Collezione di Videogiochi

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

## Abstract

*Often people would like to play something they haven’t played yet but have troubles finding the right fit for what they like or the platforms they use to look for the game are platform dependent. With this project you’ll be able to look for games independent to the platform and also get tips based on the games you like more.*

## Scopo

Lo scopo didattico del progetto è imparare a gestire un progetto IT. Lo scopo operativo riguarda invece la creazione di un applicativo che permette di fare ricerche su un catalogo di videogiochi e sulle sue informazioni, inoltre l’utente potrà decidere di aggiungere un gioco alla sua lista dei preferiti e ottenere così dei consigli simili ai giochi scelti dall’utente.

# Analisi

## Analisi del dominio

L’applicativo verrà utilizzato principalmente da appassionati di videogiochi per poter eseguire ricerche sui videogiochi e per poter ricevere consigli sui giochi basandosi su quelli già giocati, questo unito ai filtri e alla funzione di ricerca permetterà di esplorare la scelta di giochi senza vincoli dettati dalla piattaforma utilizzata o dall’anno di pubblicazione.

## Analisi e specifica dei requisiti

|  |  |
| --- | --- |
| **ID: REQ-01** | |
| **Nome** | Accesso API MobyGames |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | - |
| **Sotto requisiti** | |
| **001** | È necessario creare un account sul sito. |

|  |  |
| --- | --- |
| **ID: REQ-02** | |
| **Nome** | Creazione della maschera per la home |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | Verrà usata per la ricerca e il filtraggio dei giochi. |

|  |  |
| --- | --- |
| **ID: REQ-03** | |
| **Nome** | Creazione del Database degli utenti. |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** |  |

|  |  |
| --- | --- |
| **ID: REQ-04** | |
| **Nome** | Accesso al database tramite il software |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** |  |
| **ID: REQ-05** | |
| **Nome** | Creazione della maschera per la scheda informativa |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** |  |

|  |  |
| --- | --- |
| **ID: REQ-06** | |
| **Nome** | Sistema di rating dei giochi |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** |  |

|  |  |
| --- | --- |
| **ID: REQ-07** | |
| **Nome** | Filtri per le ricerche dei giochi |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** |  |

|  |  |
| --- | --- |
| **ID: REQ-08** | |
| **Nome** | Creazione di una lista e aggiunta dei giochi |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** |  |

|  |  |
| --- | --- |
| **ID: REQ-09** | |
| **Nome** | Collegamenti nella scheda informativa. |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** |  |

### 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.

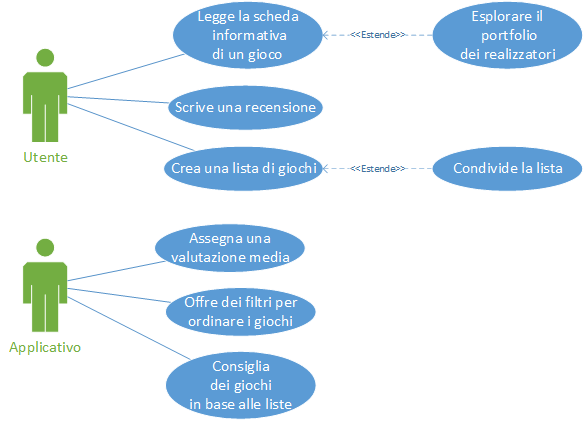
**Versione**: indica la versione del requisito. Ogni modifica del requisito avrà una versione aggiornata.

