Colorful Songs

1 Introduzione 3

1.1 Informazioni sul progetto 3

1.2 Abstract 3

1.3 Scopo 3

2 Analisi 4

2.1 Analisi del dominio 4

2.2 Analisi e specifica dei requisiti 4

2.2.1 Spiegazione elementi tabella dei requisiti: 7

2.3 Use case 7

2.4 Pianificazione 7

2.5 Analisi dei mezzi 8

2.5.1 Software 8

2.5.2 Hardware 8

3 Progettazione 8

3.1 Design dell’architettura del sistema 8

3.2 Design dei dati e database 9

3.3 Design delle interfacce 10

3.3.1 Design Login 10

3.3.2 Design Register 11

3.3.3 Design Leaderboard 11

3.4 Design degli Enigmi e ostacoli 12

3.4.1 Percorso con pedane finte 12

3.4.2 Percorso con pedana invisibile e soluzione sul muro 1

3.4.3 Portali finti 1

3.4.4 Percorso con cannoni 1

3.4.5 Percorso con parkour 1

3.5 Generazione dei Dungeon 2

3.6 Design procedurale 2

4 Implementazione 3

5 Test 3

5.1 Protocollo di test 3

5.2 Risultati test 4

5.3 Mancanze/limitazioni conosciute 4

6 Consuntivo 4

7 Conclusioni 4

7.1 Sviluppi futuri 4

7.2 Considerazioni personali 4

8 Glossario 4

9 Indice delle figure 4

10 Bibliografia 5

10.1 Bibliografia per articoli di riviste: 5

10.2 Bibliografia per libri 5

10.3 Sitografia 5

11 Allegati 5

# Introduzione

## Informazioni sul progetto

* Allievi: Lorenzo Berther, Sebastiano De Bertoldi, Simone Demarchi, Kamil Siddiqui
* Docente responsabile: Michel Palucci
* Scuola: Centro Professionale Tecnico Trevano, sezione informatica
* Data di inizio: 29.01.2025
* Data di Termine: 28.05.2025

## Abstract

È una breve e accurata rappresentazione dei contenuti di un documento, senza notazioni critiche o valutazioni. Lo scopo di un abstract efficace dovrebbe essere quello di far conoscere all’utente il contenuto di base di un documento e metterlo nella condizione di decidere se risponde ai suoi interessi e se è opportuno il ricorso al documento originale.

Può contenere alcuni o tutti gli elementi seguenti:

* **Background/Situazione iniziale**
* **Descrizione del problema e motivazione**: Che problema ho cercato di risolvere? Questa sezione dovrebbe includere l'importanza del vostro lavoro, la difficoltà dell'area e l'effetto che potrebbe avere se portato a termine con successo.
* **Approccio/Metodi**: Come ho ottenuto dei progressi? Come ho risolto il problema (tecniche…)? Quale è stata l’entità del mio lavoro? Che fattori importanti controllo, ignoro o misuro?
* **Risultati**: Quale è la risposta? Quali sono i risultati? Quanto è più veloce, più sicuro, più economico o in qualche altro aspetto migliore di altri prodotti/soluzioni?

Esempio di abstract:

*As the size and complexity of today’s most modern computer chips increase, new techniques must be developed to effectively design and create Very Large-Scale Integration chips quickly. For this project, a new type of hardware compiler is created. This hardware compiler will read a C++ program, and physically design a suitable microprocessor intended for running that specific program. With this new and powerful compiler, it is possible to design anything from a small adder, to a microprocessor with millions of transistors. Designing new computer chips, such as the Pentium 4, can require dozens of engineers and months of time. With the help of this compiler, a single person could design such a large-scale microprocessor in just weeks.*

## Scopo

Lo scopo del progetto (scopi didattici/scopi operativi). Dovrebbe descrivere il mandato, ma non vanno ricopiate le informazioni del quaderno dei compiti (che va invece allegato).

