostra una serie di azioni (cioè passi), chiamate attività che vengono eseguite nel flusso di un programma e che vengono applicate per modellare il diagramma comportamentale. Con il termine attività ci riferiamo ad una specifica attività che deve essere svolta all'interno della funzione. Essa è rappresentata da un rettangolo smussato con una descrizione dell'attività. Il flusso è rappresentato tramite delle frecce orientate, che indicano la sequenza temporale con cui devono essere effettuate le diverse attività. È previsto un simbolo per indicare l'inizio del flusso ed un altro per indicarne il termine. Le attività possono essere eseguite in parallelo, in questo caso il punto di divisione (fork) è rappresentato da frecce divergenti rispetto ad un segmento. Nel caso le attività siano alternative, cioè svolte o meno rispetto ad una scelta, il punto di decisione è rappresentato da dei rombi da cui partono i flussi alternativi. Il punto di ricongiungimento (join) è rappresentato tramite un segmento su cui le frecce si ricongiungono. Il modello delle attività, come dice lo stesso nome, modella un'attività relativa ad un qualsiasi elemento di modellazione, ad esempio: classi, casi d'uso, interfacce, componenti e operazioni di classe.

I componenti che vengono utilizzati nell'Activity Diagram sono:



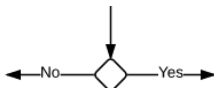
Stato iniziale



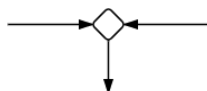
Stato di attività o d'azione



Flusso d'azione



Nodo condizionale (branch)



Nodo decisionale (merge)



Nodo Fork



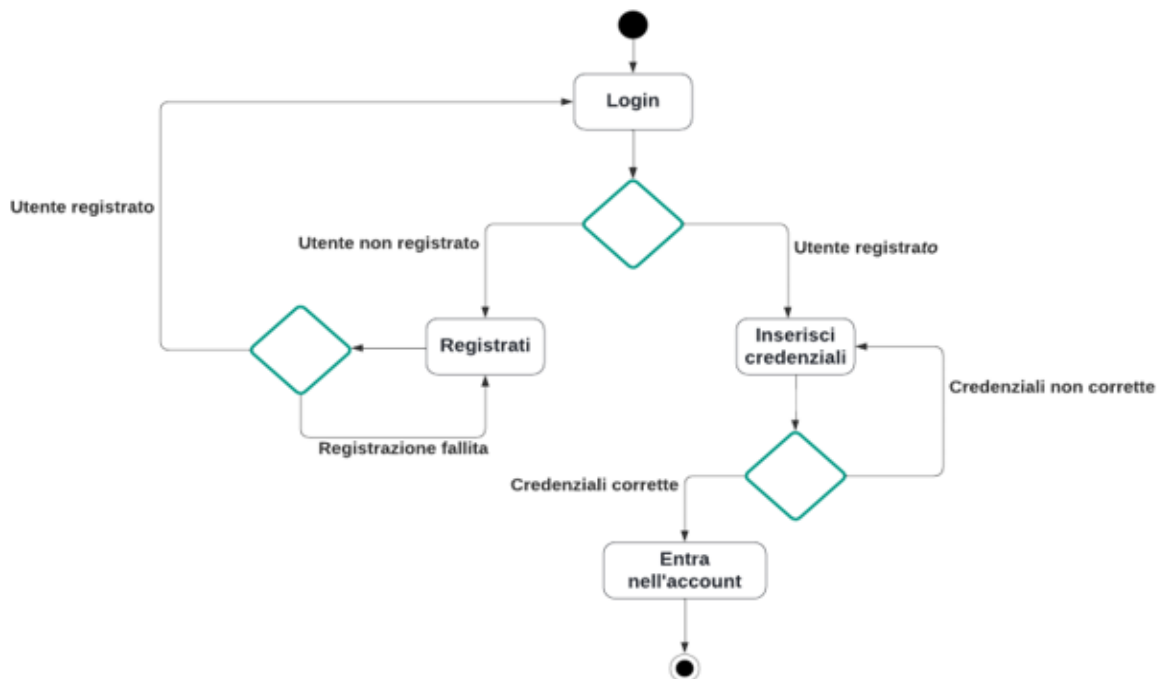
Nodo Join



Stato finale

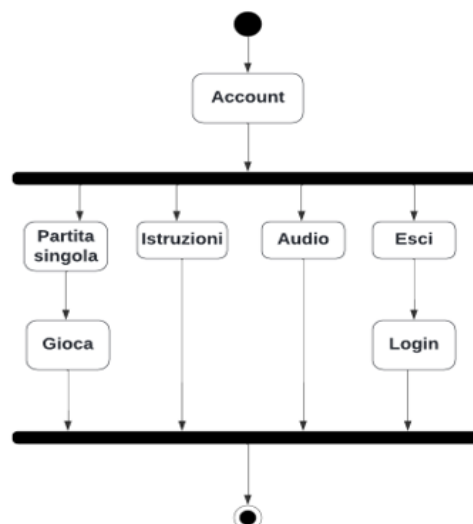
4.2.3.1. Activity Diagram Login/Registrazione Utente – 1° prototipo

Una volta avviato il gioco si verrà indirizzati nella pagina di login. A questo punto l'utente ha due possibilità: loggarsi con le proprie credenziali o registrarsi e creare un nuovo profilo. Nel primo caso se le credenziali inserite sono corrette, l'utente verrà indirizzato nell'account del gioco altrimenti, gli verrà richiesto di riprovare poiché risultano errate. Nel secondo caso, se i dati inseriti per la registrazione sono corretti, verrà creato il nuovo profilo, altrimenti verrà richiesto di modificare i dati poiché essi sono già presenti nel database oppure risultano non inseriti.



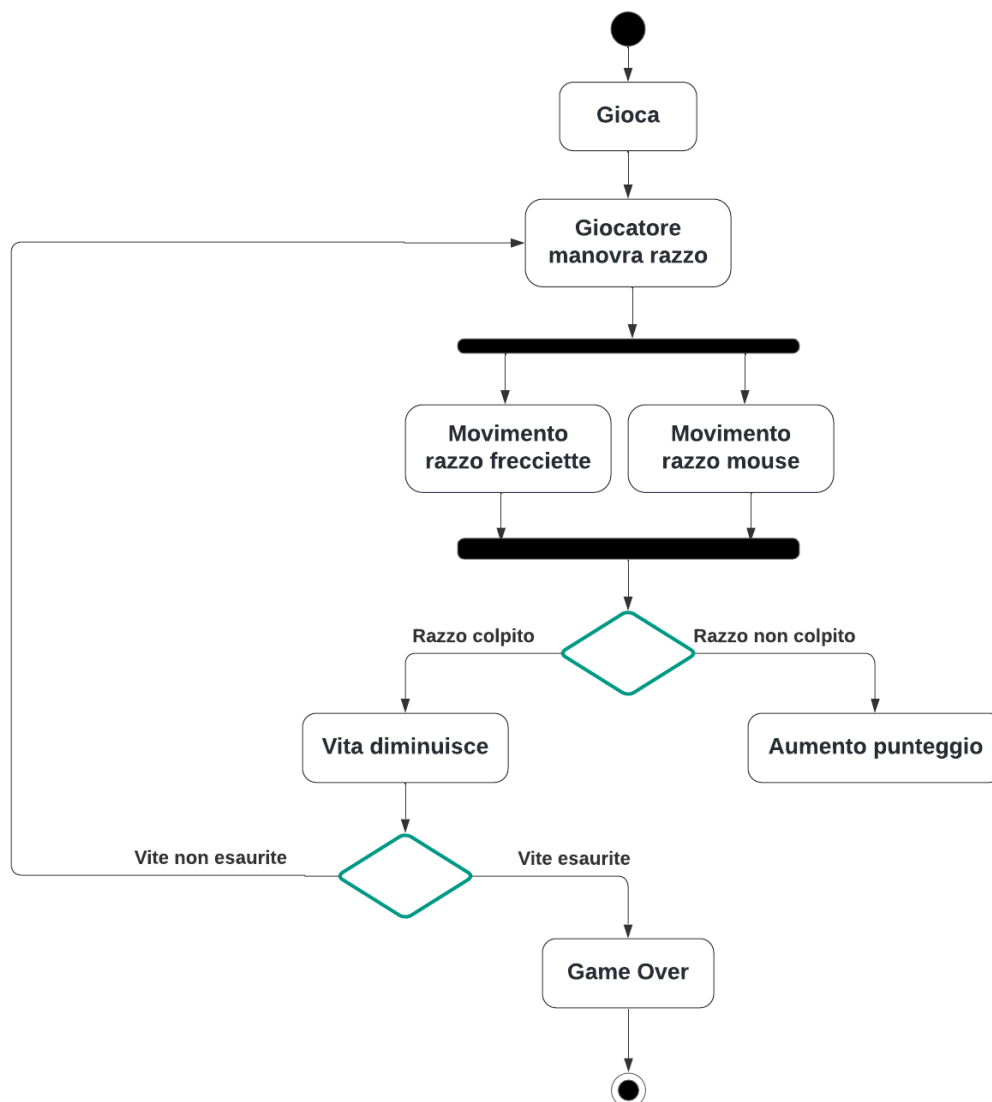
4.2.3.2. Activity Diagram Account Utente – 1° prototipo

Una volta loggatosi l'utente avrà la possibilità di scegliere se iniziare una nuova partita, leggere le istruzioni relative al gioco, attivare l'audio oppure effettuare il logout.



4.2.3.3. Activity Diagram Gioco Utente – 1° prototipo

L'utente cliccando sul bottone "PARTITA SINGOLA" potrà iniziare a giocare al gioco muovendo il razzo tramite il mouse o le frecce direzionali. Se il razzo viene colpito da un meteorite o da una navicella nemica che scendono dall'alto, la vita diminuirà in caso contrario aumenterà il punteggio che dipenderà dal tipo di nemico. Se il razzo verrà colpito ripetutamente dagli oggetti nemici, facendo giungere la vita ad un valore uguale a 0, allora il giocatore perderà la partita e si genererà lo scenario di Game Over, altrimenti l'utente continuerà a muovere il razzo durante la partita.



4.2.4. Sequence Diagram

Il **Sequence Diagram** è un diagramma utilizzato prevalentemente in fase di design che evidenzia l'evoluzione (flusso) temporale di elaborazione (comportamento) di uno specifico scenario, ovvero permette di modellare dinamicamente la comunicazione in un sistema software, evidenziando le interazioni dovute allo scambio di messaggi tra un oggetto ed un altro in relazione al trascorrere del tempo. Questi diagrammi sono costituiti da una serie di componenti che si dispongono su una dimensione verticale e orizzontale. Possiamo distinguere:

Gli oggetti che sono rappresentati da rettangoli disposti orizzontalmente aventi un nome, gli utenti invece vengono rappresentati da un omino.

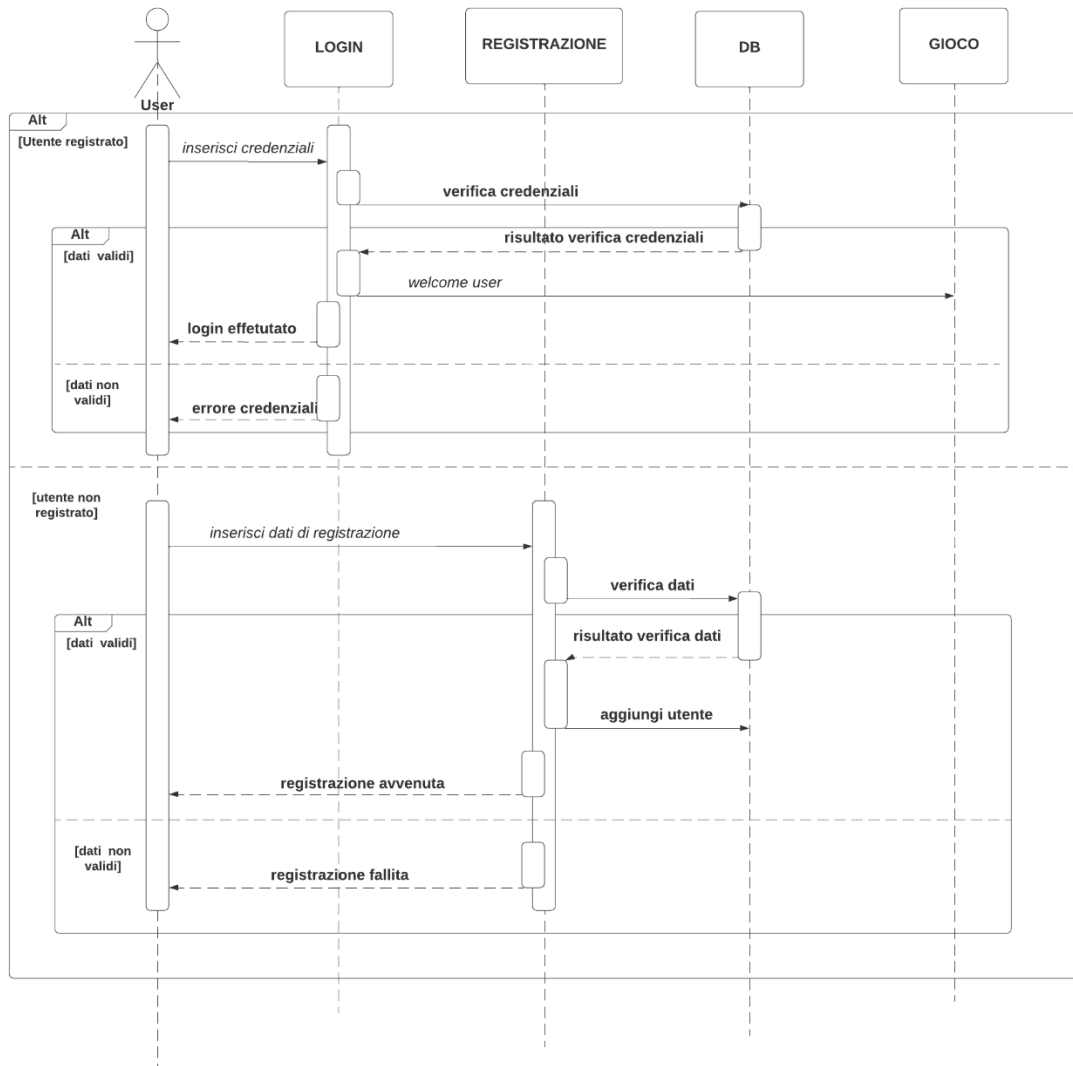
Da questi elementi, vengono poi fatte partire delle linee tratteggiate verticali dette "*lifeline*" (linea della vita) che mostrano l'ordine cronologico con cui vengono scambiati i vari messaggi. Lungo la lifeline si trova un piccolo rettangolo chiamato "*activation*" (attivazione) che specifica la durata dell'esecuzione di un'operazione di cui l'oggetto si fa carico. I messaggi possono essere: sincroni se l'emittente rimane in attesa di una risposta dal destinatario, asincrono in caso contrario, di ritorno per indicare che il destinatario del messaggio ha terminato l'elaborazione e sta restituendo il controllo al chiamante e infine ricorsivi per indicare che un oggetto manda un messaggio a se stesso (il mittente e il ricevente sono la stessa persona).

I componenti che vengono utilizzati nel Sequence Diagram sono:



4.2.4.1. Sequence Diagram Login/Registrazione Utente – 1° prototipo

Nel Sequence Diagram seguente, si mostra il flusso rappresentato dall'Activity Diagram del paragrafo precedente riguardante il login e la registrazione. In particolare mostra come l'utente interagisce con la finestra relativa alla login e alla registrazione le quali a loro volta interagiscono con il database.



4.3. Implementazione

Adesso, dopo aver stabilito le basi progettazione del software, passiamo alla **fase implementativa** del progetto. Essa è la fase di realizzazione del software, che concretizza la soluzione software attraverso la programmazione, ovvero la stesura di programmi da parte di programmatori o sviluppatori. All'interno del primo prototipo troveremo quindi:

- Schermata Benvenuto
- Schermata Login
- Schermata Registrazione
- Schermata Account
- Schermata Gioco

L'intero progetto verrà sviluppato in Java, un linguaggio di programmazione ad alto livello orientato agli oggetti. Come ambiente di sviluppo è stato scelto NetBeans. Per gli elementi grafici sono stati utilizzati le librerie Swing e Awt. Per ogni elemento del gioco verrà creata una classe, che richiamerà, quando necessario, altre classi. Quando viene creata una classe, definiamo delle informazioni (attributi) e delle azioni (metodi) che questa deve svolgere.

Di seguito vengono mostrate le implementazioni delle classi riguardanti il 1° prototipo.

4.3.1. Classe FinestraBenvenuto

La **classe FinestraBenvenuto** è la classe che definisce il frame iniziale relativo alla schermata di benvenuto al gioco.

All'interno di questa classe:

- creiamo la finestra di “Benvenuto nel gioco!” definendo le dimensioni del frame, il titolo, il colore di sfondo e l'immaginetta che apparirà come logo accanto al nome del gioco,
- definiamo il Layout per il posizionamento delle varie label all'interno della finestra,
- creiamo le etichette relative alla finestra di benvenuto e le aggiungiamo nel ContentPane del frame FinestraBenvenuto,
- infine, applichiamo il metodo sleep della classe Thread per far scomparire dopo un tempo di 9 secondi la finestra iniziale di benvenuto e far comparire quella relativa al Login.

4.3.2. Classe FinestraConLogin

La **classe FinestraConLogin** è la classe che definisce il frame relativo alla login di un utente. All'interno di questa classe:

- creiamo la finestra di “Login” definendo le dimensioni del frame, il titolo, il colore di sfondo e l'immaginetta che apparirà come logo accanto al nome del gioco,
- definiamo il Layout per il posizionamento delle varie componenti all'interno della finestra,
- definito una label “CLICCA QUI” a cui associamo un evento che richiama la classe MouseEvent nel quale ridefiniamo mediante Override il metodo MouseClicked che permetterà all'utente (cliccando sulla label) di passare alla finestra di registrazione,

- creiamo le label, le Text Field per inserire le credenziali, il bottone “CHECK” per rendere la password visibile e il bottone “LOGIN” che permetterà all’utente di loggarsi.
- implementiamo l’interfaccia ActionListener, ovvero l’ascoltatore degli eventi legato ai bottoni che implementa un metodo actionPerformed che permette di associare al rispettivo bottone il comportamento ad esso attribuitogli,
- aggiungiamo l’ascoltatore degli eventi relativo ai 2 bottoni.
- infine, effettueremo la connessione con il database per verificare che le credenziali inserite sono presenti all’interno del database.