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

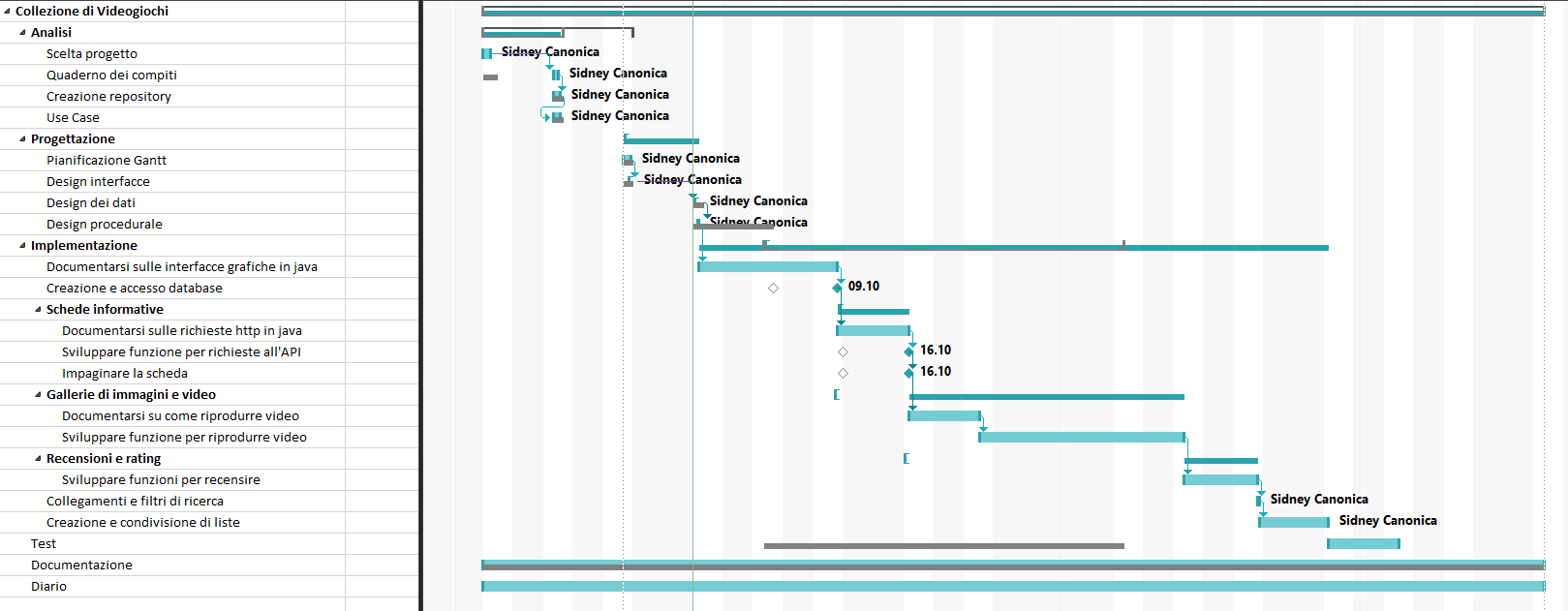
**Note**: eventuali osservazioni importanti o riferimenti ad altri requisiti.

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

## Use case



## Pianificazione



## Analisi dei mezzi

### Software

SDK, librerie, tools utilizzati per la realizzazione del progetto e eventuali dipendenze.

* Java 19.0.2
* MySQL
* NetBeans
* Swing (Java)

La versione di mysql dipende da quella dell’host.

### Hardware

1 PC: Le componenti del PC usato per sviluppare l’applicazione sono

* + Intel i7 -9700
  + 32 GB RAM
  + SSD 512 GB
  + Nvidia Geforce RTX 2060

# Progettazione

## Design dell’architettura del sistema

Il sistema è composto dal software che manda richieste http all’API di mobygames, l’API di mobygames risponde con i dati sui giochi e il software le elabora.

Inoltre il software comunica con un database su un server per gestire le liste e gli utenti.

## Design dei dati e database

Schema Logico:

Utente (username, password, commenti)

Lista (id, nome, utente\_username (FK), utente\_password (FK))

Gioco (titolo)

contiene (lista\_id (FK), gioco\_titolo (FK))