# Analisi

## Analisi del dominio

Il videogioco è destinato a qualunque fascia d’età, richiede solamente di registrarsi creando un account.  
Sarà giocabile da PC con tastiera o controller. L’obbiettivo principale è quello di creare un puzzle-game in Unity…

## Analisi e specifica dei requisiti

|  |  |
| --- | --- |
| **ID: REQ-001** | |
| **Nome** | Gestione utente |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Sotto requisiti** | |
| **001** | Creare Database ‘ColorfulSongs’ |
| **002** | Creare le tabelle ‘User’ e ‘Friend’ |

|  |  |
| --- | --- |
| **ID: REQ-002** | |
| **Nome** | Effettuare un login |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | Dipende dal requisito REQ-001 (Gestione Utente) |
| **Sotto requisiti** | |
| **001** | Dovrà essere possibile eliminare gli utenti |
| **002** | Deve spuntare una tastiera virtuale (per controller) |

|  |  |
| --- | --- |
| **ID: REQ-003** | |
| **Nome** | Effettuare una registrazione |
| **Priorità ID: REQ-003** | 1 |
| **Versione** | 1.0 |
| **Note** | Dipende dal requisito REQ-001 (Gestione Utente) |
| **Sotto requisiti** | |
| **001** | Controlli per non creare utenti identici a quelli già esistenti |
| **002** | Deve spuntare una tastiera virtuale (per controller) |

|  |  |
| --- | --- |
| **ID: REQ-004** | |
| **Nome** | Leaderboard web |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | Dipende dal requisito REQ-001 (Gestione utente) |
| **Sotto requisiti** | |
| **001** | Creare la tabella ‘Leaderboard’ |
| **002** | Creare una pagina web che prenda gli highscore degli utenti |
| **003** | Nel gioco deve esserci un collegamento alla pagina |

|  |  |
| --- | --- |
| **ID: REQ-005** | |
| **Nome** | Login nella Leaderboard Web |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | Dipende dal requisito REQ-001 (Gestione utente) |
| **Sotto requisiti** | |
| **001** | Creare una pagina di login |

|  |  |
| --- | --- |
| **ID: REQ-006** | |
| **Nome** | Creare gli enigmi |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Sotto requisiti** | |
| **001** | Creare il “[Percorso con pedane finte](#_Percorso_con_pedane)” |
| **002** | Creare il “[Percorso con pedana invisibile e soluzione sul muro](#_Percorso_con_pedana)” |
| **003** | Creare i “[Portali finti](#_Portali_finti)” |
| **004** | Creare il “[Percorso con cannoni](#_Cannoni)” |
| **005** | Creare il “[Percorso con parkour](#_Parkour)” |

|  |  |
| --- | --- |
| **ID: REQ-007** | |
| **Nome** | Creazione del dungeon casuale |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Sotto requisiti** | |
| **001** | Selezionare le stanze pre-create casualmente, inserendo all’interno di esse un puzzle tra quelli pre-creati |
| **002** | Creare id per il dungeon |
| **003** | Generare casualmente gli NPC che proteggono gli oggetti |

|  |  |
| --- | --- |
| **ID: REQ-008** | |
| **Nome** | Salvataggio highscore |
| **Priorità** | 1 |
| **Versione** | 1.0 |

|  |  |
| --- | --- |
| **ID: REQ-009** | |
| **Nome** | Tutorial testuale |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | Creare una pagina con la descrizione dei comandi |

|  |  |
| --- | --- |
| **ID: REQ-010** | |
| **Nome** | Aggiungere amici |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | Dipende dal requisito REQ-001 (Creazione DB) |

|  |  |
| --- | --- |
| **ID: REQ-011** | |
| **Nome** | Giocare con tastiera e controller |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | In caso di controller, generare una tastiera virtuale, per soddisfare requisito REQ-002 e REQ-003 |

|  |  |
| --- | --- |
| **ID: REQ-012** | |
| **Nome** | Personalizzazione skin |
| **Priorità** | 2 |
| **Versione** | 1.0 |
| **Sotto requisiti** | |
| **001** | Creare uno shop in cui acquistare le skin |
| **002** | A fine del dungeon dare delle monete |

|  |  |
| --- | --- |
| **ID: REQ-013** | |
| **Nome** | Tutorial interattivo |
| **Priorità** | 3 |
| **Versione** | 1.0 |

### Spiegazione elementi tabella dei requisiti:

**ID**: identificativo univoco del requisito

**Nome**: breve descrizione del requisito

**Priorità**: indica l’importanza di un requisito nell’insieme del progetto, definita assieme al committente. Ad esempio, poter disporre di report con colonne di colori diversi ha priorità minore rispetto al fatto di avere un database con gli elementi al suo interno. Solitamente si definiscono al massimo di 2-3 livelli di priorità.