4.3.3. Classe FinestraConRegistrazione

La **classe FinestraConRegistrazione** è la classe che definisce il frame relativo alla creazione di un account di un utente.

All’interno di questa classe:

- creiamo la finestra di “Registrazione” definendo le dimensioni del frame, il titolo, il colore di sfondo e l’immaginetta che apparirà come logo accanto al nome del gioco,
- definiamo il Layout per il posizionamento delle varie componenti all’interno della finestra,
- definito una label “TORNA ALLA LOGIN” a cui associamo un evento che richiama la classe MouseEvent nel quale ridefiniamo mediante Override il metodo MouseClicked che permetterà all’utente di passare (dopo aver fatto la registrazione) alla finestra di login,
- creiamo le label, le Text Field, per inserire i dati personali di un utente, il bottone “REGISTRATI” che permetterà all’utente di registrarsi al gioco e il bottone “ANCELLA” per eliminare i dati inseriti e poterli riscrivere.
- implementiamo l’interfaccia ActionListener, ovvero l’ascoltatore degli eventi legato ai bottoni che implementa un metodo actionPerformed che permette di associare al rispettivo bottone il comportamento ad esso attribuitogli,
- aggiungiamo l’ascoltatore degli eventi relativo ai 2 bottoni.
- infine, effettueremo la connessione con il database per verificare la correttezza dei dati inseriti e per memorizzarli all’interno del database.

4.3.4. Classe FinestraBottoni

La **classe FinestraBottoni** è la classe che definisce il frame relativo alla schermata dell’account dell’utente contenente i bottoni che permettono all’utente di svolgere un determinato evento.

All’interno di questa classe:

- creiamo la finestra contenente i 4 bottoni,
- definiamo le dimensioni del frame, il titolo, il colore di sfondo e l’immaginetta che apparirà come logo accanto al nome del gioco,
- creiamo i bottoni “PARTITA SINGOLA” – “ISTRUZIONI” – “AUDIO OFF” – “ESCI”
- definiamo il Layout per il posizionamento delle varie componenti all’interno della finestra,
- implementiamo l’interfaccia ActionListener, ovvero l’ascoltatore degli eventi

legato ai bottoni che implementa un metodo actionPerformed che permette di associare al rispettivo bottone il comportamento ad esso attribuitogli,

- aggiungiamo l'ascoltatore degli eventi relativo ai 4 bottoni.

4.3.5. Classe Spazio

La **classe Spazio** è la classe perno dove vengono implementate le varie dinamiche del gioco.

All'interno di questa classe:

- inizialmente impostiamo il colore di sfondo del gioco, passiamo la sua dimensione e definiamo la posizione del razzo all'interno del pannello,
- aggiungiamo l'interfaccia pubblica MouseMotionListener che sovrascrive il metodo mouseMoved che permette la ricezione degli eventi di movimento del mouse su un componente (razzo),
- aggiungiamo l'interfaccia pubblica KeyListener che sovrascrive il metodo keyPressed che permette la ricezione degli eventi legati al tasto che viene premuto sulla tastiera (in particolare per permettere al giocatore di spostarsi con le frecce left e right),
- metodo paint (@Override): metodo che permette di disegnare i vari oggetti: razzo, navicella nemica, meteoriti (gli oggetti nemici verranno cancellati quando usciranno dal riquadro di gioco), di inserire le varie scritte relative al gioco e definire il colore, la posizione e il formato,
- metodo nemici: metodo che permette di aggiungere in modo random sia i meteoriti che la navicella nemica,
- metodo collisione e metodo collisioneRazzo: metodi che permettono di gestire la collisione dei vari oggetti nemici con il razzo,
- metodo colpito: metodo che permette di decrementare le vite quando si verrà colpiti, di far comparire il pannello di Fine partita e di gestire il tempo di invulnerabilità,
- metodo play: metodo che permette di avviare il gioco e che andrà a gestire le varie dinamiche del gioco,
- metodo finePartita: metodo che mostrerà (appena si perde la partita) un pannello in cui l'utente potrà decidere se iniziare nuovamente la partita oppure no.

4.3.6. Classe RiproduciSuono

La **classe RiproduciSuono** è la classe che permette l'inserimento della musica di gioco quando l'utente preme sul bottone "AUDIO OFF".

All'interno di questa classe:

- passiamo il nome del file.

Nel metodo Run:

- definiamo un oggetto che rappresenterà il file,
- creiamo un oggetto del flusso audio che punta a quel file,
- definiamo un oggetto format di tipo AudioFormat che costruisce un flusso di input audio con il formato inserito,
- in seguito con SourceDataLine effettuiamo una scrittura continua del file audio nel buffer durante tutto il processo di riproduzione,
- infine andremo a leggere ripetutamente i blocchi specificati del flusso di ingresso

audio creato (questo viene ripetuto fino alla fine del flusso audio) e dopo che la lettura e il buffer sono stati completati, le risorse verranno liberate chiudendo la riga.

4.3.7. Classe Oggetto

La **classe Oggetto** è una classe astratta che descrive un generico tipo di oggetto dove andremo a definire delle caratteristiche comuni tra delle classi.

Le sottoclassi che ereditano dalla classe astratta Oggetto sono:

- La **classe Razzo** → In questa classe andremo a inserire l'immagine del Razzo e definire il suo movimento.
- La **classe NavicellaNemica** → In questa classe andremo a inserire l'immagine, definire il punteggio quando il player schiva l'oggetto nemico e definire la velocità di caduta delle navicelle.
- La **classe AsteroideNemico** → In questa classe andremo a inserire le immagini degli asteroidi, definire il punteggio in base al tipo di meteorite quando il player schiva l'oggetto nemico e definire la velocità di caduta dei meteoriti.

4.4. Testing

Una volta ultimata la fase implementativa del 1° prototipo siamo passati alla **fase del test** in cui si collauda il software utilizzando un approccio "black-box": ciò significa che il software verrà valutato "a scatola chiusa", senza verificare la validità del codice, ma controllando quale sia il comportamento del programma sviluppato, in modo da verificare come si comporta in determinate condizioni e verificare che rispetti i requisiti funzionali che erano stati specificati nella fase di "*Analisi dei requisiti*" relativa a questa iterazione.

- L'obiettivo della fase di verifica è scoprire situazioni in cui il comportamento del software è errato, indesiderato, o non conforme alle sue specifiche.
- L'obiettivo della fase di validazione è dimostrare che il software soddisfi le aspettative del cliente, cioè i suoi requisiti stabiliti secondo un contratto.

Il testing del software è quindi imprescindibile per garantirne la qualità, permettendo di creare e applicare standard e metodi per migliorare il processo di sviluppo e prevenire il verificarsi di bug. Inoltre, d'altra parte, è indispensabile per garantire all'utente una user experience soddisfacente, quest'ultimo aspetto è un fattore sempre più importante, infatti basti pensare che la maggior parte di utenti che non hanno avuto una buona esperienza con il servizio software (app) l'abbandonano e non vi fanno più ritorno.

Nelle tabelle di seguito sono riportati i test di verifica e validazione effettuati per il primo prototipo con il relativo esito.

4.4.1. Verifica

In questa fase si verificherà se il software progettato è privo di errori. Per questo motivo verranno effettuati dei test con lo scopo di cercare di trovare qualche errore, inserendo un'insiemi di dati in tutti i possibili input del programma.

Test	Descrizione	Esito
Test 1	Login con credenziali corrette	✓
Test 2	Login con credenziali errate	✗
Test 3	Login con credenziali mancanti	✗
Test 4	Registrazione con dati corretti	✓
Test 5	Registrazione con dati obbligatori mancanti	✗
Test 6	Registrazione con username già esistente	✗
Test 7	Registrazione con le due password inserite che non coincidono	✗

Come vediamo dalla tabella, gli aspetti riguardanti la login e la registrazione procedono come previsto e non si verifica nessun risultato inatteso. Pertanto nel caso in cui le credenziali inserite in fase di login sono corrette si verrà indirizzati alla finestra relativa all'account dell'utente, in caso contrario la login avrà esito negativo in quanto le credenziali risultano sbagliate o mancanti. Analogamente nel caso in cui i dati inseriti in fase di registrazione sono corretti, verrà creato il nuovo profilo relativo all'utente, in caso contrario la registrazione avrà esito negativo in quanto l'username è già esistente, i dati obbligatori inseriti sono mancanti o le due password inserite non coincidono.

4.4.2. Validazione

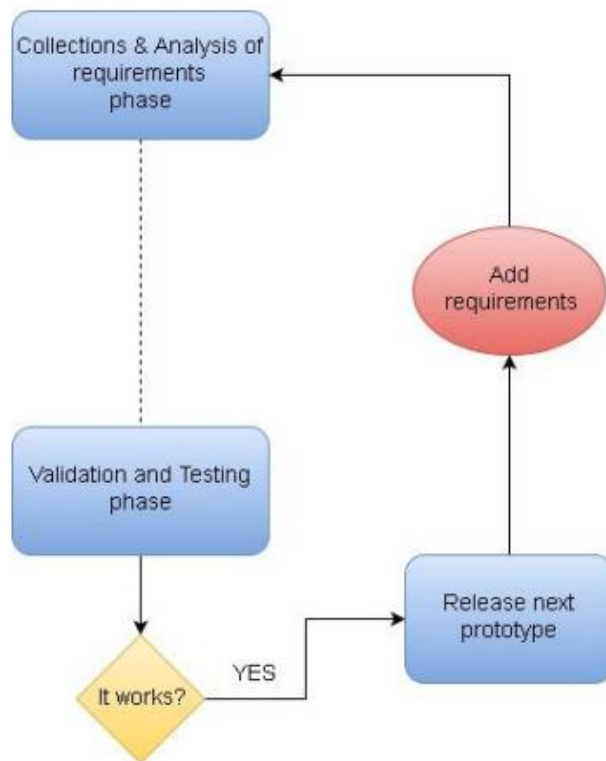
In questa fase si valuta che il software soddisfi ogni requisito funzionale definito in fase di analisi dei requisiti. Per questo motivo verranno effettuati dei test con lo scopo di verificare che tutti i requisiti stabiliti secondo il contratto sono stati implementati e risultano funzionanti.

ID	Descrizione	Esito
RF-01	L'utente deve essere registrato per poter giocare	✓
RF-02	L'utente deve poter vedere le istruzioni su come giocare	✓
RF-03	L'utente deve poter mettere l'audio del gioco	✗
RF-04	L'utente deve poter effettuare il logout in qualsiasi momento	✓
RF-05	L'utente deve poter giocare spostandosi sia con le freccette sia con il mouse	✓
RF-06	L'utente deve poter visualizzare mentre gioca le vite rimaste e il punteggio accumulato evitando gli oggetti nemici	✓
RF-07	Quando il player viene colpito le vite dovranno diminuire	✓
RF-08	Il player quando verrà colpito deve rimanere invulnerabile per un periodo uguale a 1.5 secondi	✗
RF-09	Il gioco deve terminare non appena le vite rimaste si azzerano (Game Over)	✓
RF-10	L'utente deve poter scegliere se ricominciare una nuova partita immediatamente dopo aver perso	✗

Come vediamo dalla tabella, l'esito dei requisiti RF-03, RF-08 e RF-10, catalogati come meno prioritari (tipologia Should Have) sono risultati negativi. Quindi, il team di sviluppo, provvederà ad incontrare nuovamente il cliente per risolvere i problemi riscontrati.

4.4.3. Valutazione 1° prototipo con feedback del client

A seguito di una prima valutazione sul prototipo software, esso risulta **user-friendly**, intuitivo e pronto per l'utilizzo seguendo i requisiti specificati per la prima iterazione. Pertanto una volta effettuata la fase di test, si è deciso di far visionare i risultati raggiunti al cliente che, dopo averli apprezzati, ha suggerito l'integrazione di ulteriori features per aggiungere profondità e completezza al progetto. Infatti, dopo un confronto con il cliente, si sono stabiliti dei nuovi obiettivi da raggiungere che verranno trattati nella versione del secondo prototipo.



Una volta prese in considerazione tutte le richieste del cliente, si andrà ad analizzare un secondo prototipo che rappresenta un miglioramento del primo, ma che comunque non rappresenterà la versione finale del progetto.

5. Secondo prototipo

Con il secondo prototipo il team ha implementato le nuove richieste del cliente qui descritte:

- L'implementazione della parte relativa all'admin del gioco che può visualizzare gli utenti registrati e eliminare un utente o un commento;
- L'implementazione del tasto pausa che permette di mettere il gioco in pausa,
- L'implementazione della funzionalità dello sparo dei missili contro gli oggetti nemici;
- L'implementazione dei livelli di difficoltà crescenti;
- La possibilità di poter visualizzare il migliore punteggio raggiunto dall'utente;
- La possibilità di poter aggiungere delle recensioni sul gioco.

5.1. Analisi dei requisiti

In questo paragrafo, una volta terminato la discussione con il client andremo ad analizzare i nuovi requisiti richiesti dal cliente.

5.1.1. Glossario

Definiamo i nuovi termini che verranno aggiunti nel glossario relativi al secondo prototipo:

- **Admin:** colui che gestisce il gioco;
- **Pausa:** bottone che permette di sospendere temporaneamente la partita;
- **Missile:** indica l'oggetto che viene sparato dal player;
- **Livello:** indica il livello raggiunto dal giocatore durante la partita;
- **Punteggio record:** indica il miglior punteggio raggiunto dal giocatore tra tutte le sue partite;
- **Aggiungi commento:** indica la possibilità che ha un utente di poter aggiungere una recensione sul gioco.

5.1.2. Requisiti

Adesso una volta definito il glossario dei termini che verranno utilizzati nel secondo prototipo, il primo passo è individuare quali sono i nuovi requisiti che il programma deve raggiungere, quali sono quelli più prioritari e quelli meno (utilizzando la notazione MoSCoW) dividendoli in due macro categorie: i requisiti funzionali e i requisiti non funzionali.

5.1.2.1. Requisiti funzionali

Per il secondo prototipo i requisiti funzionali individuati sono:

ID	Descrizione	Tipo	MoSCoW
RF-11	Devono essere attribuite all'admin delle credenziali per potersi connettere come amministratore del gioco	Funzionale	Must Have

RF-12	L'admin deve poter visualizzare le persone registrate al gioco	Funzionale	Should Have
RF-13	L'admin deve poter eliminare gli utenti inserendo il loro username	Funzionale	Must Have
RF-14	L'admin deve poter eliminare un commento che ritiene inappropriate	Funzionale	Must Have
RF-15	L'admin deve poter effettuare il logout in qualsiasi momento	Funzionale	Must Have
RF-16	L'admin deve poter avere la possibilità di cercare un utente inserendo l'username relativo a quell'utente	Funzionale	Should Have
RF-17	Il player deve poter mettere in pausa il gioco cliccando sul pulsante P	Funzionale	Should Have
RF-18	Il player deve poter riprendere la partita in un qualsiasi momento cliccando nuovamente sul pulsante P	Funzionale	Should Have
RF-19	Il player deve avere la possibilità di sparare dei missili in verticale usando la bara spaziatrice sia per difendersi che per accumulare dei punti	Funzionale	Must Have
RF-20	Quando una partita di un giocatore termina il punteggio ottenuto e il livello raggiunto deve essere memorizzato sul database	Funzionale	Should Have
RF-21	La velocità di caduta degli oggetti nemici deve aumentare all'aumentare del livello di difficoltà.	Funzionale	Must Have
RF-22	L'utente deve avere la possibilità di visualizzare il suo miglior punteggio ottenuto nel gioco.	Funzionale	Should Have
RF-23	L'utente deve avere la possibilità di poter aggiungere un commento sul gioco	Funzionale	Must Have

5.1.2.2. Requisiti non funzionali

Per il secondo prototipo i requisiti non funzionali individuati sono:

ID	Descrizione	Tipo	MoSCoW
RNF-01	Il missile deve muoversi verso l'alto in maniera fluida	Non Funzionale	Must Have
RNF-02	Il sistema non deve presentare bug e non deve bloccarsi (Affidabilità)	Non Funzionale	Must Have
RNF-03	L'applicazione deve risultare performante e rispondere il più velocemente possibile (Performance)	Non Funzionale	Should Have
RNF-04	Il sistema deve essere facilmente espandibile e adattabile alle future esigenze dell'applicazione (Flessibilità)	Non Funzionale	Must Have
RNF-05	Il sistema deve garantire la privacy delle informazioni trattate (Privacy)	Non Funzionale	Should Have
RNF-06	Il gioco deve risultare semplice e intuitivo	Non Funzionale	Should Have

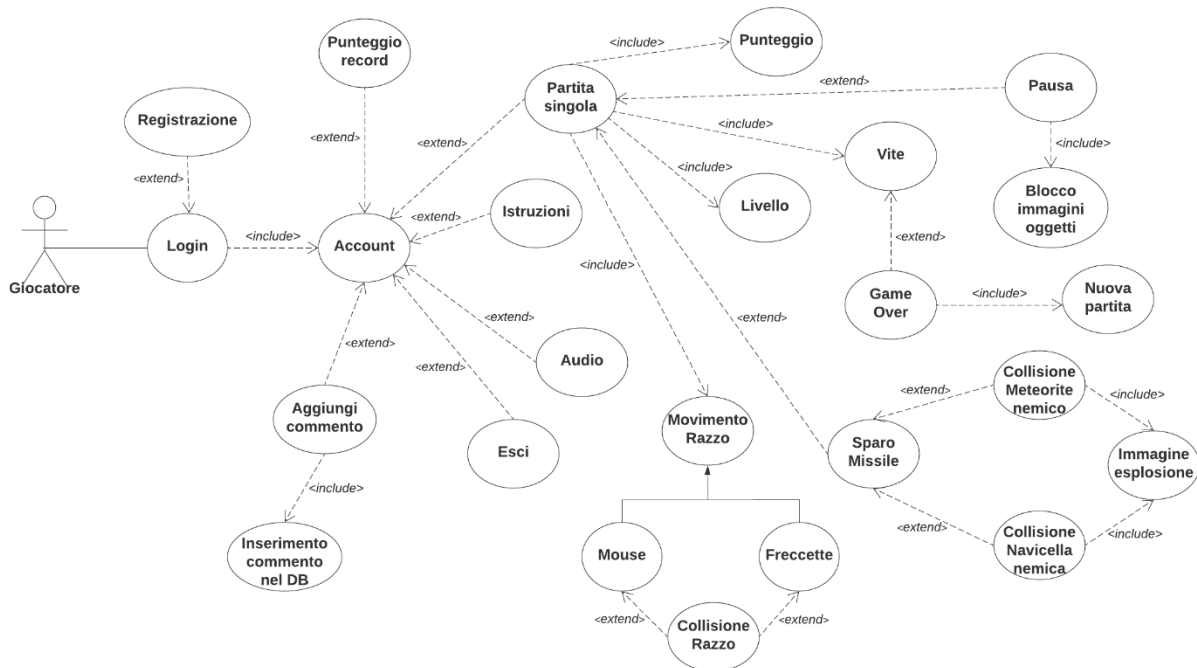
- RF: Requisito Funzionale;
- RNF: Requisito Non Funzionale.