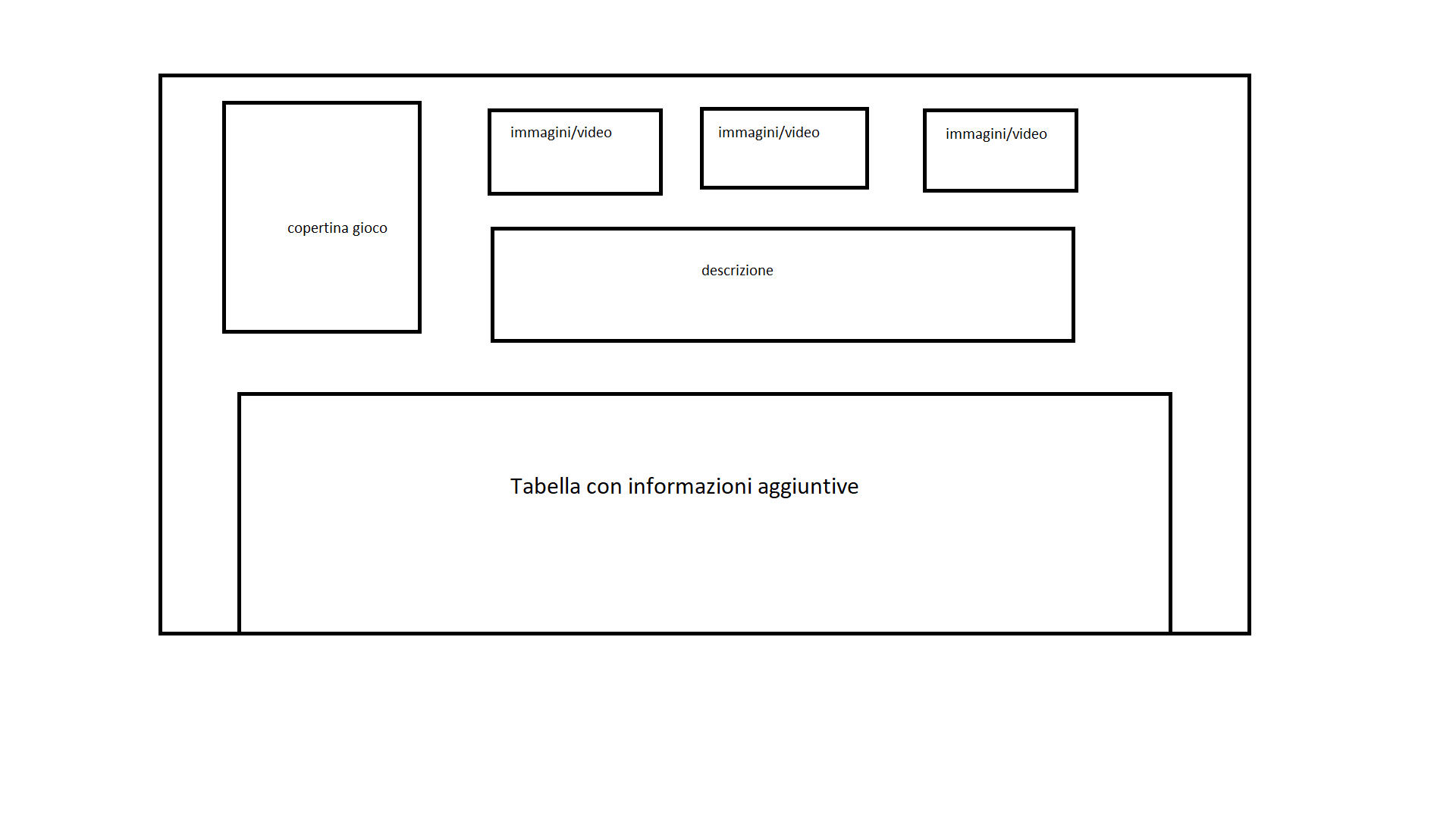
Per il database ho deciso di fare due tabelle principali e una per contenere i nomi dei titoli, la tabella Lista si occupa di salvare le Liste create dagli utenti e i la lista utente si occupa del login. Ho deciso di mettere un id alla lista per poter permettere di avere due liste con lo stesso nome.

## Design delle interfacce

La prima interfaccia è questa, è quella che viene visualizzata quando viene avviato il programma.