**Versione**: indica la versione del requisito. Ogni modifica del requisito avrà una versione aggiornata.

Sulla documentazione apparirà solamente l’ultima versione, mentre le vecchie dovranno essere inserite nei diari.

**Note**: eventuali osservazioni importanti o riferimenti ad altri requisiti.

**Sotto requisiti**: elementi che compongono il requisito.

## Use case

I casi d’uso rappresentano l’interazione tra i vari attori e le funzionalità del prodotto.

## Pianificazione

Prima di stabilire una pianificazione bisogna avere almeno una vaga idea del modello di sviluppo che si intende adottare. In questa sezione bisognerà inserire il modello concettuale di sviluppo che si seguirà durante il progetto. Gli elementi di riferimento per una buona pianificazione derivano da una scomposizione top-down della problematica del progetto.

La pianificazione può essere rappresentata mediante un diagramma di Gantt:

|  |
| --- |
| Figura : Esempio di diagramma di Gantt. |

Se si usano altri metodi di pianificazione (p.es. scrum), dovranno apparire in questo capitolo.

## Analisi dei mezzi

### Software

* Unity 6
* MS-Office
* MS-Project
* GitHub (Desktop)
* PHP Storm 2024.1.1
* MySQL Workbench 8.0
* Bootstarp 5.2.3
* Figma

### Hardware

PC forniti dalla scuola e Server scolastico dove mettere il progetto e il Database

# Progettazione

Questo capitolo descrive esaustivamente come deve essere realizzato il prodotto fin nei suoi dettagli. Una buona progettazione permette all’esecutore di evitare fraintendimenti e imprecisioni nell’implementazione del prodotto.

## Design dell’architettura del sistema

## Design dei dati e database

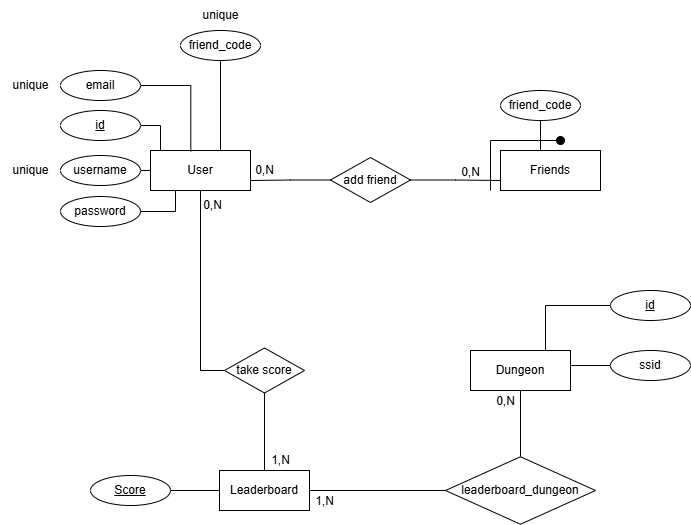


Figura : Schema E-R



Figura : Schema logico

## Design delle interfacce

### Design Login



Figura : Design interfaccia di Login

### Design Register



Figura : Design interfaccia di Registrazione

### Design Leaderboard



Figura Design interfaccia leaderboard

#### Design della mappa



Figura : Design di una stanza

## Design degli Enigmi e ostacoli

### Percorso con pedane finte

In questo percorso il giocatore dovrà percorrere un percorso lungo 4 pedane, dove alcune pedane saranno finte. Se l’utente ci passerà sopra, verrà riportato all’inizio della stanza, perdendo eventuali oggetti che possedeva, come la chiave per la porta d’uscita, e la pedana sparirà, lasciando un buco.

Questo percorso occuperebbe uno spazio di colore rosso sulla mappa.



Figura : Design del percorso con pedane finte



Figura : Design della pedana (Come apparirebbe all'utente dopo qualche tentativo)

### Percorso con pedana invisibile e soluzione sul muro

Questo percorso è simile all’enigma “Percorso con pedane finte”, ma a sua differenza le pedane sono invisibili e il percorso da percorrere è disegnato sul muro, qua se si sbaglia e si passa su una pedana inesistente si perderà la partita.