5.2. Design

5.2.1. Use case diagram

Gli Use Case Diagram di questo secondo prototipo sono più completi di quello della prima versione del progetto. Le peculiarità che differenzia questa versione dalla precedente sono molteplici e riguardano sia l'inserimento delle funzionalità amministrative (come la possibilità di visualizzare e ricercare un utente, di eliminare un commento o un utente) e sia l'aggiunta di nuove funzionalità al gioco (come la possibilità di mettere il gioco in pausa, l'implementazione dei livelli di difficoltà, la possibilità di sparare i missili).

5.2.1.1. Use case Giocatore – 2° prototipo



Vediamo gli scenari mancanti che sono stati definiti in questo secondo prototipo:

Caso d'uso: Aggiungi commento

Autore: Giocatore

Pre-condizione: Il giocatore si è loggato e accede alla schermata introduttiva

Post-condizione: Il giocatore può scrivere un commento sul gioco

Scenario principale:

- Il giocatore preme sul pulsante “Aggiungi Commento”
- Il giocatore inserirà un commento relativo al gioco

Scenario alternativo:

- Il giocatore non inserirà un commento relativo al gioco

Caso d'uso: Inserimento commento nel DB

Autore: Giocatore

Pre-condizione: Il giocatore si è loggato e accede alla schermata introduttiva aggiunge il commento

Post-condizione: La recensione viene inserita nel database

Scenario principale:

- Il commento sarà inserito nella tabella “Commenti” insieme all’username del giocatore che lo ha scritto

Caso d'uso: Punteggio record

Autore: Giocatore

Pre-condizione: Il giocatore si è loggato e accede alla schermata introduttiva

Post-condizione: Il giocatore visualizza il suo miglior punteggio ottenuto

Scenario principale:

- Il giocatore visualizzerà il suo miglior punteggio ottenuto dopo essersi loggato

Caso d'uso: Pausa

Autore: Giocatore

Pre-condizione: Il giocatore si è loggato e accedendo alla schermata introduttiva avvia il gioco

Post-condizione: Il giocatore preme sul pulsante P per mettere il gioco in pausa

Scenario principale:

- Il giocatore sceglierà di premere sul pulsante P per mettere il gioco in pausa

Scenario alternativo:

- Il giocatore non sceglierà di premere sul pulsante P per mettere il gioco in pausa

Caso d'uso: Blocco immagini oggetti

Autore: Giocatore

Pre-condizione: Il giocatore si è loggato e accedendo alla schermata introduttiva avvia il gioco e lo mette in pausa

Post-condizione: Gli oggetti relativi al gioco si bloccano dopo che il giocatore preme sul pulsante P per mettere il gioco in pausa

Scenario principale:

- Gli oggetti relativi al gioco risulteranno bloccati sulla schermata di gioco

Scenario principale:

- Gli oggetti relativi al gioco si muoveranno in base alla loro funzione sulla schermata di gioco

Caso d'uso: Livello

Autore: Giocatore

Pre-condizione: Il giocatore si è loggato e accedendo alla schermata introduttiva avvia il gioco

Post-condizione: La velocità di caduta degli oggetti nemici aumenta all'aumentare del livello di difficoltà.

Scenario principale:

- La velocità di caduta degli oggetti nemici aumenterà all'aumentare del livello di difficoltà.

Scenario alternativo:

- La velocità di caduta degli oggetti nemici rimarrà invariata in quanto non si è ancora arrivati al livello successivo

Caso d'uso: Sparo Missile

Autore: Giocatore

Pre-condizione: Il giocatore si è loggato e accedendo alla schermata introduttiva avvia il gioco

Post-condizione: Il giocatore preme sulla bara spaziatrice e spara un missile

Scenario principale:

- Il giocatore premerà sulla bara spaziatrice e sparerà un missile

Scenario alternativo:

- Il giocatore non premerà sulla bara spaziatrice

Caso d'uso: Collisione Meteorite nemico

Autore: Giocatore

Pre-condizione: Il giocatore si è loggato e accedendo alla schermata introduttiva avvia il gioco spara un missile

Post-condizione: Il missile colpisce il meteorite e il giocatore acquisisce dei punti

Scenario principale:

- Il meteorite nemico verrà colpito

Scenario alternativo:

- Il meteorite nemico non verrà colpito

Caso d'uso: Collisione Navicella nemica

Autore: Giocatore

Pre-condizione: Il giocatore si è loggato e accedendo alla schermata introduttiva avvia il gioco spara un missile

Post-condizione: Il missile colpisce la navicella e il giocatore acquista dei punti

Scenario principale:

- La navicella nemica verrà colpita

Scenario alternativo:

- La navicella nemica non verrà colpita

Caso d'uso: Immagine esplosione

Autore: Giocatore

Pre-condizione: Il giocatore si è loggato e accedendo alla schermata introduttiva avvia il gioco spara un missile che colpisce un oggetto nemico

Post-condizione: Il missile colpisce l'oggetto nemico e si genera l'esplosione

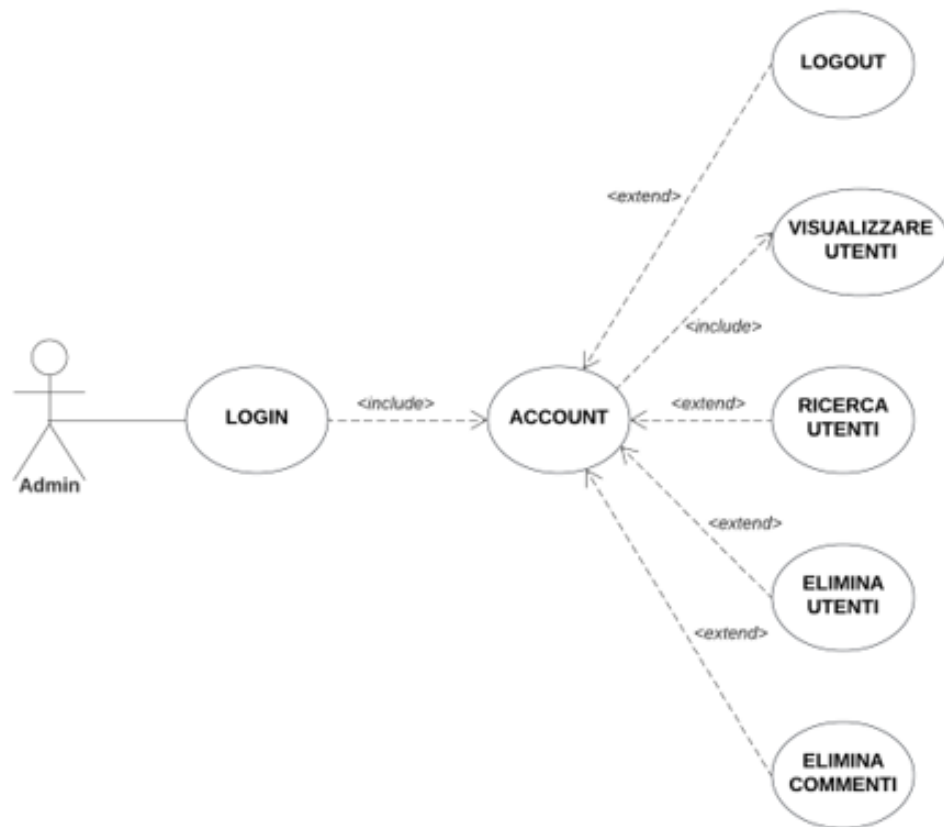
Scenario principale:

- L'oggetto nemico verrà colpito e si genererà l'esplosione

Scenario alternativo:

- L'oggetto nemico non verrà colpito

5.2.1.2. Use case Admin – 2° prototipo



Caso d'uso: Login

Autore: Admin

Pre-condizione: Le credenziali di accesso, username e password, devono essere già presenti in archivio

Post-condizione: Se l'autenticazione ha avuto successo, l'admin accede alla finestra relativa al suo account e può eseguire tutte le attività di sua competenza

Scenario principale:

- L'admin inserisce le credenziali
- L'admin accede al suo account

Scenario alternativo:

- Se le credenziali di accesso inserite dall'admin non sono presenti in archivio, il sistema richiede all'admin di verificare la correttezza delle credenziali

Caso d'uso: Account

Autore: Admin

Pre-condizione: Le credenziali inserite per effettuare l'accesso, username e password, devono essere corrette

Post-condizione: L'admin loggato può eseguire tutte le attività di sua competenza (Ricerca utente, elimina utente, elimina commento)

Scenario principale:

- L'utente loggato potrà scegliere l'attività da eseguire.

Scenario alternativo:

- L'utente, in qualunque momento, è libero di poter uscire dal gioco premendo il pulsante 'x' in alto a destra

Caso d'uso: Logout

Autore: Admin

Pre-condizione: L'admin si è loggato e accede al proprio account

Post-condizione: La schermata viene chiusa e l'admin ritorna alla schermata di login

Scenario principale:

- L'admin preme sul pulsante "Logout"
- L'admin ritornerà alla schermata di login

Scenario alternativo:

- L'admin non uscirà dal gioco

Caso d'uso: Visualizza utenti

Autore: Admin

Pre-condizione: L'admin si è loggato e accede al proprio account

Post-condizione: L'admin visualizza gli utenti registrati al gioco

Scenario principale:

- L'admin visualizzerà gli utenti registrati al gioco dopo essersi loggato

Caso d'uso: Ricerca utente

Autore: Admin

Pre-condizione: L'admin si è loggato e accede al proprio account

Post-condizione: L'admin ricerca un utente

Scenario principale:

- L'admin inserisce l'username dell'utente che vuole ricercare
- L'admin preme sul pulsante "Cerca"
- L'utente viene visualizzato nella tabella

Scenario alternativo:

- L'admin non ricercherà nessun utente del gioco

Caso d'uso: Elimina utente

Autore: Admin

Pre-condizione: L'admin si è loggato e accede al proprio account

Post-condizione: L'admin elimina l'account dell'utente

Scenario principale:

- L'admin inserisce l'username dell'utente che vuole eliminare
- L'admin preme sul pulsante "Elimina Utente"
- L'utente viene eliminato

Scenario alternativo:

- L'admin non eliminerà nessun utente dal gioco

Caso d'uso: Elimina commento

Autore: Admin

Pre-condizione: L'admin si è loggato e accede al proprio account

Post-condizione: L'admin elimina il commento dell'utente

Scenario principale:

- L'admin inserisce l'username dell'utente che ha inserito il commento
- L'admin preme sul pulsante "Elimina Utente"
- L'utente viene eliminato

Scenario alternativo:

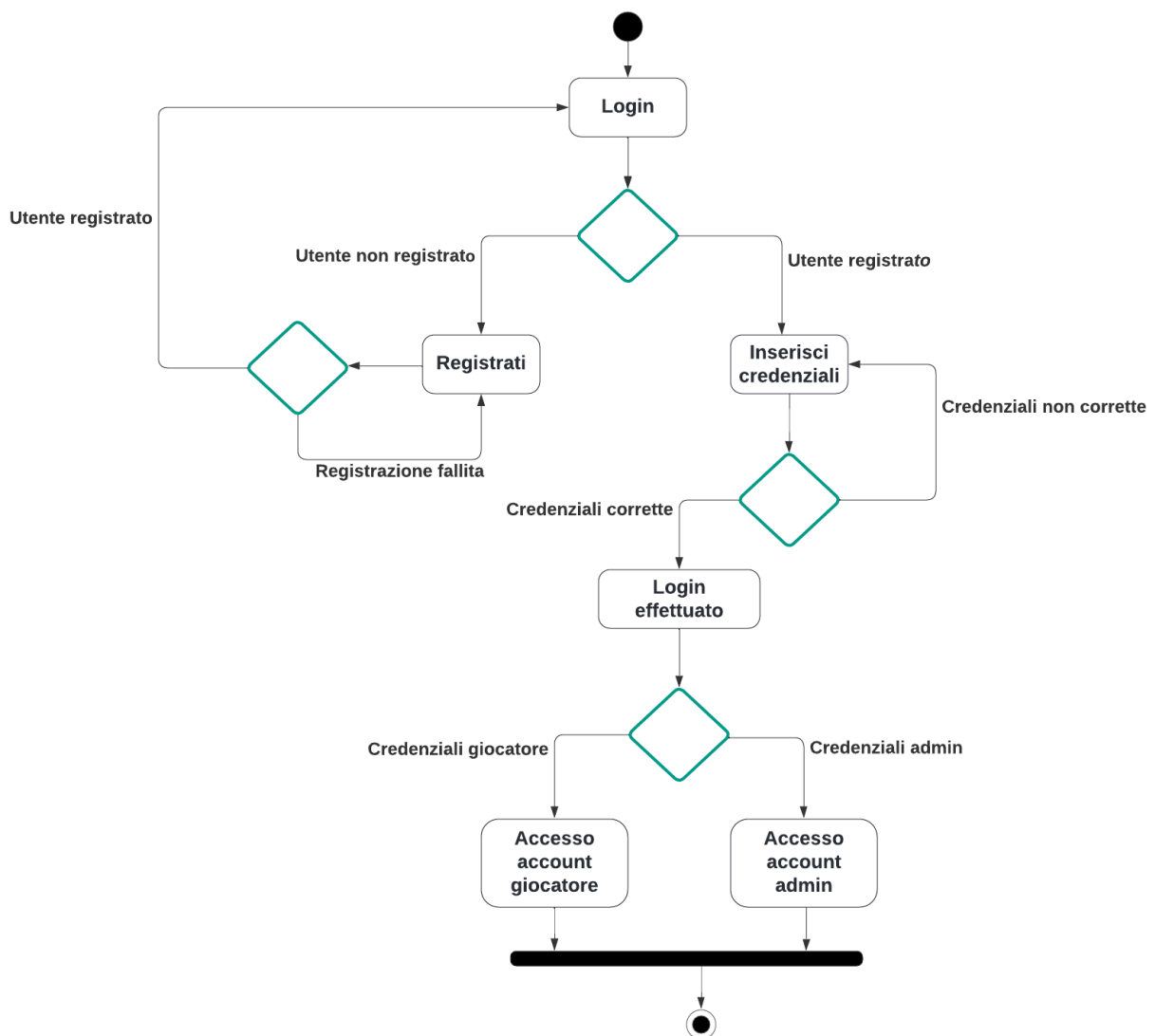
- L'admin non eliminerà nessun commento relativo al gioco

5.2.2. Activity Diagram

In questo secondo prototipo gli activity diagram definiti nel primo prototipo sono stati modificati aggiungendo le nuove funzionalità descritte negli use case del secondo prototipo.

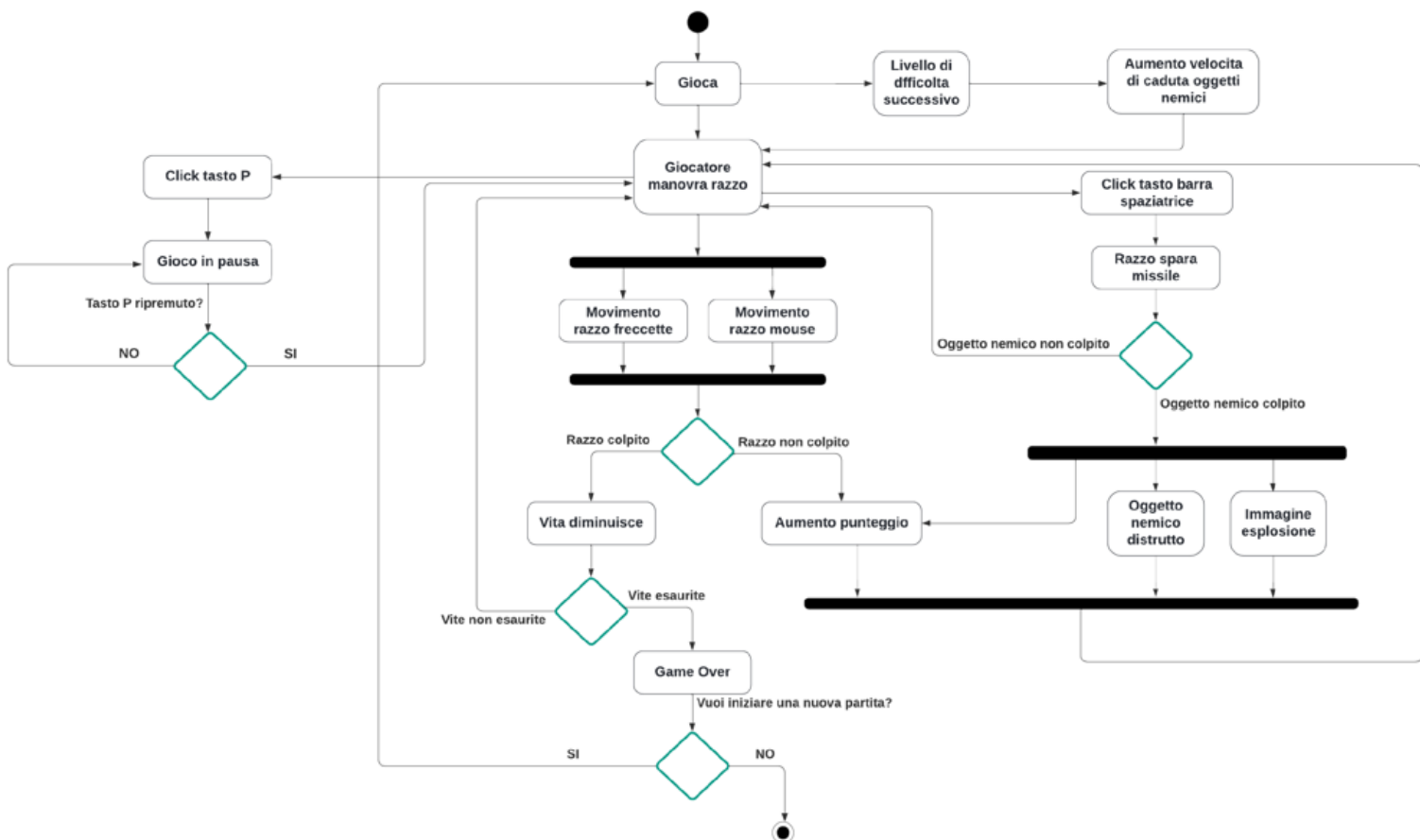
5.2.2.1. Activity Diagram Login/Registrazione – 2° prototipo

Una volta avviato il gioco si verrà indirizzati nella pagina di login. A questo punto si hanno due possibilità: loggarsi con le credenziali del giocatore (o registrarsi e creare un nuovo profilo) oppure loggarsi con le credenziali dell'admin. Nel primo caso avverrà ciò che abbiamo descritto nell'activity diagram "Login/Registrazione" del primo prototipo. Nel secondo caso se le credenziali inserite dall'amministratore sono corrette, esso verrà indirizzato nel proprio account altrimenti, gli verrà richiesto di riprovare poiché risultano errate.