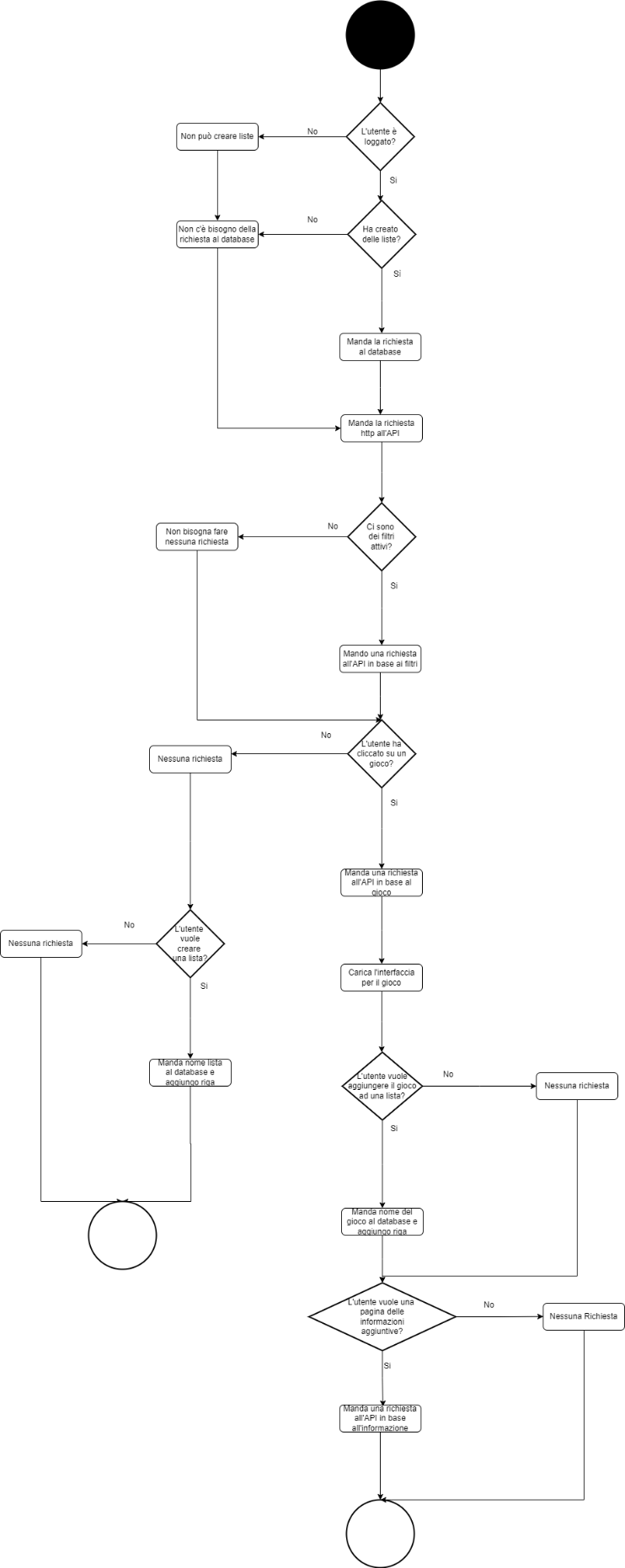
A sinistra c’è un menù con le liste di giochi create, cliccando una lista già creata verranno visualizzati i giochi della lista nella parte centrale del programma, in questa sezione è presente anche un bottone per aggiungere una nuova lista, bisognerà dare un nome alla lista e verrà aggiunta alle altre.

Nella parte centrale c’è la lista dei giochi che inizialmente sarà riempita di tutti i giochi dell’API di mobygames, ma che potrà essere manipolata grazie al menù dei filtri che sarà nella parte sinistra dell’applicazione dove questi giochi potranno essere filtrati.



Questa è l’interfaccia che si ottiene una volta cliccato un gioco, nello spazio in cui ci sono le informazioni aggiuntive ci sarà anche un bottone per aggiungere il gioco ad una delle liste create e cliccando sulle informazioni del gioco sarà possibile sapere di più su cosa hanno fatto le persone che hanno lavorato a quel gioco.

## Design procedurale



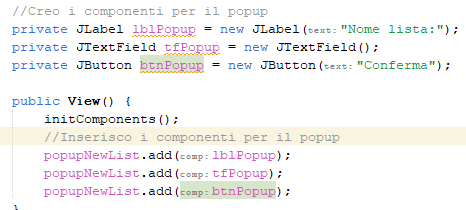
# Implementazione

Per iniziare mi sono documentato sulle librerie per fare applicazioni con interfacce grafiche in java e ho trovato questo sito con molti esempi e codici commentati per capire il funzionamento delle librerie swing e awt.