Questo percorso occuperebbe uno spazio di colore rosso sulla mappa.

Ecco un esempio di come un’ipotetica pedana potrebbe essere e come potrebbe venir generata:

La parte rossa è bianca sono invisibili/trasparenti. Sul muro grigio vediamo la strada corretta da percorrere (bianca).



Figura : Design dell’enigma “Percorso con pedana invisibile e soluzione sul muro”



Figura : Design dell'enigma "Percorso con pedana invisibile e soluzione sul muro" visto dall'utente

### Portali finti

Nella stanza saranno presenti dei portali, solo uno di questi porterà il player dalla parte opposta del percorso. Se l’utente prende il portale sbagliato perderà la partita, per riconoscere il portale giusto bisognerà mettersi davanti il portale e ascoltare l’audio, che sarà diverso per quello giusto.

Se il player ha problemi d’udito, nelle impostazioni potrà attivare il fatto che questo enigma non gli appare.



Figura : Design enigma dei portali



Figura : Design dell'enigma dei portali visto dall'utente

### Percorso con cannoni

Questo percorso avrà sui muri dei cannoni che spareranno a intervalli regolari, alternandosi tra i cannoni in posizione pari con quelli in posizione dispari, se si verrà colpiti da cannoni si perderà la partita

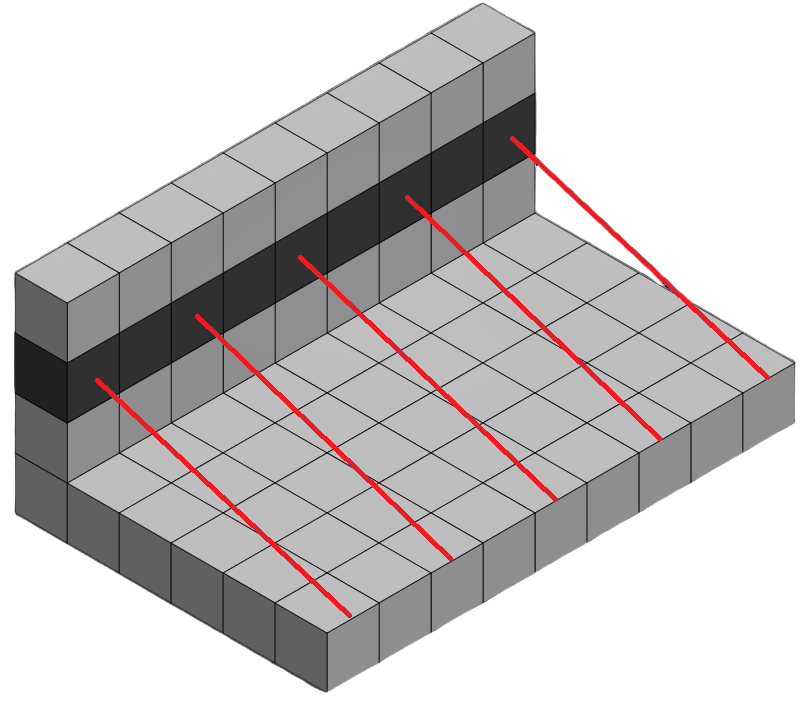


Figura : Cannoni in posizione pari sparano

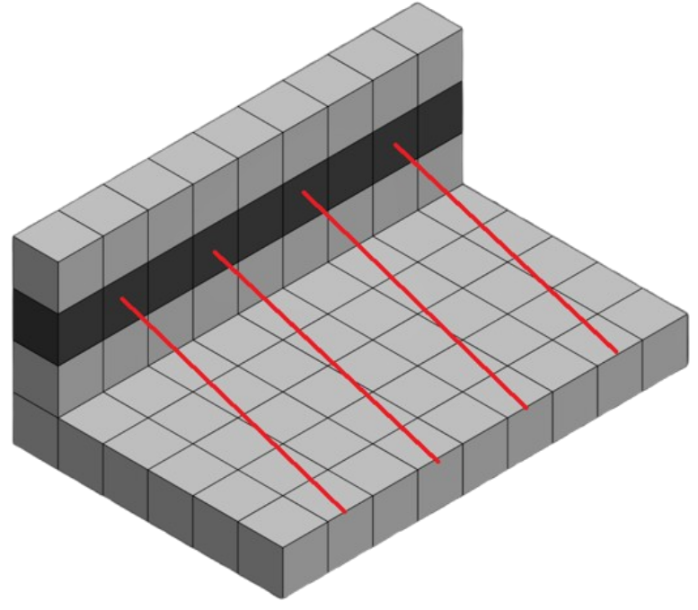


Figura : Cannoni in posizione dispari sparano

### 

### Percorso con parkour

Questo percorso sarà un semplice parkour, l’utente dovrà superarlo usando la meccanica del salto. Se cadrà perderà la partita.

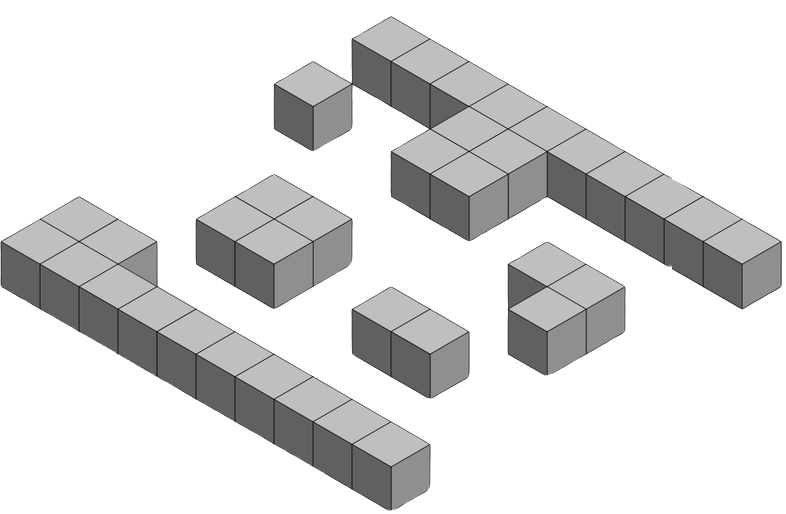


Figura : Percorso parkour