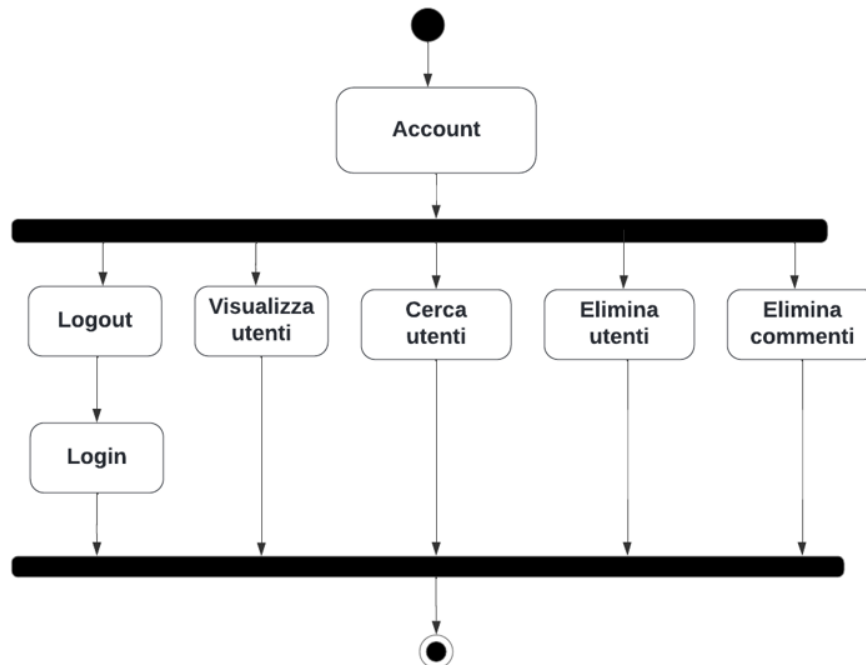
5.2.2.2. Activity Diagram Gioco – 2° prototipo

In questo activity diagram sono state aggiunte nuove funzionalità tra le quali: il giocatore ha la possibilità di giocare la partita con un livello di difficoltà crescente, infatti man mano che il giocatore passerà al livello successivo aumenterà la velocità di caduta degli oggetti nemici. Il giocatore inoltre avrà la possibilità di decidere se mettere il gioco in pausa premendo il tasto P e di riprendere a giocare ripremendo lo stesso tasto. In più avrà la possibilità di sparare dei missili per distruggere gli oggetti nemici. Se l'oggetto nemico verrà colpito, il nemico verrà distrutto, apparirà l'immagine dell'esplosione e il giocatore accumulerà dei punti.



5.2.2.3. Activity Diagram Account Admin – 2° prototipo

Una volta loggatosi l'admin avrà la possibilità di scegliere l'attività da svolgere cioè potrà visualizzare gli utenti registrati al gioco, cercare un utente specifico, eliminare l'account di un utente o un suo commento oppure disconnettersi dal proprio account.

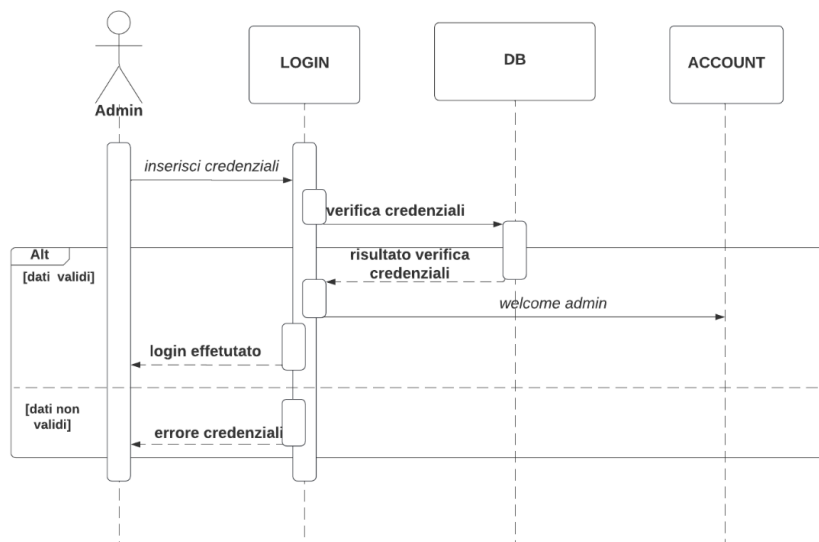


5.2.3. Sequence Diagram

In questo secondo prototipo abbiamo creato un sequence diagram riguardate l'admin.

5.2.3.1. Sequence Diagram Login Admin – 2° prototipo

Nel Sequence Diagram seguente, si mostra il flusso rappresentato dall'Activity Diagram del paragrafo precedente riguardante il login dell'admin. In particolare mostra come l'admin interagisce con la finestra relativa alla login la quale a sua volta interagisce con il database.



5.3. Implementazione

Nel secondo prototipo abbiamo implementato la schermata relativa all'admin e abbiamo aggiunto le implementazioni che il cliente ha richiesto e che sono state descritte nella fase di analisi dei requisiti.

Di seguito vengono mostrate le modifiche effettuate nelle rispettive classi e l'implementazione della classe riguardante la finestra admin del 2° prototipo.

5.3.1. Classe FinestraConLogin

Nella **classe FinestraConLogin** abbiamo aggiunto la parte del codice che permette all'admin di potersi connettere utilizzando come credenziali username:

“AdminGioco” e come password: “admin”

5.3.2. Classe FinestraAdminHome

La **classe FinestraAdminHome** è la classe che definisce il frame relativo alla schermata dell'account dell'admin contenente le funzionalità che permettono all'admin di svolgere delle attività.

All'interno di questa classe:

- definiamo le dimensioni del frame, il titolo, il colore di sfondo e l'immaginetta che apparirà come logo accanto al nome del gioco,
- definiamo il Layout per il posizionamento delle varie componenti all'interno della finestra,
- creiamo i bottoni “LOGOUT” – “CERCA” – “ELIMINA UTENTE” – “ELIMINA COMMENTO” e i text field che permettono all'admin di ricercare, eliminare gli utenti i commenti e di effettuare il logout,
- associamo alla tabella un evento che richiama la classe MouseEvent nel quale ridefiniamo mediante Override il metodo MouseClicked che permetterà all'admin (cliccando sull'utente) di visualizzare il suo username e il suo commento (se presente),
- implementiamo l'interfaccia ActionListener, ovvero l'ascoltatore degli eventi legato ai bottoni che implementa un metodo actionPerformed che permette di associare al rispettivo bottone il comportamento ad esso attribuitogli,
- aggiungiamo l'ascoltatore degli eventi relativo ai 4 bottoni,
- infine effettuiamo la connessione con il database per effettuare le modifiche all'interno del database.

5.3.3. Classe Utenti

La **classe Utenti** è la classe che abbiamo utilizzato per definire i costruttori e i metodi getter per accedere ai dati relativi agli utenti.

5.3.4. Classe FinestraBottoni

Nella **classe FinestraBottoni** sono state aggiunte la funzionalità che permette di visualizzare il miglior punteggio raggiunto dall'utente e la funzionalità che permette di poter aggiungere una recensione sul gioco. Per far questo abbiamo:

- implementato il metodo che permette di prelevare dal database il miglior punteggio ottenuto dall'utente tra tutte le partite giocate,
- definito una text filed, un bottone "AGGIUNGI COMMENTO", un metodo per effettuare la connessione al database e inserire un commento sul gioco.

5.3.5. Classe Spazio

Nella **classe Spazio** sono state aggiunte la funzionalità del livello di difficoltà crescente, la funzionalità che permette all'utente di mettere il gioco in pausa, e la possibilità di sparare dei missili premento la barra spaziatrice. Per far questo abbiamo:

- aggiunto all'interfaccia pubblica KeyListener che sovrascrive il metodo keyPressed che permette la ricezione degli eventi legati al click sul tasto "PAUSA" e al click sul tasto della barra spaziatrice che permetterà di sparare il missile,
- disegnato nel metodo paint (@Override) il missile che verrà cancellato non appena uscirà dallo schermo di gioco e aggiunto la scritta relativa al livello raggiunto,
- definito un metodo collisioneProiettile che permette di gestire la collisione tra il missile e i vari oggetti nemici, eliminare l'oggetto nemico colpito, assegnare il punteggio corrispondente all'oggetto colpito e che permette di far apparire l'immagine dell'esplosione,
- aggiunto nel metodo play le dinamiche relative al pulsante p e al missile,
- aggiunto il metodo memorizzaPunti che permette di memorizzare nel database il punteggio ottenuto dal giocatore per quella partita.

5.3.6. Classe Oggetto

La **classe Oggetto** come abbiamo detto nel precedente capitolo è una classe astratta che descrive un generico tipo di oggetto dove andremo a definire delle caratteristiche comuni tra delle classi.

Alle tre sottoclassi che ereditano dalla classe astratta Oggetto abbiamo aggiunto la classe:

- La **classe Proiettile** → In questa classe andremo a inserire l'immagine del Missile definire il punteggio quando il missile colpisce gli oggetti nemici.

5.4. Testing

Una volta ultimata la fase implementativa del 2° prototipo siamo passati alla fase del test. In questo caso bisognerà valutare non soltanto la correttezza dei test relativi al secondo prototipo, ma anche verificare che le nuove funzionalità introdotte non hanno avuto effetti dannosi o hanno peggiorato le funzionalità del sistema.

Nelle tabelle di seguito sono riportati i nuovi test di verifica e validazione effettuati più i test ripetuti del primo prototipo con il relativo esito.

5.4.1. Verifica

In questa fase si verificherà se il software progettato è privo di errori. Per questo motivo verranno effettuati dei test con lo scopo di cercare di trovare qualche errore, inserendo un'insiemi di dati in tutti i possibili input del programma.

Test	Descrizione	Esito
Test 1	Login con credenziali giocatore corrette	✓
Test 2	Login con credenziali admin corrette	✓
Test 3	Login con credenziali errate	✗
Test 4	Login con credenziali mancanti	✗
Test 5	Registrazione con dati corretti	✓
Test 6	Registrazione con dati obbligatori mancanti	✗
Test 7	Registrazione con username già esistente	✗
Test 8	Registrazione con le due password inserite che non coincidono	✗


Come vediamo dalla tabella, gli aspetti riguardanti la login e la registrazione procedono come previsto e non si verifica nessun risultato inatteso. Pertanto nel caso in cui le credenziali inserite in fase di login dal giocatore (o dall'admin) sono corrette si verrà indirizzati alla finestra relativa all'account del giocatore (o dell'admin), in caso contrario la login avrà esito negativo in quanto le credenziali risultano sbagliate o mancanti. Analogamente nel caso in cui i dati inseriti in fase di registrazione sono corretti, verrà creato il nuovo profilo relativo all'utente, in caso contrario la registrazione avrà esito negativo in quanto l'username è già esistente, i dati obbligatori inseriti sono mancanti o le due password inserite non coincidono.

5.4.2. Validazione

In questa fase si valuta che il software soddisfi ogni requisito funzionale definito in fase di analisi dei requisiti. Per questo motivo verranno effettuati dei test con lo scopo di verificare che tutti i requisiti stabiliti secondo il contratto sono stati implementati e risultano funzionanti.

ID	Descrizione	Esito
RF-01	L'utente deve essere registrato per poter giocare	✓
RF-02	L'utente deve poter vedere le istruzioni su come giocare	✓
RF-03	L'utente deve poter mettere l'audio del gioco	✓
RF-04	L'utente deve poter effettuare il logout in qualsiasi momento	✓
RF-05	L'utente deve poter giocare spostandosi sia con le freccette sia con il mouse	✓
RF-06	L'utente deve poter visualizzare mentre gioca le vite rimaste e il punteggio accumulato evitando gli oggetti nemici	✓
RF-07	Quando il player viene colpito le vite dovranno diminuire	✓
RF-08	Il player quando verrà colpito deve rimanere invulnerabile per un periodo uguale a 1.5 secondi	✓
RF-09	Il gioco deve terminare non appena le vite rimaste si azzerano (Game Over)	✓
RF-10	L'utente deve poter scegliere se ricominciare una nuova partita immediatamente dopo aver perso	✓

RF-11	Devono essere attribuite all'admin delle credenziali per potersi connettere come amministratore del gioco	✓
RF-12	L'admin deve poter visualizzare le persone registrate al gioco	✓
RF-13	L'admin deve poter eliminare gli utenti inserendo il loro username	✓
RF-14	L'admin deve poter eliminare un commento che ritiene inappropriato	✓
RF-15	L'admin deve poter effettuare il logout in qualsiasi momento	✓
RF-16	L'admin deve poter avere la possibilità di cercare un utente inserendo l'username relativo a quell'utente	✗
RF-17	Il player deve poter mettere in pausa il gioco cliccando sul pulsante P	✓
RF-18	Il player deve poter riprendere la partita in un qualsiasi momento cliccando nuovamente sul pulsante P	✓
RF-19	Il player deve avere la possibilità di sparare dei missili in verticale usando la barra spaziatrice sia per difendersi che per accumulare dei punti	✓
RF-20	Quando una partita di un giocatore termina il punteggio ottenuto e il livello raggiunto deve essere memorizzato sul database	✓
RF-21	La velocità di caduta degli oggetti nemici deve aumentare all'aumentare del livello di difficoltà.	✓
RF-22	L'utente deve avere la possibilità di visualizzare il suo miglior punteggio ottenuto nel gioco.	✗

RF-23	L'utente deve avere la possibilità di poter aggiungere un commento sul gioco	
-------	--	---

Come vediamo dalla tabella, sono stati risolti i problemi emersi nel prototipo precedente, ovvero l'esito negativo per quanto riguarda i requisiti RF-03, RF-08 e RF-10.

Il team dopo aver risolto con successo il problema, ha riscontrato degli ulteriori esiti negativi per i requisiti funzionali RF-16 e RF-22 (entrambi catalogati come meno prioritari). Anche stavolta, quindi, il team, in accordo con il cliente, provvederà a risolvere al meglio i problemi emersi.

5.4.3. Valutazione 2° prototipo con feedback del client

Una volta effettuata la fase di test, anche per questo secondo prototipo si è deciso di far visionare i risultati raggiunti al cliente che, dopo averli apprezzati, ha suggerito l'integrazione di ulteriori features per aggiungere profondità e completezza al progetto.

Infatti, dopo un confronto con il cliente, si sono stabiliti dei nuovi obiettivi da raggiungere che verranno trattati nella versione del terzo prototipo.

6. Terzo prototipo

Con il terzo e ultimo prototipo il team ha implementato le nuove richieste del cliente qui descritte:

- Miglioramento della grafica del gioco;
- L'implementazione della parte relativa all'utente ospite il quale può visualizzare le immagini e le recensioni relative al gioco;
- L'implementazione della funzionalità che permette a un utente di recuperare la propria password;
- La possibilità di poter visualizzare la classifica dei 5 miglior punteggi;
- Infine la possibilità di far scegliere all'utente se giocare in modalità carriera o in modalità multiplayer (a quest'ultima modalità è possibile giocare solamente se il giocatore ha raggiunto almeno il livello 5 nella modalità carriera).

6.1. Analisi dei requisiti

In questo paragrafo, una volta terminato la discussione con il client andremo ad analizzare i nuovi requisiti richiesti dal cliente.

6.1.1. Glossario

Definiamo i nuovi termini che verranno aggiunti nel glossario relativi al terzo prototipo:

- **Utente ospite:** utente non registrato che può solamente visualizzare immagine e recensioni sul gioco per decidere se registrarsi;
- **Recupera password:** indica la possibilità che ha un utente di poter inserire una nuova password;
- **Classifica:** permette al giocatore di visualizzare i migliori 5 punteggi del gioco;
- **Carriera:** permette al giocatore di avviare una partita in modalità single-player e memorizzare i suoi risultati;
- **Multiplayer:** permette al giocatore di avviare una partita in modalità multiplayer (1 vs. 1);

6.1.2. Requisiti

Adesso una volta definito il glossario dei termini che verranno utilizzati nel terzo prototipo, il primo passo è individuare quali sono i nuovi requisiti che il programma deve raggiungere, quali sono quelli più prioritari e quelli meno (utilizzando la notazione MoSCoW) dividendoli in due macro categorie: i requisiti funzionali e i requisiti non funzionali.

