Documentazione Gava

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: 5

2.3 Use case 5

2.4 Pianificazione 5

2.5 Analisi dei mezzi 5

2.5.1 Software 6

2.5.2 Hardware 6

3 Progettazione 6

3.1 Design dell’architettura del sistema 6

3.2 Design dei dati e database 6

3.3 Design delle interfacce 6

3.4 Design procedurale 6

4 Implementazione 7

5 Test 7

5.1 Protocollo di test 7

5.2 Risultati test 8

5.3 Mancanze/limitazioni conosciute 8

6 Consuntivo 8

7 Conclusioni 8

7.1 Sviluppi futuri 8

7.2 Considerazioni personali 8

8 Glossario 8

9 Bibliografia 9

9.1 Bibliografia per articoli di riviste: 9

9.2 Bibliografia per libri 9

9.3 Sitografia 9

10 Allegati 9

# Introduzione

## Informazioni sul progetto

In questo capitolo raccogliere le informazioni relative al progetto, ad esempio:

* Allievo: Mirco Gava
* SAM Trevano I3BB – Labo progetti
* Inizio: 12.09.2025 – Fine: 19:12.2025

## Abstract

I decided to do this project because I wanted to get better at programming with objects. I wanted an interface to see, and so I decided that .NET MAUI was my best option.  
The idea for this project was born simply because I’m a single person and I had limited time to finish it. I wanted to work on something that I like, and this is why I’m doing something with music.  
I’m trying to solve a problem that I have with Spotify: I can see my music stats only once a year, so I decided to solve this problem by creating an application that allows you to see your stats whenever you want. I want to make a better version of Apple Music that solves the problem of stats by letting you see them whenever you want, but their interface is not very simple and practical to use.

## Scopo

Lo scopo del progetto sta per me nell’allenarmi con la programmazione a classi: all’inizio la mia idea era di farlo su java, ma la mancanza di un’interfaccia mi limitava un po’ nella costruzione del progetto, così ho deciso di lavorare su maui essendo che in passato ho avuto difficoltà in quel linguaggio ed è una buona occasione per rinforzare le mie conoscenze.

Ho scelto di fare l’organizzatore di playlist perché volevo fare un progetto con un tema che mi piace e che non fosse impossibile da fare con il tempo limitato e le mie attuali conoscenze: così ho deciso di fare qualcosa che riguardasse la musica. La mia prima idea era di fare un quiz, ma essendo che era un po’ banale e diverse persone avevano già fatto qualcosa di simile solo con temi diversi, ispirandomi ad apple music e il loro algoritmo per vedere brani, artisti, generi preferiti ho deciso di fare questo progetto.

# Analisi

## Analisi del dominio

Il mio progetto non è nulla di innovativo: esistono già diverse applicazioni per creare playlist o per vedere le tue statistiche (ore di musica ascoltata, brani/artisti preferiti, …), che posso sfruttare per prendere spunto e vedere cosa voglio cambiare.

Potrebbe essere utilizzato da diverse persone e fasce di età: principalmente però il range dell’età penso sia tra i 12-30 anni. Le persone che la userebbero sono principalmente gli utenti di Spotify o YouTube music: possono creare le playlist ma non hanno modo di vedere all’intero della loro app le statistiche quando vogliono: così sono obbligati a usare tools esterni per vederle.

Il tool dovrà funzionare su pc e telefono, che sono i due modi principali dove la gente ascolta musica

## Analisi e specifica dei requisiti

I bisogni del committente sono di avere un applicativo facile da capire e da usare, in modo da rendere l’esperienza gradevole e non far passare gli utenti ad un altro tool. Il tool dovrà essere in grado di poter dire all’utente le sue statistiche, in modo semplice e chiaro.

L’utente dovrà essere in grando di aggiungere i brani che vuole alla sua playlist e di poterla denominare come vuole.