## Generazione dei Dungeon

I dungeon sono composti da cinque stanze, ognuna contenenti due enigmi.

Per generare le stanze useremo una stanza vuota come base e aggiungeremo due enigmi casualmente. Bisognerà quindi assegnare degli id alle stanze vuote e agli enigmi, così da poter usare un generatore di numeri per selezionare stanze ed enigmi. Questo renderà facile generare gli seed dei dungeon, che saranno composti dall’id della stanza e gli id egli enigmi, un esempio di seed di un dungeon potrebbe essere:  
“115-424-322-112-552”, quindi la stanza con id=1 aveva al suo interno gli enigmi con id=1 e id=5.

## Design procedurale

Descrive i concetti dettagliati dell’architettura/sviluppo utilizzando ad esempio:

* Diagrammi di flusso e Nassi.
* Tabelle.
* Classi e metodi.
* Tabelle di routing
* Diritti di accesso a condivisioni …

Questi documenti permetteranno di rappresentare i dettagli procedurali per la realizzazione del prodotto.

# Implementazione

In questo capitolo dovrà essere mostrato come è stato realizzato il lavoro. Questa parte può differenziarsi dalla progettazione in quanto il risultato ottenuto non per forza può essere come era stato progettato.

Sulla base di queste informazioni il lavoro svolto dovrà essere riproducibile.

In questa parte è richiesto l’inserimento di codice sorgente - Print Screen - di maschere solamente per quei passaggi particolarmente significativi e/o critici.

Inoltre, dovranno essere descritte eventuali varianti di soluzione o scelte di prodotti con motivazione delle scelte.

Non deve apparire nessuna forma di guida d’uso di librerie o di componenti utilizzati. Eventualmente questa va allegata.

Per eventuali dettagli si possono inserire riferimenti ai diari.