6.1.2.1. Requisiti funzionali

Per il terzo prototipo i requisiti funzionali individuati sono:

ID	Descrizione	Tipo	MoSCoW
RF-24	L'utente ospite deve poter visualizzare le immagini relative al gioco	Funzionale	Should Have
RF-25	L'utente ospite deve poter visualizzare delle recensioni sul gioco	Funzionale	Must Have
RF-26	L'utente ospite una volta visualizzate le immagini e le recensioni deve poter scegliere se registrarsi al gioco	Funzionale	Must Have
RF-27	L'utente registrato deve poter inserire una nuova password dopo aver verificato che la risposta alla domanda sia corretta e che l'username corrisponde all'utente in questione	Funzionale	Should Have
RF-28	Il giocatore una volta loggato deve poter visualizzare i 5 migliori risultati ottenuti dagli utenti registrati al gioco	Funzionale	Must Have
RF-29	Il giocatore deve poter scegliere se giocare in modalità carriera o multiplayer	Funzionale	Must Have
RF-30	Deve essere effettuato un controllo per verificare che il giocatore ha raggiunto almeno il livello 5 nella modalità carriera per giocare una partita in modalità multiplayer	Funzionale	Should Have
RF-31	I due player della modalità multiplayer devono poter inserire i loro rispettivi nomi per giocare	Funzionale	Must Have
RF-32	Nella modalità multiplayer dovrà apparire il messaggio "HAI VINTO" o "HAI PERSO" a seconda del risultato ottenuto dai due player	Funzionale	Must Have
RF-33	Nella modalità multiplayer dovrà apparire il punteggio dei 2 player	Funzionale	Should Have

6.1.2.2. Requisiti non funzionali

Per il terzo prototipo i requisiti non funzionali individuati sono:

ID	Descrizione	Tipo	MoSCoW
RNF-01	È necessario effettuare un miglioramento della grafica	Non Funzionale	Must Have
RNF-02	Il sistema non deve presentare bug e non deve bloccarsi (Affidabilità)	Non Funzionale	Must Have
RNF-03	L'applicazione deve risultare performante e rispondere il più velocemente possibile (Performance)	Non Funzionale	Should Have
RNF-04	Il sistema deve essere facilmente espandibile e adattabile alle future esigenze dell'applicazione (Flessibilità)	Non Funzionale	Must Have
RNF-05	Il sistema deve garantire la privacy delle informazioni trattate (Privacy)	Non Funzionale	Should Have
RNF-06	Il gioco deve risultare semplice e intuitivo	Non Funzionale	Should Have

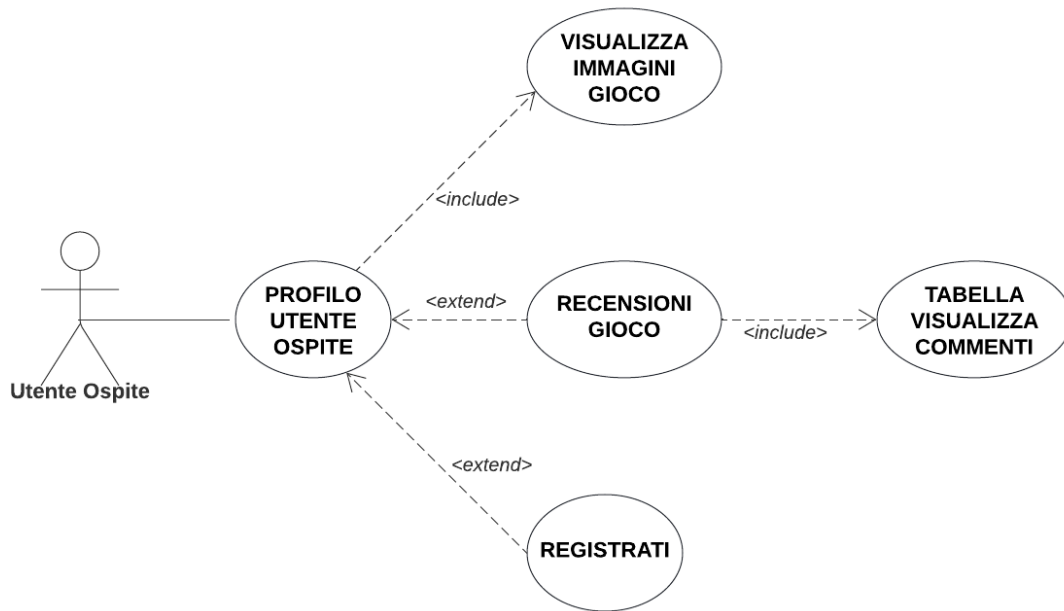
- RF: Requisito Funzionale;
- RNF: Requisito Non Funzionale.

6.2. Design

6.2.1. Use case diagram

Lo Use Case Diagram di questo terzo prototipo è più completo di quello della seconda versione del progetto. Le peculiarità che differenzia questa versione dalla precedente sono molteplici e riguardano sia l'inserimento delle funzionalità relative all'utente ospite (come la possibilità di visualizzare le immagini e le recensioni sul gioco), l'aggiunta della funzionalità al menù di gioco della classifica dei migliori punteggi, la funzionalità che permette di recuperare la password e la possibilità di scegliere se giocare in modalità carriera o multiplayer.

6.2.1.1. Use case Utente Ospite – 3° prototipo



Caso d'uso: Profilo utente ospite

Autore: Utente ospite

Pre-condizione: L'utente ospite accede alla schermata relativa all'ospite

Post-condizione: L'utente ospite può eseguire tutte le attività che gli sono permesse (visualizza immagini gioco, recensioni gioco, registrati)

Scenario principale:

- L'utente ospite preme sul pulsante "OSPITE"
- L'utente ospite potrà accedere alla schermata relativa all'ospite

Scenario alternativo:

- L'utente ospite non entrerà nella schermata relativa all'ospite

Caso d'uso: Visualizza immagini gioco

Autore: Utente ospite

Pre-condizione: L'utente ospite accede alla schermata relativa all'ospite

Post-condizione: L'utente ospite visualizza le immagini del gioco

Scenario principale:

- L'utente ospite preme sui pulsanti per scorrere le immagini del gioco
- L'utente ospite potrà visualizzare tutte le immagini inerenti al gioco

Scenario alternativo:

- L'utente ospite visualizzerà soltanto la prima immagine

Caso d'uso: Recensioni gioco

Autore: Utente ospite

Pre-condizione: L'utente ospite accede alla schermata relativa all'ospite

Post-condizione: L'utente ospite può vedere le recensioni sul gioco

Scenario principale:

- L'utente ospite preme sul pulsante "VEDI RECENSIONI"
- L'utente ospite potrà vedere le recensioni sul gioco

Scenario alternativo:

- L'utente ospite non vedrà le recensioni sul gioco

Caso d'uso: Tabella Visualizza commenti

Autore: Utente ospite

Pre-condizione: L'utente ospite accede alla schermata relativa all'ospite e clicca sul tasto per vedere le recensioni

Post-condizione: L'utente ospite vede le recensioni sul gioco

Scenario principale:

- Il commento verrà prelevato dal database e visualizzato nella tabella

Caso d'uso: Registrati

Autore: Utente ospite

Pre-condizione: L'utente ospite accede alla schermata relativa all'ospite

Post-condizione: L'utente ospite può decidere se registrarsi al gioco

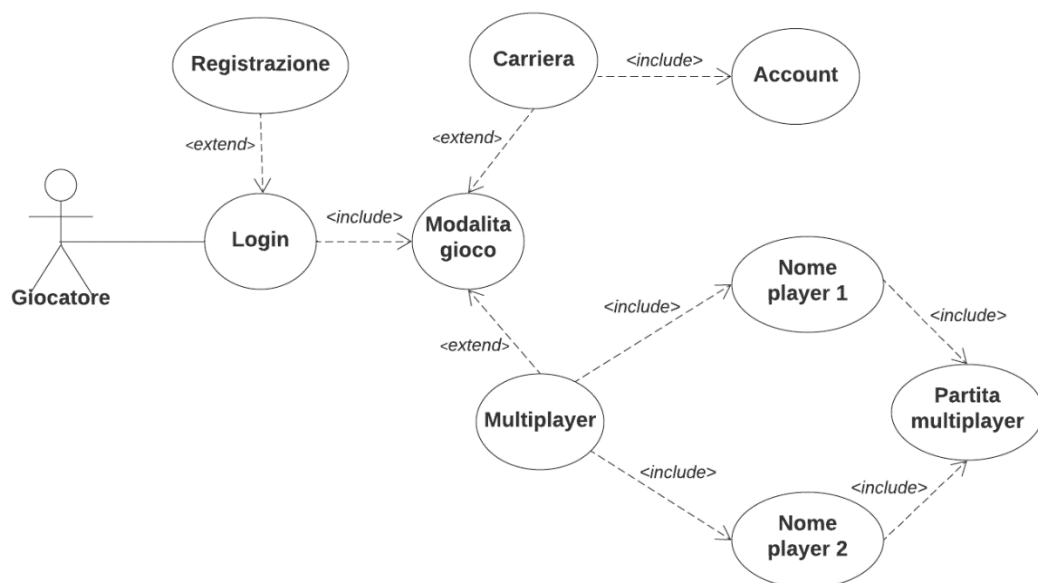
Scenario principale:

- L'utente ospite potrà decidere se registrarsi al gioco cliccando sul pulsante "REGISTRATI"

Scenario alternativo:

- L'utente ospite non sceglierà di registrarsi al gioco

6.2.1.2. Use case Modalità Gioco – 3° prototipo



Vediamo gli scenari mancanti che sono stati definiti in questo terzo prototipo:

Caso d'uso: Modalità gioco

Autore: Giocatore

Pre-condizione: Il giocatore si è loggato e accede alla schermata introduttiva relativa alla modalità di gioco

Post-condizione: Il giocatore può scegliere se giocare in modalità carriera o multiplayer

Scenario principale:

- Il giocatore sceglierà con quale modalità giocare

Caso d'uso: Carriera

Autore: Giocatore

Pre-condizione: Il giocatore si è loggato e accede alla schermata introduttiva relativa alla modalità di gioco

Post-condizione: Il giocatore sceglie di giocare alla modalità carriera

Scenario principale:

- Il giocatore preme sul pulsante “MODALITA' CARRIERA”
- Il giocatore potrà accedere all'account del gioco

Scenario alternativo:

- Il giocatore sceglie la modalità multiplayer

Caso d'uso: Multiplayer

Autore: Giocatore

Pre-condizione: Il giocatore si è loggato e accede alla schermata introduttiva relativa alla modalità di gioco

Post-condizione: Il giocatore sceglie di giocare alla modalità multiplayer

Scenario principale:

- Il giocatore preme sul pulsante “MODALITA' MULTIPLAYER”
- Il giocatore potrà inserire i nomi dei due player che si scontreranno tra loro

Scenario alternativo:

- Il giocatore sceglie la modalità carriera

Caso d'uso: Nome player 1

Autore: Giocatore

Pre-condizione: Il giocatore si è loggato e accedendo alla schermata introduttiva relativa alla modalità di gioco sceglie la modalità multiplayer

Post-condizione: Il giocatore inserisce il nome del primo player

Scenario principale:

- Il giocatore inserito il primo nominativo, attenderà l'inserimento del nome del secondo player

Scenario alternativo:

- Il giocatore non avrà inserito il nome del primo player

Vediamo gli scenari mancanti che sono stati definiti in questo terzo prototipo:

Caso d'uso: Classifica

Autore: Giocatore

Pre-condizione: Il giocatore si è loggato e accedendo alla schermata introduttiva sceglie la modalità carriera

Post-condizione: Il giocatore può vedere la classifica con i migliori 5 risultati ottenuti dagli utenti registrati al gioco

Scenario principale:

- Il giocatore preme sul pulsante “CLASSIFICA”
- Il giocatore vedrà la classifica relativa ai migliori 5 risultati

Scenario alternativo:

- Il giocatore non avendo premuto sul pulsante “CLASSIFICA” non vedrà i 5 migliori risultati

Caso d'uso: Prelevo migliori 5 punteggi dal DB

Autore: Giocatore

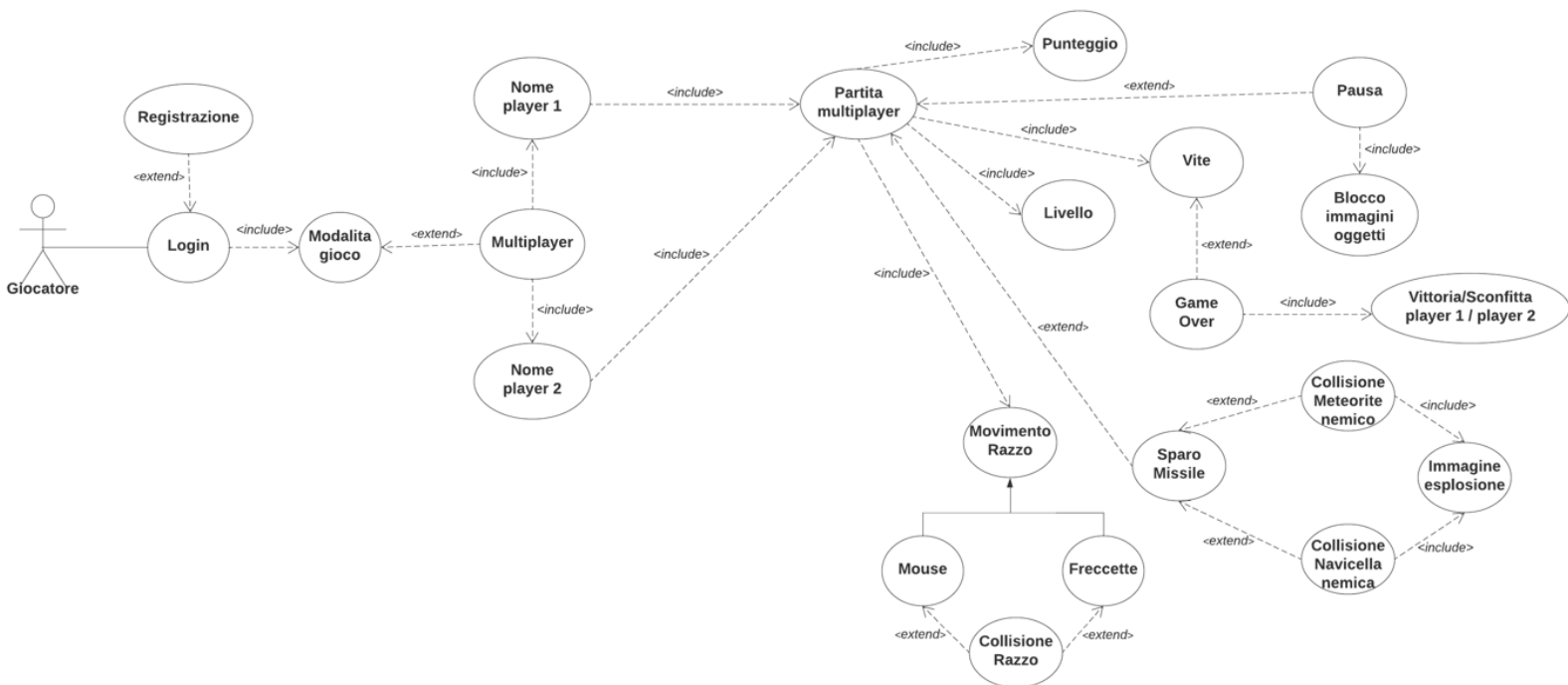
Pre-condizione: Il giocatore si è loggato e accedendo alla schermata introduttiva sceglie la modalità carriera e visualizza i 5 migliori risultati

Post-condizione: La classifica viene prelevata dal database

Scenario principale:

- I migliori 5 risultati ottenuti dai giocatori saranno prelevati dalla tabella “Classifica” e visualizzati dal giocatore

6.2.1.4. Use case Multiplayer – 3° prototipo



Vediamo gli scenari mancanti che sono stati definiti in questo terzo prototipo:

Caso d'uso: Vittoria/Sconfitta player 1 / player 2

Autore: Giocatore

Pre-condizione: Il giocatore si è loggato e accedendo alla schermata introduttiva sceglie la modalità multiplayer e inserisce i nomi dei due player

Post-condizione: I due player hanno esaurito le vite e di conseguenza apparita la schermata “IL PLAYER (1/2) HA VINTO / HA PERSO”

Scenario principale:

- Il player avrà vinto o perso la partita
- Apparirà la schermata per comunicare l'esito della partita hai 2 player

Scenario alternativo:

- La partita tra i due player non è ancora terminata

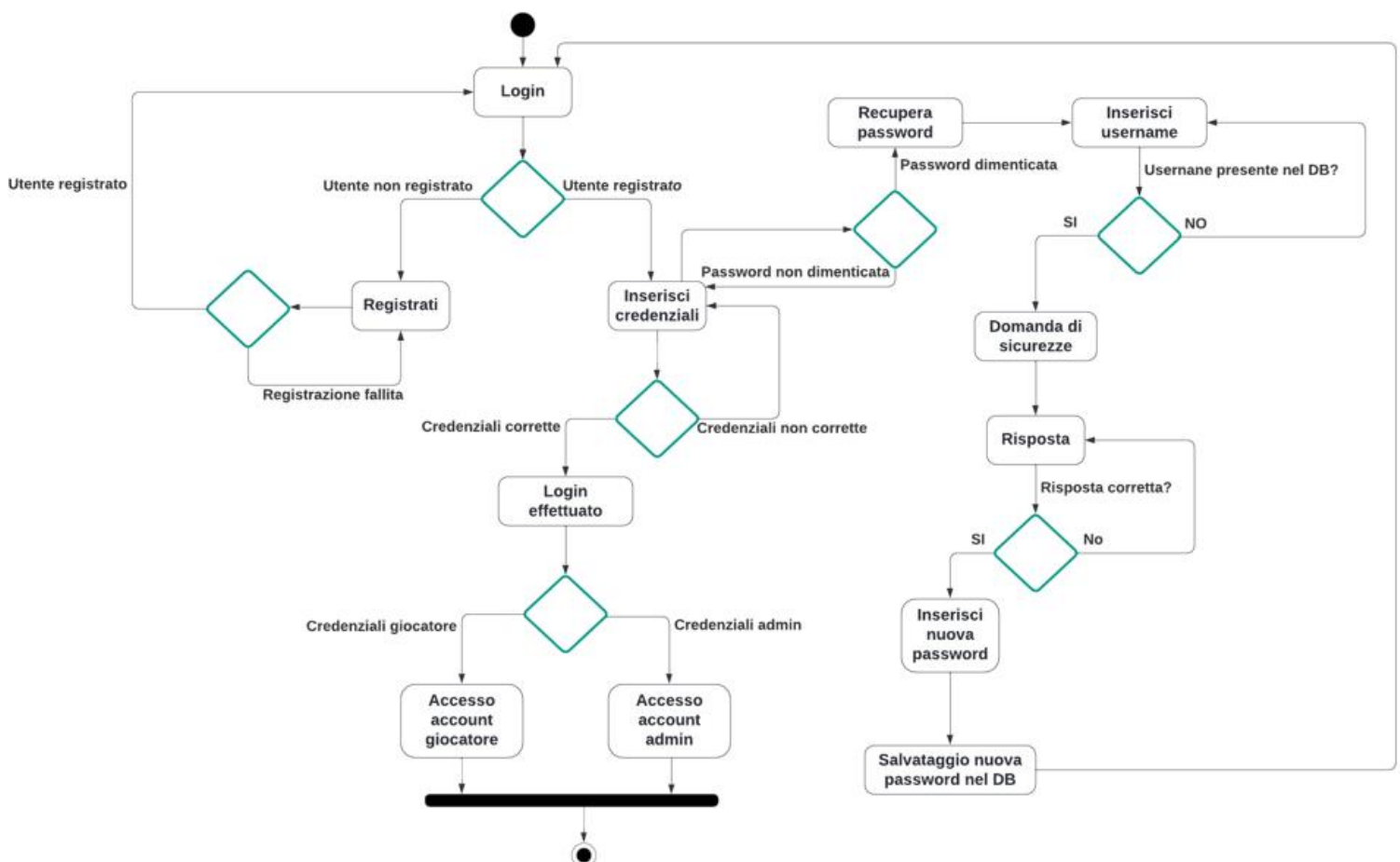
6.2.2. Activity Diagram

In questo terzo prototipo gli activity diagram definiti nel secondo prototipo sono stati modificati aggiungendo le nuove funzionalità descritte negli use case del secondo prototipo.