La sicurezza del prodotto può essere bassa, anche se ci fosse una fuga di dati non ci saranno dati sensibili ma solo musica all’interno dell’applicativo.

|  |  |
| --- | --- |
| **ID: REQ-01** | |
| **Nome** | Organizzatore playlist |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** |  |
| **Sotto requisiti** | |
| **001** | Si necessità di un’interfaccia pulita e facile da capire |
| **002** | Si dovranno poter aggiungere nuovi brani: assieme ad essi il genere, artista e album |
| **003** | Dovrà essere possibile la ricerca di brani, artisti, album e generi |

### Spiegazione elementi tabella dei requisiti:

**ID**: identificativo univoco del requisito

**Nome**: Si chiama organizzatore perché ti permette di ordinare i tuoi brani e vedere le tue statistiche

**Priorità**: I requisiti marcati sono le mie priorità assolute: la cosa più importante è avere un’interfaccia pulita e funzionante: quindi la prima cosa che farò è creare il sistema delle playlist e la possibilità di poter aggiungere brani (livello 1): poi pulirò l’interfaccia e aggiungerò le statistiche (livello 2) e poi penserò alla ricerca dei brani e alla pulizia finale di tutta l’interfaccia (livello 3). Quando avrò finito con questo potrò pensare agli extra

**Versione**: Questa è la prima versione dei requisiti, potrebbero cambiare più in avanti

**Note**: Per ora non ci sono note

**Sotto requisiti**: Requisiti più importanti che avranno la priorità nello sviluppo dell’applicativo

## Use case

Ci sarà un solo attore (l’utente), verrà usato per memorizzare brani (inseriti dall’utente) e dividerli in playlist, in questo modo l’utente potrà facilmente ritrovare la sua musica.

Un altro uso è che l’utente potrà vedere artisti, album e generi preferiti in base a quanti ne ha salvati (es: se salva 3 canzoni di artista1 e solo 2 di artista2 il programma dirà che artista1 è il suo preferito).

## 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 1: Esempio di diagramma di Gantt. |

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

## Analisi dei mezzi

Ho a disposizione vecchi progetti ed internet

### Software

.NET MAUI – visual studio 2022

### Hardware

Pc dell:

* Scheda grafica: 4GB
* RAM: 32GB
* Processore: 12th Gen Intel(R) Core(TM) i7-12700

# 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

Descrive:

* La struttura del programma/sistema lo schema di rete...
* Gli oggetti/moduli/componenti che lo compongono.
* I flussi di informazione in ingresso ed in uscita e le relative elaborazioni. Può utilizzare *diagrammi di flusso dei dati* (DFD).
* Eventuale sitemap

## Design dei dati e database

Descrizione delle strutture di dati utilizzate dal programma in base agli attributi e le relazioni degli oggetti in uso.

Schema E-R, schema logico e descrizione.

Se il diagramma E-R viene modificato, sulla doc dovrà apparire l’ultima versione, mentre le vecchie saranno sui diari.

## Design delle interfacce

Descrizione delle interfacce interne ed esterne del sistema e dell’interfaccia utente. La progettazione delle interfacce è basata sulle informazioni ricavate durante la fase di analisi e realizzata tramite mockups.

## 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.

# 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

Inserite una semplice tabella con due colonne che spieghi i termini specifici del progetto (lista dei termini in ordine alfabetico A-Z)

Esempio:

|  |  |
| --- | --- |
| **Termine** | **Descrizione** |
| AJAX | **Asynchronous JavaScript And XML**: una tecnica che permette di eseguire richieste ed ottenere dati da una pagina web in modo asincrono. |
| CSS | **Cascading Style Sheets**: linguaggio che permette di definire il layout e la grafica di una pagina web. |

# 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

1. URL del sito (se troppo lungo solo dominio, evt completo nel diario),
2. Eventuale titolo della pagina (in italico),
3. Data di consultazione (GG-MM-AAAA).

**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
* …