## Generazione della stanza

**AGGIUNGERE SPIEGAZIONE SCRIPT SUL BLOCCO (AMBIENTE UNITY)?**

Per generare la stanza abbiamo lasciato molti valori da serializzare da unity, così da lasciare una grande libertà per la generazione della stanza:

|  |
| --- |
| private int x;      private int z;      private int h;      private bool empty;      //Valori per la lunghezza, larghezza e l'altezza da dove generare la stanza      [SerializeField] private int row;      [SerializeField] private int column;      [SerializeField] private int y;      //Valori per la X e la Z da dove iniziare a generare la stanza e l'altezza  dei muri      [SerializeField] private int startingX;      [SerializeField] private int startingZ;      [SerializeField] private int wallHeight;      //Valori per la X di dove inizia l'enigma (dove cominciare a lasciare spazio  vuoto) e per quanto si espande      [SerializeField] private int xEnigma;      [SerializeField] private int lEnigma;      //Oggetti che contengono i blocchi per generare il pavimento e il muro      // e per contenere i blocchi generati, così da avere tutto più ordinato      [SerializeField] private GameObject floor;      [SerializeField] private GameObject wall;      [SerializeField] private GameObject parent; |

**METTERE FOTO DEL MENU DI UNITY DOVE INSERIRE LE VARIABILI???**

Dopodichè nel “void start(){…”, tramite un ciclo for annidato per gestire X e Z, genero i blocchi per il pavimento, controllando se la X e la Z non corrispondo a quelle dove è presente l’enigma, e quindi bisogna lasciar vuoto.  
Controllo anche quando la X o la Z sono uguali ai valori massimi (row e column), se uno dei due è uguale vuol dire che devo generare i muri, quindi in un ciclo for genero i blocchi del muro.

|  |
| --- |
| for (int r = startingX; r <= row; r++)          {              x = r;              //Controllo se sono nelle zone dove generare il pavimento,              //Quindi prima del xEnigma, e dopo xEnigma+lEnigma.              //Se xEnigma==0 vuol dire che non devo lasciare spazio vuoto              if ((r < xEnigma || r >= xEnigma + lEnigma) || xEnigma==0)              {                  empty = false;              }              else              {                  empty = true;              }              for (int c = startingZ; c <= column; c++)              {                  z = c;                  if (!empty)                  {                      string name = "pavimento[" + r.ToString() + ";" +  y.ToString() + ";" + c.ToString() + "]";                      var cube = Instantiate(floor, new Vector3(x, y, z),  Quaternion.identity);                      cube.name = name;                      cube.transform.SetParent(parent.transform);                  }                  if (r == row || c == column)                  {                      for (int a = y; a <= wallHeight; a++)                      {                          string nameWall = "wall[" + r.ToString() + ";" +  a.ToString() + ";" + c.ToString() + "]";                          var wallObj = Instantiate(wall, new Vector3(x, a, z),  Quaternion.identity);                          wallObj.name = nameWall;                          wallObj.transform.SetParent(parent.transform);                      }                  }              }          } |

La variabile name serve per dare un nome sensato ai blocchi generati

**FOTO DEL NOME DEI BLOCCHI???**

# Test

## Protocollo di test

Definire in modo accurato tutti i test che devono essere realizzati per garantire l’adempimento delle richieste formulate nei requisiti. I test fungono da garanzia di qualità del prodotto. Ogni test deve essere ripetibile alle stesse condizioni.

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-001  REQ-012 | **Nome:** | Import a card with KIC, KID and KIK keys, but not shown with the GUI |
| **Descrizione:** | Import a card with KIC, KID and KIK keys with no obfuscation, but not shown with the GUI | | |
| **Prerequisiti:** | Store on local PC: Profile\_1.2.001.xml (appendix n\_n) and Cards\_1.2.001.txt (appendix n\_n).  PIN (OTA\_VIEW\_PIN\_PUK\_KEY) and ADM (OTA\_VIEW\_ADM\_KEY) user right not set. | | |
| **Procedura:** | 1. Go to “Cards manager” menu,  in main page click “Import Profiles” link, Select the “1.2.001.xml” file, Import the Profile 2. Go to “Cards manager” menu,  in main page click “Import Cards” link, Select the “1.2.001.txt” file, Delete the cards,  Select the “1.2.001.txt” file, Import the cards 3. Research the “41795924770” Card, Click the imsi card link Check the card details 4. Execute the SQL: SELECT imsi, dir, keyset, cntr, rawtohex(kickey), rawtohex(kidkey), rawtohex(kikkey), rawtohex(chv), rawtohex(dap)FROM otacardkey a where imsi='340041795924770' ORDER BY keyset; | | |
| **Risultati attesi:** | Keys visible in the DB (OtaCardKey) but not visible in the GUI (Card details) | | |