6.2.2.1. Activity Diagram Recupera password – 3° prototipo

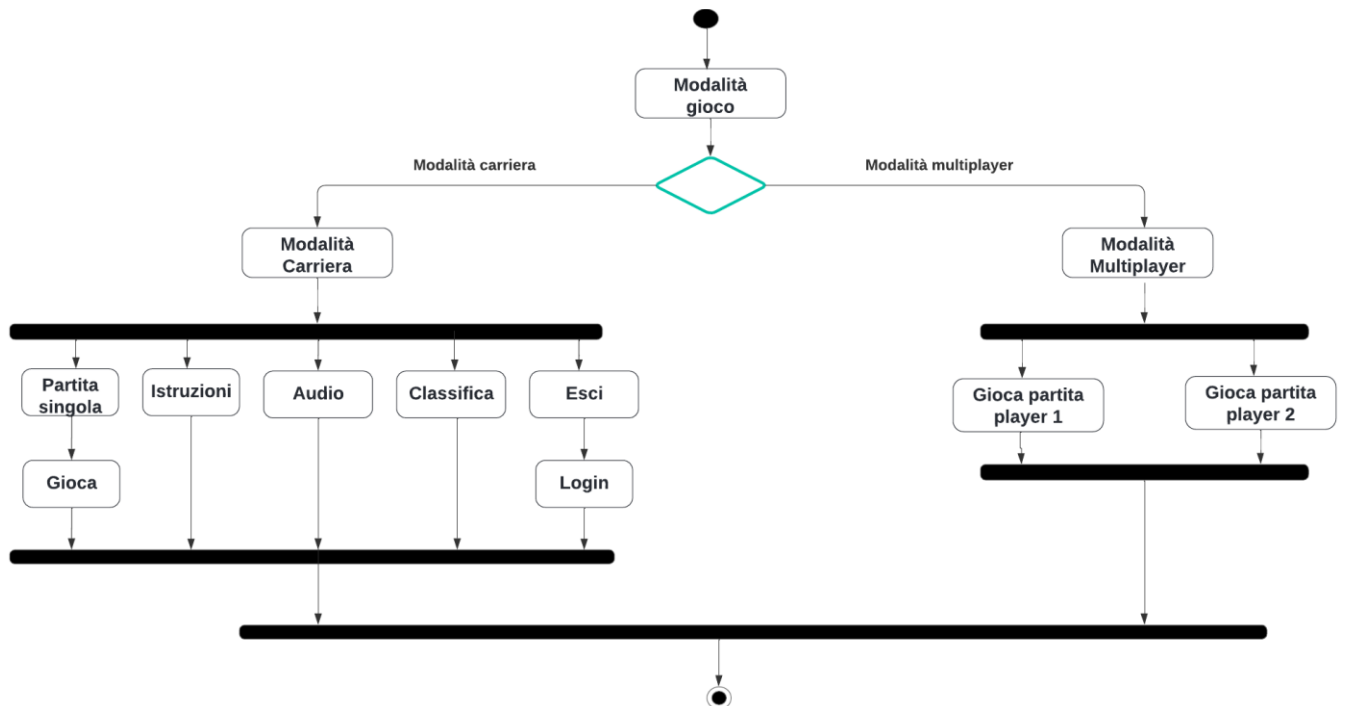
Una volta avviato il gioco si verrà indirizzati nella pagina di login. A questo punto si hanno tre possibilità: loggarsi con le credenziali del giocatore, loggarsi con le credenziali dell'admin (queste due attività sono state descritte nell'activity diagram "Login/Registrazione" del primo e del secondo prototipo) oppure (nel caso del giocatore) andare a inserire una nuova password. In quest'ultimo caso se l'username inserito dal giocatore è presente nel database, apparirà la domanda di sicurezza al quale il giocatore dovrà rispondere, altrimenti verrà comunicato al giocatore che l'username non risulta memorizzato nel DB.

A questo punto se la risposta è corretta (cioè la risposta coincide con quella data in fase di registrazione) il giocatore potrà scrivere la nuova password che verrà aggiornata nel database, altrimenti verrà richiesto di riscrivere la risposta in quanto quest'ultima risulta sbagliata.



6.2.2.2. Activity Diagram Modalità Gioco – 3° prototipo

Una volta effettuato il login si verrà indirizzati nella pagina relativa alla scelta della modalità di gioco. A questo punto il giocatore ha due possibilità: giocare in modalità carriera oppure, se ha raggiunto il livello 5 in quest'ultima modalità potrà scegliere se giocare in modalità multiplayer. Scegliendo la prima modalità, il giocatore avrà la possibilità di scegliere l'attività da svolgere (tutte le attività sono state descritte nel prototipo precedente a eccezione dell'attività classifica che permetterà di visualizzare i 5 migliori risultati ottenuti nel gioco dai giocatori loggati). Scegliendo la seconda modalità, il giocatore avrà la possibilità di iniziare una partita multiplayer in locale, inserendo prima i due nomi dei player.

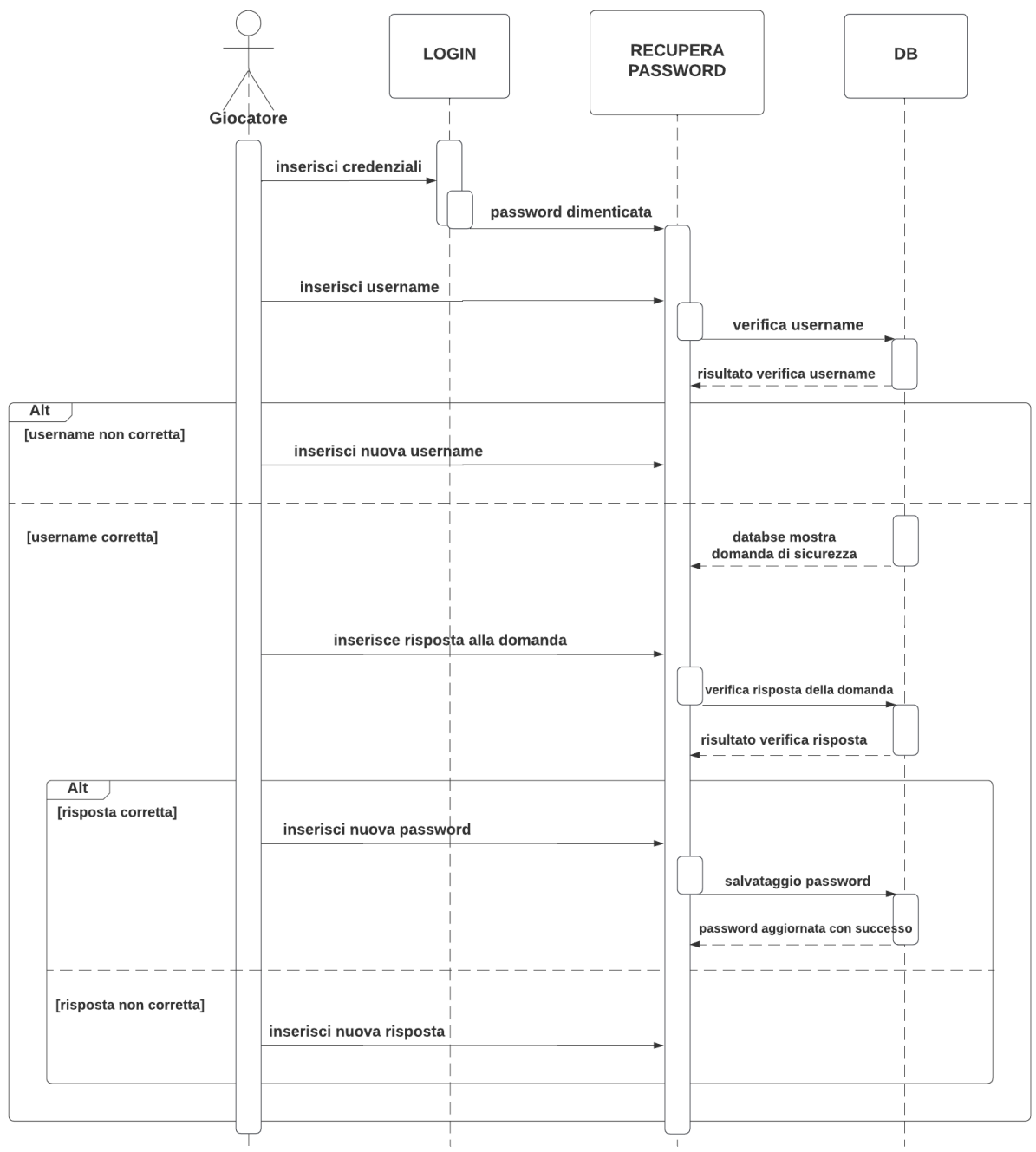


6.2.3. Sequence Diagram

In questo terzo prototipo abbiamo creato due sequence diagram, uno riguardante il recupero password e uno riguardante il multiplayer.

6.2.3.1. Sequence Diagram Recupera password – 3° prototipo

Nel Sequence Diagram seguente, si mostra il flusso rappresentato dall'Activity Diagram del paragrafo precedente riguardante la login e la registrazione del giocatore. In particolare mostra come il giocatore interagisce con la finestra relativa alla login la quale a sua volta interagisce con la finestra relativa al recupero password che interagirà con il database per permettere all'utente giocatore di memorizzare una nuova password.



6.2.3.2. Sequence Diagram Partita multiplayer - 3° prototipo

Nel Sequence Diagram seguente, si mostra il flusso riguardante la partita multiplayer. In particolare mostra come i due player interagiscono con il server il quale a sua volta interagisce con entrambi per stabilire l'inizio della partita e comunicare quale dei due player ha vinto/perso la partita.



6.3. Implementazione

Nel terzo prototipo abbiamo implementato le schermate relative:

- All'utente ospite
- Al recupera password
- Alla scelta della modalità di gioco.

Inoltre, abbiamo implementato la funzionalità aggiuntive per la modalità carriera come la possibilità di visualizzare i 5 migliori risultati ottenuti dai giocatori nel gioco.

Di seguito vengono mostrate le modifiche effettuate nelle rispettive classi e l'implementazione delle nuove classi definite all'interno del terzo prototipo.

6.3.1. Classe FinestraBenvenuto

Nella **classe FinestraBenvenuto** abbiamo aggiunto un pannello contenente due bottoni “OSPITE” e “UTENTE” che permettono alla persona di poter scegliere se continuare come utente ospite o come utente giocatore.

6.3.2. Classe FinestraOspite

La **classe FinestraOspite** è la classe che definisce il frame relativo alla schermata dell’ospite contenente le opzioni che permettono all’utente ospite di svolgere delle attività.

All’interno di questa classe:

- definiamo le dimensioni del frame, il titolo, l’immagine di sfondo e l’immaginetta che apparirà come logo accanto al nome del gioco,
- creiamo due label a cui associamo un evento che richiama la classe MouseEvent nel quale ridefiniamo, mediante Override, il metodo MouseClicked che permetterà all’utente ospite (cliccando sulla rispettiva immagine della label) di scorrere le immagini di gioco,
- creiamo i bottoni “REGISTRATI” – “VEDI RECENSIONI” e una text field che permettono all’utente ospite di passare alla finestra di registrazione e vedere le recensioni che appariranno sotto forma di tabella all’interno della text field che verranno prelevate (dopo aver effettuato la connessione) dal database,
- definiamo il posizionamento delle varie componenti all’interno della finestra,
- implementiamo l’interfaccia ActionListener, ovvero l’ascoltatore degli eventi legato ai bottoni che implementa un metodo actionPerformed che permette di associare al rispettivo bottone il comportamento ad esso attribuitogli,
- aggiungiamo l’ascoltatore degli eventi relativo ai 2 bottoni.

6.3.3. Classe FinestraConLogin

Nella **classe FinestraConLogin** abbiamo aggiunto il bottone “RECUPERA PASSWORD” che indirizzerà l’utente (nel caso non ricordi la password) nella schermata riguardante il recupero della password.

6.3.4. Classe FinestraRecuperaPassword

La **classe FinestraRecuperaPassword** è la classe che definisce il frame relativo alla schermata che permetterà all’utente giocatore di inserire una nuova password nel caso non la ricordi.

All’interno di questa classe:

- definiamo le dimensioni del frame, il titolo, l’immagine di sfondo e l’immaginetta che apparirà come logo accanto al nome del gioco,
- creiamo le label e le Text Field, per inserire l’username dell’utente, la domanda di sicurezza, la risposta e la nuova password,
- creiamo i bottoni “CERCA UTENTE” – “SALVA PASSWORD” che permettono all’utente giocatore di cercare l’username e far comparire la domanda di sicurezza (nel caso l’username fosse corretta) e salvare la nuova password inserita (nel caso la risposta alla domanda fosse corretta),
- definiamo il posizionamento delle varie componenti all’interno della finestra,

- implementiamo l'interfaccia ActionListener, ovvero l'ascoltatore degli eventi legato ai bottoni che implementa un metodo actionPerformed che permette di associare al rispettivo bottone il comportamento ad esso attribuitogli,
- aggiungiamo l'ascoltatore degli eventi relativo ai 2 bottoni,
- definito una label "TORNA ALLA LOGIN" a cui associamo un evento che richiama la classe MouseEvent nel quale ridefiniamo mediante Override il metodo MouseClicked che permetterà all'utente di passare (dopo aver inserito la nuova password) alla finestra di login,
- infine, effettueremo la connessione con il database per verificare la correttezza dei dati inseriti e per memorizzarli all'interno del database.

6.3.5. Classe FinestraModalitaGioco

La **classe FinestraModalitaGioco** è la classe che definisce il frame relativo alla schermata che permetterà all'utente giocatore di scegliere la modalità con cui giocare.

All'interno di questa classe:

- definiamo le dimensioni del frame, il titolo, l'immagine di sfondo e l'immaginetta che apparirà come logo accanto al nome del gioco,
- definiamo delle label per arricchire la grafica del gioco e permettere al giocatore di visualizzare il suo livello massimo raggiunto (tale livello verrà poi utilizzato per verificare che sia \geq a 5 permettendo così di avviare, nel caso venga scelta, la modalità multiplayer),
- creiamo i bottoni "MODALITA' CARRIERA" – "MODALITA' MULTIPLAYER" che permettono all'utente giocatore di passare o alla finestra di gioco relativa alla modalità carriera o di iniziare una partita in modalità multiplayer (dopo aver effettuato la connessione con il database per verificare che il livello raggiunto dal giocatore sia \geq 5),
- definiamo il posizionamento delle varie componenti all'interno della finestra,
- implementiamo l'interfaccia ActionListener, ovvero l'ascoltatore degli eventi legato ai bottoni che implementa un metodo actionPerformed che permette di associare al rispettivo bottone il comportamento ad esso attribuitogli,
- aggiungiamo l'ascoltatore degli eventi relativo ai 2 bottoni.

6.3.6. Classe FinestraBottoni

Nella **classe FinestraBottoni** è stata aggiunta la funzionalità che permette di visualizzare i 5 miglior risultati raggiunti dagli utenti registrati al gioco.

Per far questo abbiamo:

- implementato un metodo per creare un pannello all'interno del quale verrà inserita una tabella contenente i 5 migliori risultati del gioco,
- definito un bottone "CLASSIFICA" e implementati dei metodi che permettono di prelevare dal database l'username, il livello e il punteggio dei migliori 5 risultati ottenuti dagli utenti registrati e inserirli nella tabella definita all'interno del pannello.

6.3.7. Classe Client

Nella **classe Client** un client invia messaggi al server, il server genera un thread per comunicare con il client. Ogni comunicazione con un client viene aggiunta a un array list in modo che qualsiasi messaggio venga inviato a tutti gli altri client in sequenza. Per far questo abbiamo fatto in modo che:

- un client utilizza una socket per connettersi al server e un bufferreader/bufferwriter rispettivamente per ricevere e inviare messaggi,
- una volta che il primo client avrà inserito il suo nickname, rimarrà in attesa del secondo client (in attesa di GO), quando il secondo utente si collegherà inserendo il suo nickname avrà inizio la partita (GO),
- quando verrà letto il messaggio END per entrambi i player la partita tra i due player terminerà,
- infine abbiamo creato un metodo per chiudere la socket e i due buffer.