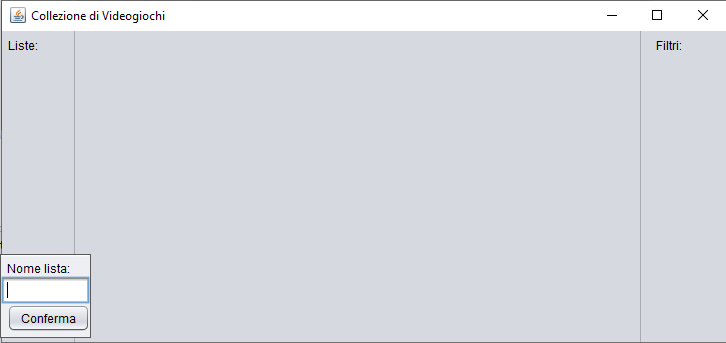
<https://www3.ntu.edu.sg/home/ehchua/programming/java/J4a_GUI.html>

Grazie alla guida ho scoperto che l’editor NetBeans ha integrata la possibilità di creare interfacce grafiche con un menù drag and drop. Quindi lo userò per aiutarmi a creare le interfacce.

<https://www3.ntu.edu.sg/home/ehchua/programming/howto/NetBeans_HowTo.html#netbeans-swing>



Per creare il menù popup che viene mostrato quando si vuole creare una nuova lista ho dovuto creare individualmente le parti dell’interfaccia ed aggiungerle alla finestra popup e quando viene cliccato il bottone per aggiungere una nuova lista rendo visibile la finestra di popup.

Il risultato è questo:

Per applicare le immagini dei giochi nella parte centrale mi sono aiutato con l’interfaccia grafica che permette di aggiungere un’immagine ad una label e andando a guardare nel codice sorgente ho trovato la linea di codice per farlo, questa istruzione mi servirà poi da usare con le immagini ricevute dall’API.

Arrivato a questo punto ho capito che non ha molto senso continuare a lavorare sulle interfacce senza i dati

Da inserire presi dall’API, quindi ho deciso di proseguire creando il database e creare la funzione per le richieste e finire le interfacce una volta che ho la possibilità di fare richieste all’API.

Per il database ho usato mysql tramite la versione 3.3 di xampp installata sui computer scolastici, ho creato il mio utente con il comando:

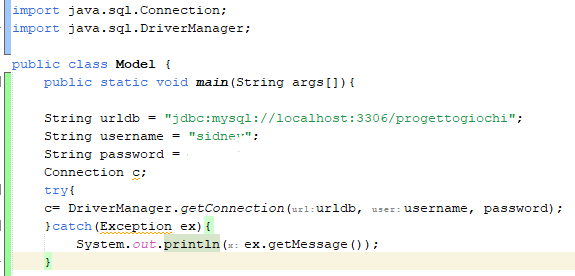
CREATE USER sidney@localhost IDENTIFIED BY ‘password’;

gli ho dato i permessi sul database che uso con

GRANT ALL ON progettogiochi.\* TO 'sidney'@'localhost' IDENTIFIED BY 'password';

e ho scritto in un file i comandi necessari per creare il database in base allo schema logico.

Per creare una connessione al database ho usato la libreria mysql connector per java, ho scaricato la libreria da https://dev.mysql.com/downloads/connector/j/ e l’ho aggiunta tra le librerie del mio progetto di netbeans.

Ecco il codice per la connessione:

Grazie al codice commentato presente in questo sito <https://www3.ntu.edu.sg/home/ehchua/Programming/java/JDBC_Basic.html>

Ho creato uno statement sempre grazie alla libreria mysql connector e sono riuscito ad eseguire una query sul database con dei dati di testing.