## Risultati test

Tabella riassuntiva in cui si inseriscono i test riusciti e non del prodotto finale. Se un test non riesce e viene corretto l’errore, questo dovrà risultare nel documento finale come riuscito (la procedura della correzione apparirà nel diario), altrimenti dovrà essere descritto l’errore con eventuali ipotesi di correzione.

## Mancanze/limitazioni conosciute

Descrizione con motivazione di eventuali elementi mancanti o non completamente implementati, al di fuori dei test case. Non devono essere riportati gli errori e i problemi riscontrati e poi risolti durante il progetto.

# Consuntivo

Consuntivo del tempo di lavoro effettivo e considerazioni riguardo le differenze rispetto alla pianificazione (cap. 1.7) (ad esempio Gantt consuntivo).

# Conclusioni

Quali sono le implicazioni della mia soluzione? Che impatto avrà? Cambierà il mondo? È un successo importante? È solo un’aggiunta marginale o è semplicemente servita per scoprire che questo percorso è stato una perdita di tempo? I risultati ottenuti sono generali, facilmente generalizzabili o sono specifici di un caso particolare? ecc.

## Sviluppi futuri

Migliorie o estensioni che possono essere sviluppate sul prodotto.

## Considerazioni personali

Cosa ho imparato in questo progetto? ecc.

# Glossario

|  |  |
| --- | --- |
| **Termine** | **Descrizione** |
| Godot Engine | Engine per sviluppare videogiochi 2D/3D in GDscript |

# Indice delle figure

[Figura 1: Esempio di diagramma di Gantt. 7](#_Toc188452039)

[Figura 2: Schema E-R 9](#_Toc188452040)

[Figura 3: Schema logico 9](#_Toc188452041)

[Figura 4: Design interfaccia di Login 10](#_Toc188452042)

[Figura 5: Design interfaccia di Registrazione 11](#_Toc188452043)

[Figura 6: Design di una stanza 11](#_Toc188452044)

[Figura 7: Design del percorso con pedane finte 12](#_Toc188452045)

[Figura 8: Design dell’enigma “Percorso con pedana invisibile e soluzione sul muro” 13](#_Toc188452046)

# Bibliografia

## Bibliografia per articoli di riviste:

1. Cognome e nome (o iniziali) dell’autore o degli autori, o nome dell’organizzazione,
2. Titolo dell’articolo (tra virgolette),
3. Titolo della rivista (in italico),
4. Anno e numero
5. Pagina iniziale dell’articolo.

## Bibliografia per libri

1. Cognome e nome (o iniziali) dell’autore o degli autori, o nome dell’organizzazione,
2. Titolo del libro (in italico),
3. ev. Numero di edizione,
4. Nome dell’editore,
5. Anno di pubblicazione,
6. ISBN.

## Sitografia

* URL del sito (se troppo lungo solo dominio, evt completo nel diario),
* Eventuale titolo della pagina (in italico),
* Data di consultazione (GG-MM-AAAA)
* <https://www.youtube.com/watch?v=dclc8w6JW7Y>, *Tutorial Godot ‘ThinkWithGames’,* 15-01-2025
* <https://www.youtube.com/watch?v=8HvcHtauKoc>, *Tutorial Collision* *‘ThinkWithGames’*, 15-01-2025

**Esempio:**

* http://standards.ieee.org/guides/style/section7.html, *IEEE Standards Style Manual*, 07-06-2008.

# Allegati

Elenco degli allegati, esempio:

* Diari di lavoro
* Codici sorgente/documentazione macchine virtuali
* Istruzioni di installazione del prodotto (con credenziali di accesso) e/o di eventuali prodotti terzi
* Documentazione di prodotti di terzi
* Eventuali guide utente / Manuali di utilizzo
* Mandato e/o QdC
* Prodotto
* …