6.3.8. Classe ClientHandler

Nella **classe ClientHandler** permette di gestire la comunicazione tra più client. Quando un client si connette, il server genera un thread per gestire il client, in questo modo il server può gestire più client contemporaneamente. Per far questo:

- definiamo una socket per connettersi al server, un bufferreader e un bufferwriter rispettivamente per ricevere e inviare messaggi e un array list di tutti i thread che gestiscono i client in modo che ogni messaggio possa essere inviato al client,
- creiamo un gestore client che passeremo alla classe GameServer. Quando un client si connette, viene inviato il suo username, viene aggiunto il nuovo gestore client all'array in modo che i client possano ricevere messaggi e viene controllato che il numero di partecipanti sia uguale a 2,
- una volta stabilita la connessione, leggiamo i messaggi inviati da i due client, se il messaggio ricevuto è KO, verrà effettuato un confronto tra i due punteggi e verrà stabilito il player vincitore e il player sconfitto,
- inoltre abbiamo creato un metodo che fa in modo che se il client si disconnette per qualsiasi motivo, verrà rimosso dall'elenco in modo tale che il messaggio non verrà inviato a una connessione interrotta,
- infine abbiamo creato un metodo per chiudere la socket, i due buffer e se il client si è disconnesso o si è verificato un errore, lo rimuoviamo dall'elenco in modo che nessun messaggio venga trasmesso.

6.3.9. Classe GameServer

Nella **classe GameServer** è la classe che utilizziamo per definire la connessione sulla porta 1234 della socket e ascoltare le connessioni (dei due client da collegare) sulla medesima porta.

6.3.10. Classe PlayerClient

La **classe PlayerClient** è la classe perno dove vengono implementate le varie dinamiche del gioco per la modalità multiplayer.

All'interno di questa classe:

- inizialmente impostiamo l'immagine di sfondo del gioco, passiamo la sua dimensione e definiamo la posizione del razzo all'interno del pannello,
- aggiungiamo l'interfaccia pubblica MouseMotionListener che sovrascrive il metodo mouseMoved che permette la ricezione degli eventi di movimento del mouse su un componente (razzo),
- aggiungiamo l'interfaccia pubblica KeyListener che sovrascrive il metodo keyPressed che permette la ricezione degli eventi legati al tasto che viene premuto sulla tastiera (in particolare per permettere al giocatore di spostarsi con le frecce left e right e di sparare il missile con la barra spaziatrice),
- metodo paint (@Override): metodo che permette di disegnare i vari oggetti: razzo, navicella nemica, meteoriti (gli oggetti nemici verranno cancellati quando usciranno dal riquadro di gioco), missile, di inserire le varie scritte relative al gioco, definire l'immagine di sfondo, la posizione e il formato,
- metodo nemici: metodo che permette di aggiungere in modo random sia i meteoriti che la navicella nemica,
- metodo collisione e metodo collisioneRazzo: metodi che permettono di gestire la collisione dei vari oggetti nemici con il razzo,
- metodo collisione e metodo collisioneProiettile: metodo che permette di gestire la collisione tra il missile e i vari oggetti nemici, eliminare l'oggetto nemico colpito, assegnare il punteggio corrispondente all'oggetto colpito e che permette di far apparire l'immagine dell'esplosione,
- metodo colpito: metodo che permette di decrementare le vite quando si verrà colpiti, di far comparire la schermata FinestraFineGameMultiplayer con le scritte relative al punteggio e al vincitore/sconfitto e di gestire il tempo di invulnerabilità,
- metodo play: metodo che permette di avviare il gioco e che andrà a gestire le varie dinamiche del gioco,
- metodo connectToServer: metodo che permette di inserire il nickname dell'utente e di creare la socket per la connessione con al server.

6.3.11. Classe FinestraFineGameMultiplayer

La **classe FinestraFineGameMultiplayer** è la classe che definisce il frame relativo alla schermata nella quale verrà visualizzato il player vincitore e i punteggi ottenuti dai due utenti.

All'interno di questa classe:

- definiamo le dimensioni del frame e dei componenti al suo interno, la posizione di questi componenti, il titolo, il colore di sfondo e l'immaginetta che apparirà come logo accanto al nome del gioco,
- creiamo il pannello con dentro lo ScrollPane che conterrà il messaggio,
- infine creiamo un TextPane che conterrà il messaggio con il nome del player che ha vinto/perso e i punteggi raggiunti dai due player.
-

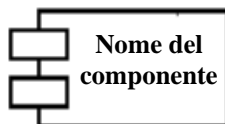
6.4. Component Diagram e Deployment Diagram

In UML esistono due tipi di diagrammi per la modellazione degli aspetti implementativi (detti Implementation Diagram) di un sistema software. Questi due diagrammi sono:

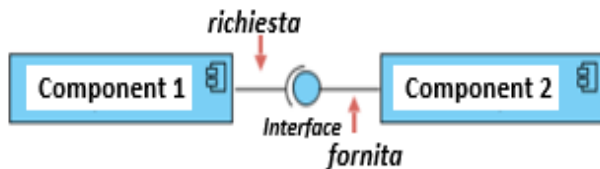
- **Component Diagram**
- **Deployment Diagram**

Il **Component Diagram** è un diagramma UML che serve a definire l'architettura modulare del nostro sistema software da un punto di vista statico.

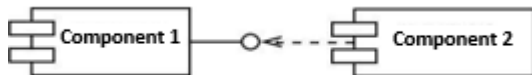
L'obiettivo è capire quali sono le componenti che realizzano l'applicazione, le loro dipendenze e come le interfacce interagiscono tra questi componenti.



I componenti sono un'astrazione del concetto di classe e rappresentano un'unità logica del sistema; vengono rappresentati con un rettangolo contenente un piccolo rettangolo in alto a destra.



Le interfacce descrivono un gruppo di operazioni usate, richieste, create o fornite dai componenti. Un cerchio rappresenta un'interfaccia usata o fornita da un componente, mentre un semicerchio rappresenta un'interfaccia richiesta.

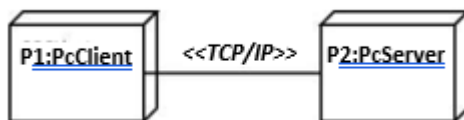


Le dipendenze vengono rappresentate da una linea tratteggiata, e mostrano quali componenti sono dipendenti da altri componenti.

Il **Deployment Diagram** è un diagramma UML che serve per rappresentare l'infrastruttura hardware (cioè l'architettura fisica) sul quale dovrà girare il nostro sistema. L'obiettivo è quello di favorire la creazione di un quadro chiaro su come la composizione finale dell'hardware dovrà essere, permettendoci, in più, di capire come le componenti software (collegati tra loro) verranno distribuito fisicamente sull'hardware. Per quanto riguarda gli elementi grafici di questo diagramma possiamo distinguere due elementi:



un cubo che definisce un nodo che raffigura una risorsa hardware disponibile al sistema

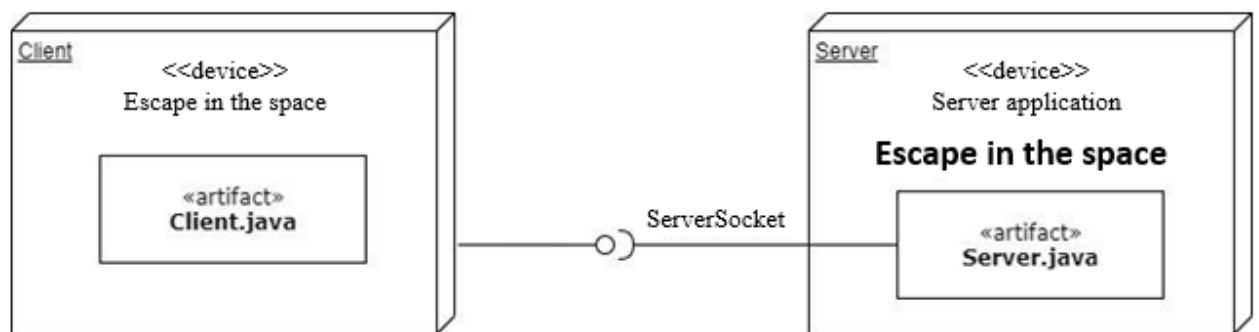


una linea di connessione che definisce un collegamento tra due nodi.

6.4.1. Component/Deployment Diagram -Escape in the space

Spesso si utilizza un diagramma che mostra come le componenti software siano distribuite rispetto alle risorse hardware disponibili sul sistema; questo tipo di diagramma viene rappresentato unendo il Component Diagram e il Deployment Diagram.

Di seguito viene riportato il Component/Deployment Diagram relativo al sistema client-server del nostro progetto:



6.5. Testing

Una volta ultimata la fase implementativa del 3° prototipo siamo passati alla fase del test. In questo caso bisognerà valutare non soltanto la correttezza dei test relativi al primo e secondo prototipo, ma anche verificare che le nuove funzionalità introdotte non hanno avuto effetti dannosi o hanno peggiorato le funzionalità del sistema. Nelle tabelle di seguito sono riportati i nuovi test di verifica e validazione effettuati più i test ripetuti del primo e del secondo prototipo con il relativo esito.

6.5.1. Verifica

In questa fase si verificherà se il software progettato è privo di errori. Per questo motivo verranno effettuati dei test con lo scopo di cercare di trovare qualche errore, inserendo un'insiemi di dati in tutti i possibili input del programma.

Test	Descrizione	Esito
Test 1	Login con credenziali giocatore corrette	✓
Test 2	Login con credenziali admin corrette	✓
Test 3	Login con credenziali errate	✗
Test 4	Login con credenziali mancanti	✗

Test 5	Username corretta per recuperare la password genera domanda di sicurezza	✓
Test 6	Username non corretta per recuperare la password genera domanda di sicurezza	✗
Test 7	Risposta corretta permette di inserire la nuova password	✓
Test 8	Risposta non corretta permette di inserire la nuova password	✗
Test 9	Registrazione con dati corretti	✓
Test 10	Registrazione con dati obbligatori mancanti	✗
Test 11	Registrazione con username già esistente	✗
Test 12	Registrazione con le due password inserite che non coincidono	✗

Come vediamo dalla tabella, gli aspetti riguardanti la login, la registrazione e il recupera password procedono come previsto e non si verifica nessun risultato inatteso. Pertanto nel caso in cui le credenziali inserite in fase di login dal giocatore (o dall'admin) sono corrette si verrà indirizzati alla finestra relativa all'account del giocatore (o dell'admin), in caso contrario la login avrà esito negativo in quanto le credenziali risultano sbagliate o mancanti. Analogamente nel caso in cui l'username inserita dal giocatore è corretta, comparirà la domanda di sicurezza con il quale l'utente si è registrato, in caso contrario si avrà esito negativo e la domanda di sicurezza non comparirà. A questo punto nel caso in cui l'username inserita è corretta e la risposta inserita dal giocatore è corretta, esso potrà inserire la nuova password, in caso contrario si avrà esito negativo e la password del giocatore non sarà memorizzata nel database. Analogamente nel caso in cui i dati inseriti in fase di registrazione sono corretti, verrà creato il nuovo profilo relativo all'utente, in caso contrario la registrazione avrà esito negativo in quanto l'username è già esistente, i dati obbligatori inseriti sono mancanti o le due password inserite non coincidono.

6.5.2. Validazione

In questa fase si valuta che il software soddisfi ogni requisito funzionale definito in fase di analisi dei requisiti. Per questo motivo verranno effettuati dei test con lo scopo di verificare che tutti i requisiti stabiliti secondo il contratto sono stati implementati e risultano funzionanti.

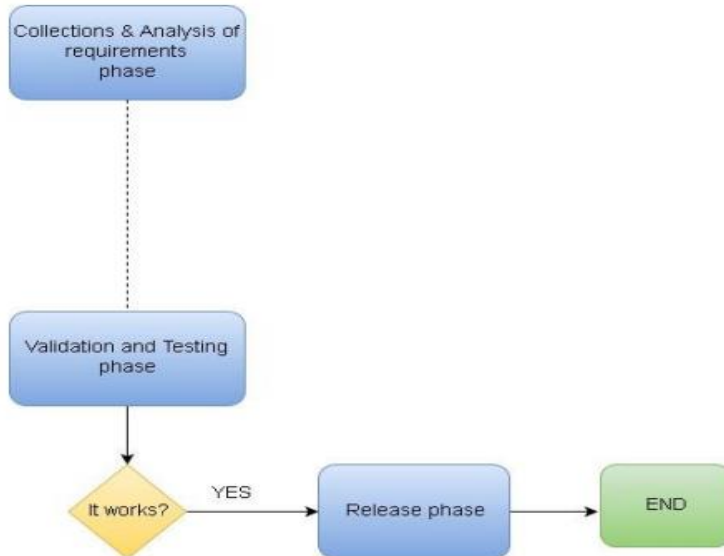
ID	Descrizione	Esito
RF-01	L'utente deve essere registrato per poter giocare	✓
RF-02	L'utente deve poter vedere le istruzioni su come giocare	✓
RF-03	L'utente deve poter mettere l'audio del gioco	✓
RF-04	L'utente deve poter effettuare il logout in qualsiasi momento	✓
RF-05	L'utente deve poter giocare spostandosi sia con le freccette sia con il mouse	✓
RF-06	L'utente deve poter visualizzare mentre gioca le vite rimaste e il punteggio accumulato evitando gli oggetti nemici	✓
RF-07	Quando il player viene colpito le vite dovranno diminuire	✓
RF-08	Il player quando verrà colpito deve rimanere invulnerabile per un periodo uguale a 1.5 secondi	✓
RF-09	Il gioco deve terminare non appena le vite rimaste si azzerano (Game Over)	✓
RF-10	L'utente deve poter scegliere se ricominciare una nuova partita immediatamente dopo aver perso	✓

RF-11	Devono essere attribuite all'admin delle credenziali per potersi connettere come amministratore del gioco	✓
RF-12	L'admin deve poter visualizzare le persone registrate al gioco	✓
RF-13	L'admin deve poter eliminare gli utenti inserendo il loro username	✓
RF-14	L'admin deve poter eliminare un commento che ritiene inappropriato	✓
RF-15	L'admin deve poter effettuare il logout in qualsiasi momento	✓
RF-16	L'admin deve poter avere la possibilità di cercare un utente inserendo l'username relativo a quell'utente	✓
RF-17	Il player deve poter mettere in pausa il gioco cliccando sul pulsante P	✓
RF-18	Il player deve poter riprendere la partita in un qualsiasi momento cliccando nuovamente sul pulsante P	✓
RF-19	Il player deve avere la possibilità di sparare dei missili in verticale usando la barra spaziatrice sia per difendersi che per accumulare dei punti	✓
RF-20	Quando una partita di un giocatore termina il punteggio ottenuto e il livello raggiunto deve essere memorizzato sul database	✓
RF-21	La velocità di caduta degli oggetti nemici deve aumentare all'aumentare del livello di difficoltà.	✓
RF-22	L'utente deve avere la possibilità di visualizzare il suo miglior punteggio ottenuto nel gioco.	✓

RF-23	L'utente deve avere la possibilità di poter aggiungere un commento sul gioco	✓
RF-24	L'utente ospite deve poter visualizzare le immagini relative al gioco	✓
RF-25	L'utente ospite deve poter visualizzare delle recensioni sul gioco	✓
RF-26	L'utente ospite una volta visualizzate le immagini e le recensioni deve poter scegliere se registrarsi al gioco	✓
RF-27	L'utente registrato deve poter inserire una nuova password dopo aver verificato che la risposta alla domanda sia corretta e che l'username corrisponde all'utente in questione	✓
RF-28	Il giocatore una volta loggato deve poter visualizzare i 5 migliori risultati ottenuti dagli utenti registrati al gioco	✓
RF-29	Il giocatore deve poter scegliere se giocare in modalità carriera o multiplayer	✓
RF-30	Deve essere effettuato un controllo per verificare che il giocatore ha raggiunto almeno il livello 5 nella modalità carriera per giocare una partita in modalità multiplayer	✓
RF-31	I due player della modalità multiplayer devono poter inserire i loro rispettivi nomi per giocare	✓
RF-32	Nella modalità multiplayer dovrà apparire il messaggio "HAI VINTO" o "HAI PERSO" a seconda del risultato ottenuto dai due player	✓
RF-33	Nella modalità multiplayer dovrà apparire il punteggio dei 2 player	✓

Come vediamo dalla tabella, dopo aver risolto i problemi riscontrati nel prototipo precedente, ovvero l'esito negativo per quanto riguarda i requisiti RF-16 e RF-22, tutti i Requisiti Funzionali hanno avuto esito positivo.

6.5.3. Valutazione 3° prototipo con feedback del client

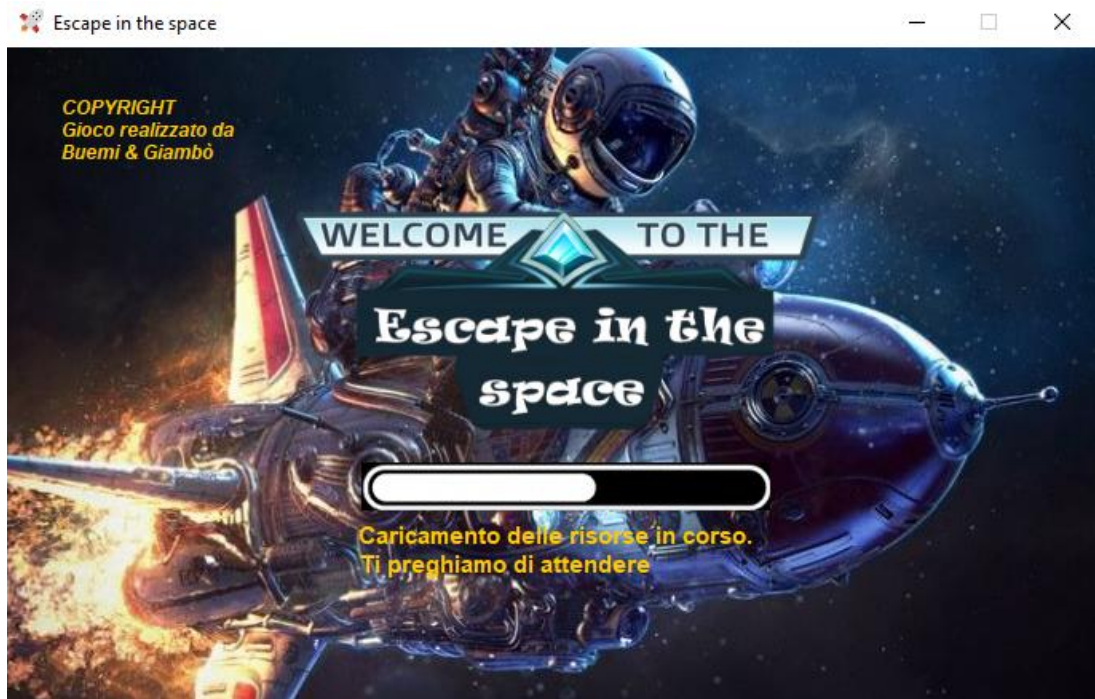


Subito dopo aver effettuato la fase di test, il team ha deciso di mostrare il 3° prototipo al cliente che ha apprezzato il lavoro svolto. Quest'ultimo non ha richiesto ulteriori features, tutti i requisiti sono stati rispettati e la versione finale risulta essere completa ed efficiente in ogni sua parte. Di conseguenza, si è considerato il terzo prototipo come il prototipo finale (**Final Release**).