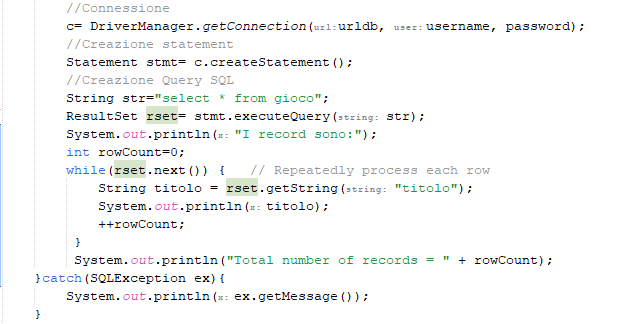


Figura 1 Codice per la query

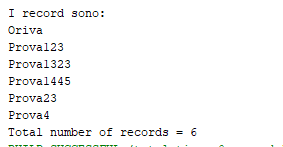
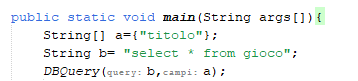
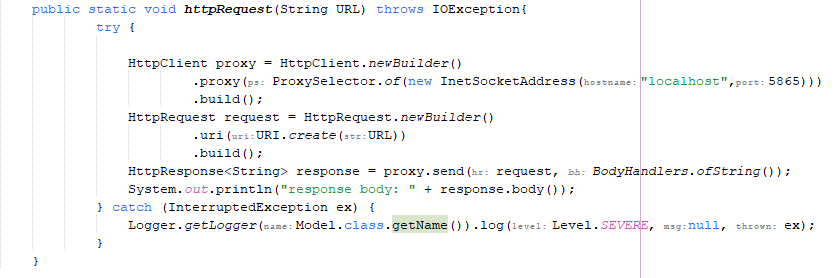
Ho fatto diventare il codice per connettersi al database una funzione chiamata DBQuery dove posso passargli la query da fare e in un array il nome delle colonne da leggere.

Figura 2 Test della funzione DBQuery

Figura 3 Risultato della query

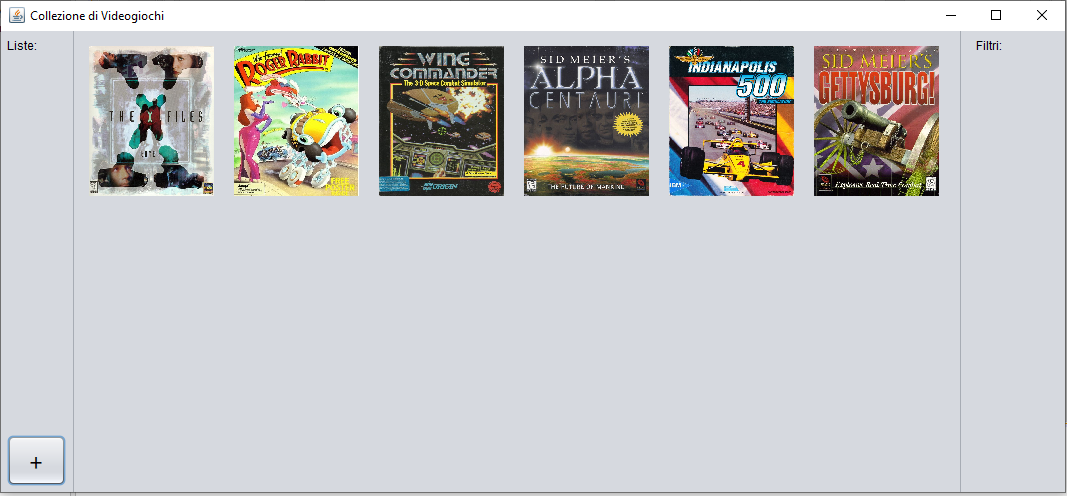
Ho sviluppato la funzione per le richieste http usando le librerie java.net.http.httpClient , httpRequest e httpResponse. Inizialmente ho provato ad usarla in localhost e funzionava, ma se provavo a fare una richiesta all’API dava un errore, dopo un po’ ho capito che il problema potrebbe essere il proxy e ho messo un proxy che manda alla porta di px 5865 per reindirizzare la richiesta e grazie a questo sono riuscito a far funzionare il codice anche per le richieste all’API.

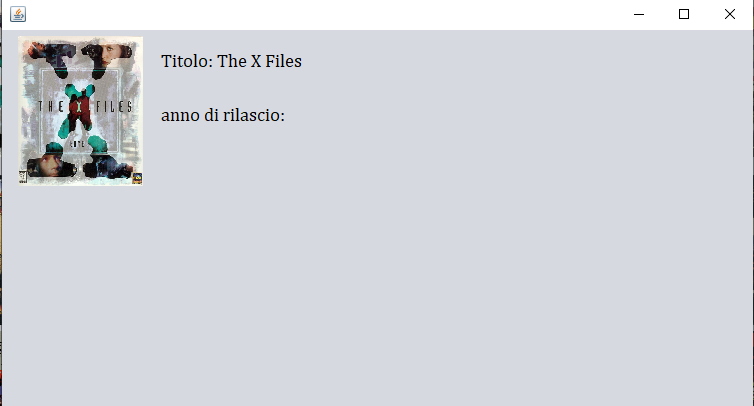
Una volta riuscito a mandare le richieste ho fatto con l’aiuto di ChatGPT per generare l’espressione regolare una funzione che permette di cercare dei campi specifici dai dati ritornati dalla richiesta all’API, è solo per il campo delle immagini che mi serviva ma verrà adattata in modo da trovare qualsiasi dato mi serva:



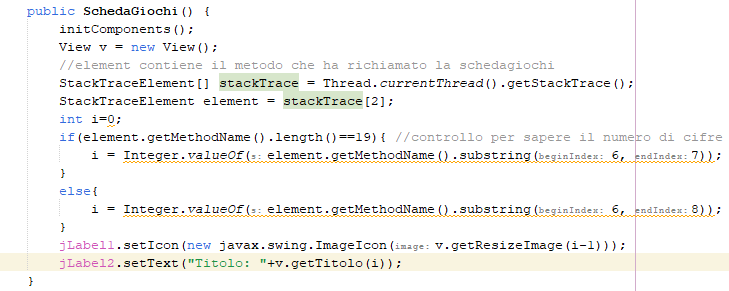
Utilizzando queste due funzioni sono riuscito a scrivere questo codice che partendo dai dati restituiti dall’API permette di visualizzare delle immagini dei giochi nella GUI:

e questo è il risultato:



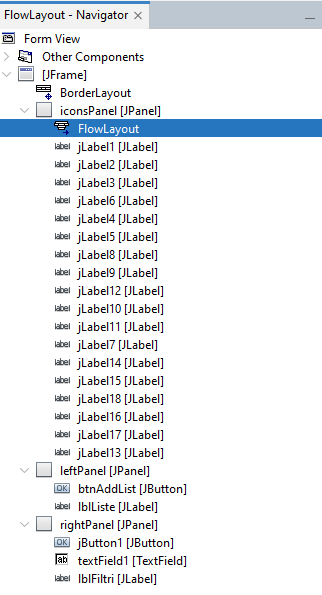
Ho lavorato alla view dei singoli giochi e adesso cliccando l’immagine di un gioco mi esce una nuova finestra con la scheda:

Per fare in modo che quando clicco un’immagine appare la nuova scheda ho aggiunto in tutte le immagini dei giochi l’questa linea di codice una volta cliccate:

In questo modo appare la finestra del gioco e questo è il codice che viene eseguito:

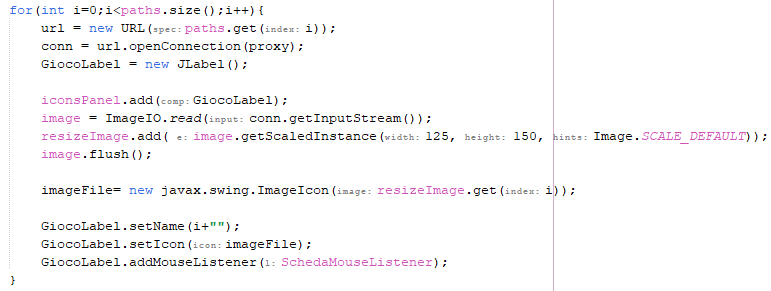
Il professore Petrini mi ha aiutato a cambiare la grafica dell’applicativo: ha fatto in modo che l’applicazione fosse responsive e mi ha dato diversi consigli:

* creare una nuova label per ogni gioco che la risposta dell’API dà (prima avevo messo 18 label nella schermata manualmente).
* Usare un evento unico per tutti i giochi e passare tramite parametro ciò che la scheda del gioco deve mostrare.
* Usare un oggetto Dialog per la scheda dei giochi in modo da poter chiudere la finestra senza che si chiuda tutta l’applicazione.