7. Meccaniche del gioco

In questa sezione vedremo le varie meccaniche di gioco con le relative schermate

7.1. Schermata di benvenuto



Una volta che l'utente avvierà il gioco, apparirà la schermata introduttiva e dopo aver aspettato il caricamento delle risorse, l'utente potrà scegliere tra due bottoni:

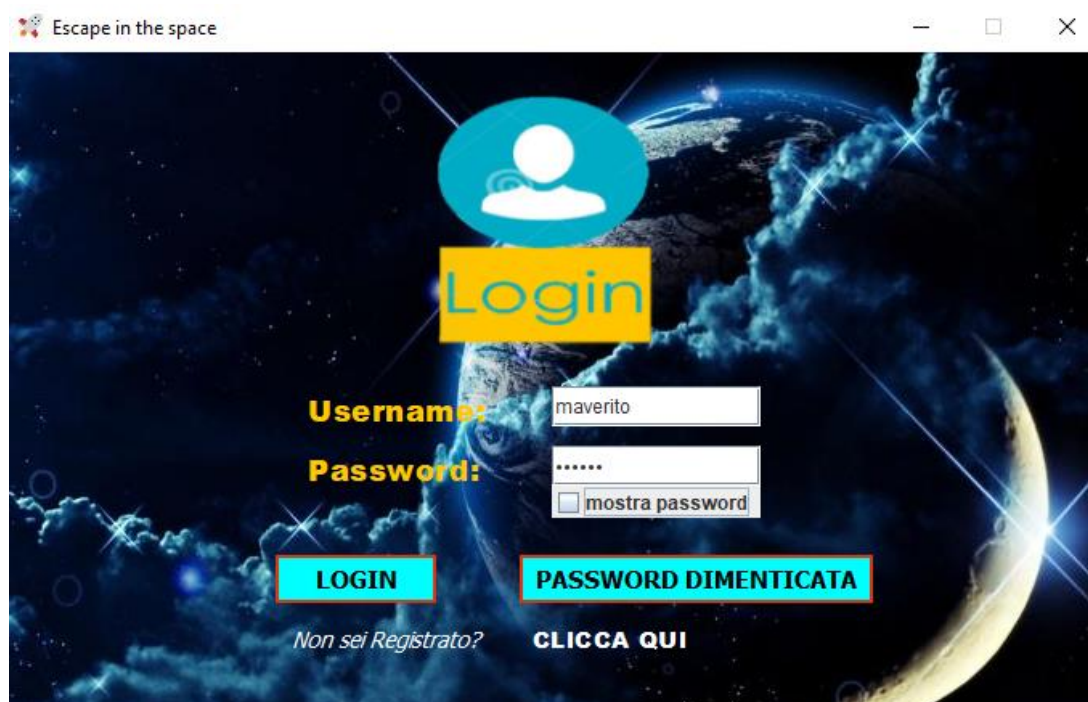
- **OSPITE:** permetterà all'utente di accedere al gioco senza essersi registrato.
- **UTENTE:** permetterà all'utente di effettuare il login al gioco.

7.2. Schermata ospite



Se l'utente avrà cliccato sul bottone **OSPITE**, apparirà tale schermata in cui l'utente potrà visualizzare le immagini inerenti al gioco, vedere le recensioni sul gioco e decidere se registrarsi cliccando sull'apposito bottone.

7.3. Schermata di login



Se l'utente avrà cliccato sul bottone **UTENTE**, apparirà la schermata per il login, in cui l'utente inserendo le sue credenziali corrette potrà accedere al suo account e scegliere la modalità di gioco.

7.4. Schermata per recuperare la password

Escape in the space

Username: **CERCA UTENTE**

Domanda di sicurezza:

Risposta:

Nuova password:

SALVA PASSWORD

[Torna alla login](#)

Se l'utente nel caso del login, non si dovesse ricordare la password, potrà cliccare sul bottone **PASSWORD DIMENTICATA** e apparirà tale schermata in cui l'utente inserendo i dati corretti potrà impostare una nuova password.

7.5. Schermata di registrazione

Escape in the space

REGISTER NOW

Nome utente:

Anno di nascita:

Username:

Password:

Ripeti password:

Domanda di sicurezza:

Risposta:

Email:

[Torna alla login](#) **REGISTRATI** **CANCELLA**

Se l'utente nel caso del login non fosse registrato, potrà farlo cliccando sul bottone **CLICCA QUI** e apparirà la schermata di registrazione in cui l'utente inserendo i suoi dati personali potrà registrarsi al gioco.

7.6. Schermata per la scelta della modalità di gioco



Una volta che l'utente si sarà loggato, apparirà la schermata relativa alla scelta della modalità di gioco e l'utente potrà scegliere tra due bottoni:

- **MODALITA' CARRIERA:** permetterà all'utente di accedere al suo account per giocare in modalità carriera.
- **MODALITA' MULTIPLAYER:** permetterà all'utente di giocare una partita in modalità multiplayer inserendo i due nominativi dei player.

7.6.1. Schermata per la partita multiplayer

Input

✕

?

Inserire il nome

ema

OK

Cancel

Input

✕

?

Inserire il nome

fil

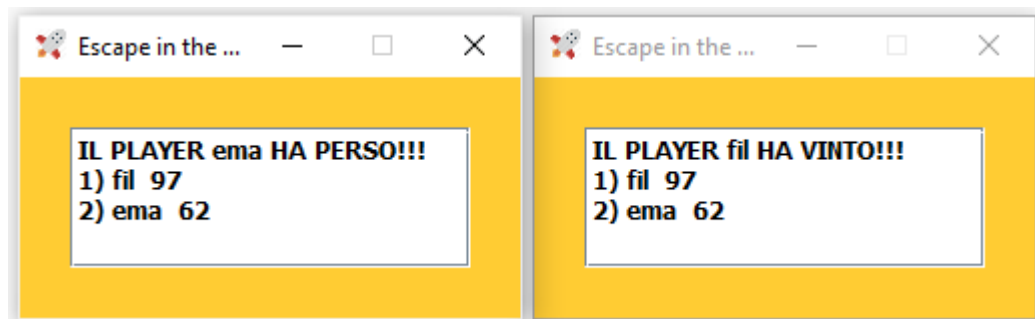
OK

Cancel

Qui possiamo vedere, nel caso l'utente scelga la modalità multiplayer un esempio di come apparirà la schermata quando verrà svolta una partita in questa modalità.



Dopo che i due player avranno terminato la partita, apparirà la finestra con il player vincitore e il player sconfitto (insieme ai relativi punteggi ottenuti).



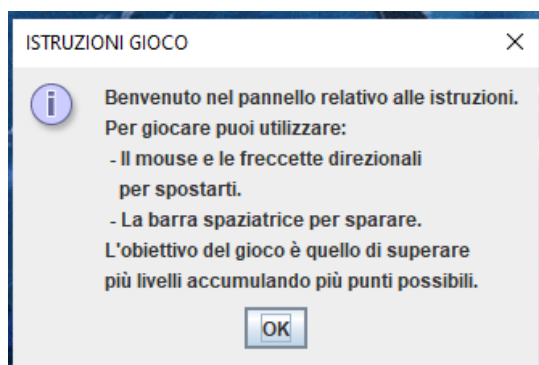
7.7. Schermata dell'account per la modalità carriera



Nel caso in cui l'utente sceglie di giocare in modalità carriera, apparirà la schermata relativa al suo account e l'utente potrà scegliere tra 5 bottoni:

- **PARTITA SINGOLA:** permetterà all'utente di iniziare una partita.
- **ISTRUZIONI:** permetterà all'utente di vedere le istruzioni su come giocare.
- **AUDIO OFF:** permetterà all'utente di attivare l'audio del gioco.
- **CLASSIFICA:** permetterà all'utente di visualizzare i 5 migliori risultati (record) tra tutti i giocatori registrati al gioco.
- **ESCI:** permetterà all'utente di effettuare il logout.

In più l'utente nell'apposito spazio inserito in basso potrà scrivere un suo parere sul gioco e cliccare sul bottone **AGGIUNGI COMMENTO** per pubblicare il commento.



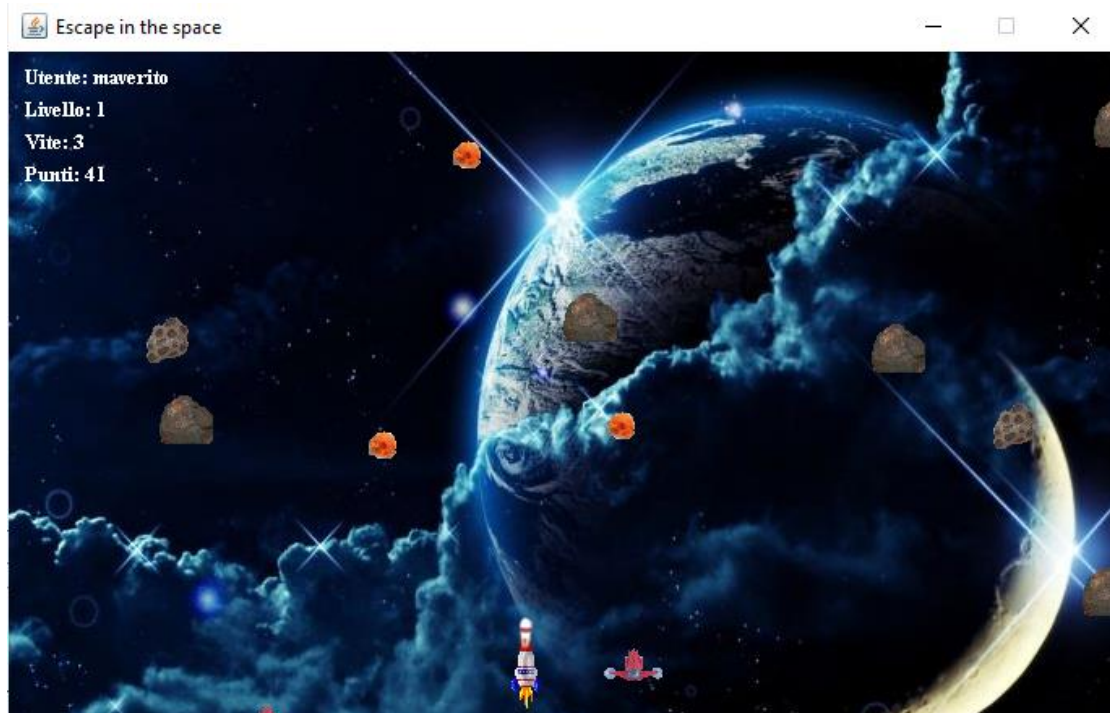
Pannello istruzioni

I 5 migliori punteggi del gioco

USERNAME	RECORD	LIVELLO
francy	1430	7
maverito	1109	6
espana	667	5
dovipower	576	4
maverito	539	4

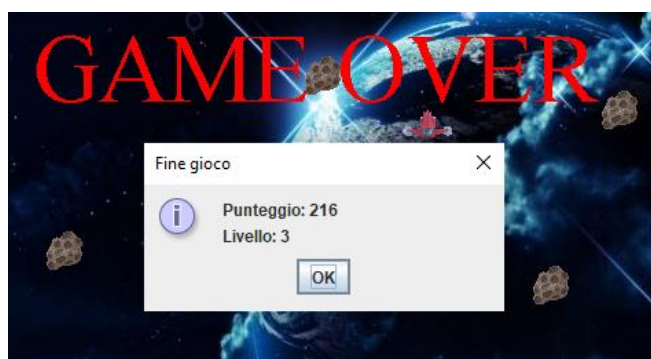
Pannello classifica

7.7.1. Schermata per la partita singola



Qui possiamo vedere, nel caso l'utente scelga la modalità carriera un esempio di come apparirà la schermata quando verrà svolta una partita in questa modalità.

In alto a sinistra possiamo notare 4 features molto importanti, cioè l'username dell'utente, il livello attualmente raggiunto, le vite rimaste e i punti accumulati. Il giocatore avrà a disposizione 3 vite e ogni volta che un meteorite o una navicella nemica colpirà il razzo, il valore della vita verrà decrementato. Inoltre l'utente avrà la possibilità di difendersi sparando dei missili. Il punteggio avrà un valore iniziale uguale a 0 e aumenterà quando il missile, lanciato dal giocatore colpirà l'oggetto nemico e in base al meteorite o alla navicella che verrà schivata dal razzo.

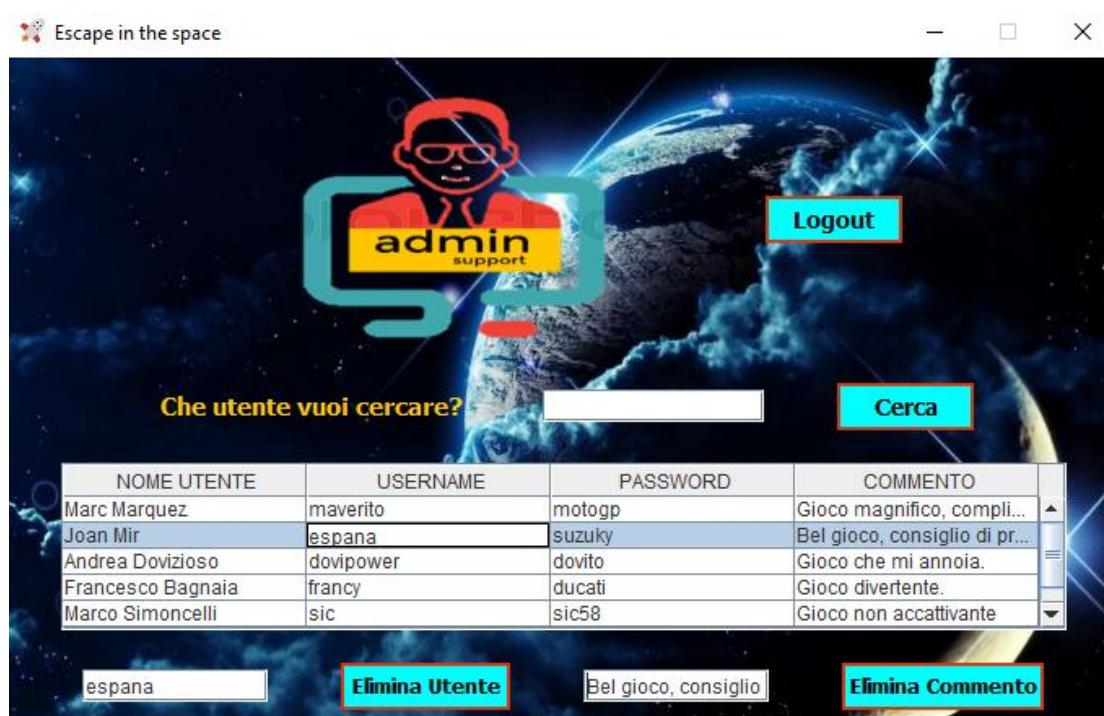


Una volta che il valore della vita raggiungerà lo 0, avremo lo scenario di **GAME OVER**. In tale scenario l'utente potrà visualizzare il punteggio ottenuto e il livello raggiunto.



Una volta cliccato sul tasto **OK** l'utente avrà la possibilità di scegliere se iniziare una nuova partita cliccando sul tasto **YES**, oppure ritornare alla schermata dell'account cliccando sul tasto **NO**.

7.8. Schermata dell'account admin



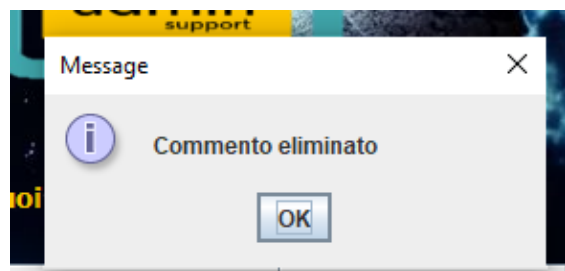
L'admin per entrare nel suo account dovrà inserire nella schermata di login, come username "**AdminGioco**" e come password "**admin**".

Una volta inseriti questi dati apparirà la schermata relativa all'account dell'amministratore, in cui l'admin potrà visualizzare gli utenti registrati al gioco e i loro commenti, cercare un utente inserendo l'username e cliccando sul bottone **CERCA**, eliminare un utente inserendo l'username e cliccando sul bottone **ELIMINA UTENTE**, ed eliminare una recensione inserendo il commento e cliccando sul bottone **ELIMINA COMMENTO**.

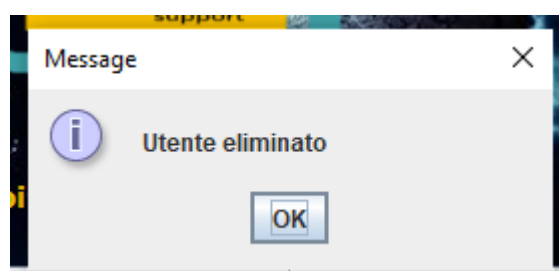
Qui sotto, vediamo degli esempi di come si svolgono le attività descritte sopra.



Esempio di ricerca di un utente



Esempio eliminazione di un commento



Esempio eliminazione di un utente