La GUI è stata strutturata usando dei JPanel per contenere gli elementi e il FlowLayout dispone automaticamente le label in ordine.  
Ecco il risultato:



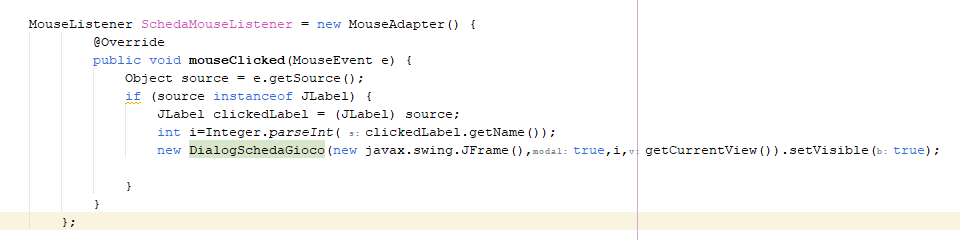
Per rendere la creazione delle label dinamica ho cambiato il codice dentro il metodo disponiImmagini nella classe View:

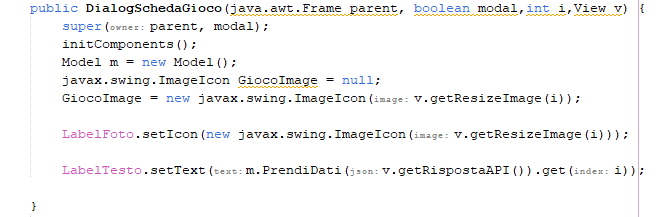


Ho usato un ciclo for che cicla tutto il contenuto dell’ArrayList paths che contiene tutti gli URL delle immagini e per ogni URL dell’immagine salvo l’immagine nella variabile image per poi aggiungerla all’arrayList resizeImage che contiene le immagini rimpicciolite.

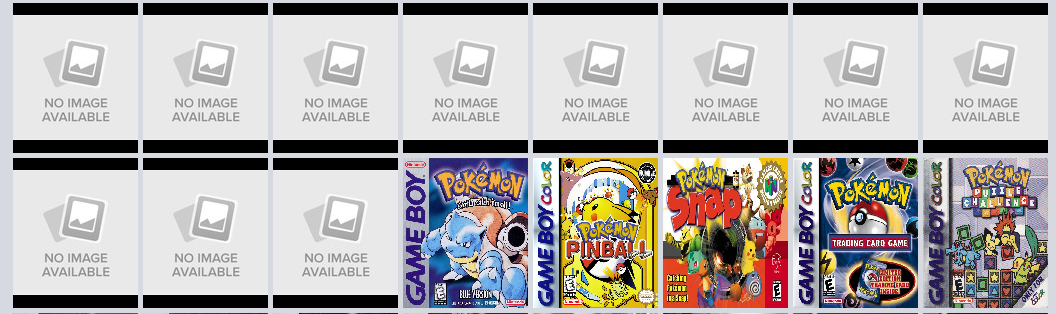
Infine metto alla label come nome l’indice che non verrà visualizzato a schermo e che mi serve per aprire la scheda, setto l’icona e aggiungo il mouse listener per controllare se ci siano dei click.

Per la gestione degli eventi ho comunque creato un evento da aggiungere ad ogni etichetta in questo modo:



Ho creato un oggetto MouseListener e fatto l’override del metodo mouseClicked, in cui prendo l’oggetto JLabel della etichetta cliccata e prendo il nome per avere l’indice e istanzio e rendo visibile la classe DialogSchedaGioco che si occupa della scheda del gioco.

Ho scoperto che a volte il sito manda giochi senza immagine, quindi ho usato un’immagine di default da mettere in caso ci sia un’immagine senza copertina:



# 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
* …