

# Documentazione Applicazione 8BitWallet

## Progetto Programmazione Avanzata AA. 24-25

Alex Mongelluzzi

January 2, 2025

### Contents

<b>1</b>	<b>Introduzione</b>	<b>2</b>
<b>2</b>	<b>Architettura dell'Applicazione</b>	<b>2</b>
<b>3</b>	<b>Descrizione delle schermate</b>	<b>3</b>
3.1	Popolamento Database, Login e Registrazione . . . . .	3
3.2	HomePage . . . . .	4
3.3	Ricarica e Catalogo . . . . .	5
3.4	Carrello . . . . .	6
3.5	Precisazioni Extra . . . . .	6
<b>4</b>	<b>Struttura Codice</b>	<b>7</b>
<b>5</b>	<b>Lista API</b>	<b>8</b>
<b>6</b>	<b>Testing</b>	<b>8</b>
<b>7</b>	<b>Prompt posti a ChatGPT</b>	<b>8</b>

## 1 Introduzione

*8Bit Wallet* è un simulatore di gestione wallet finalizzato a guidare l'utente nei suoi ipotetici acquisti videoludici.

L'applicazione permette di confrontare i prezzi dei titoli disponibili nei vari store online, evidenziando le migliori offerte del momento e restituendo dati statistici come le percentuali di sconto in vigore.

I dati presenti sono aggiornati in tempo reale, perché estrapolati da un'API pubblica che asservisce a questo scopo: **CheapSharkAPI** (Vengono effettuate diverse richieste HTTP con metodo GET quando viene sollecitato il popolamento del database).

## 2 Architettura dell'Applicazione

*8Bit Wallet* è un'applicazione distribuita scritta in linguaggio Java, formata da un Client, un Server ed un database MYSQL composto dalle tabelle "users", "games", "stores" e "deals".

I componenti comunicano tra loro tramite richieste HTTP, con i dati serializzati in codifica JSON (in forma di valori singoli, JSONArray o oggetti più complessi) mentre l'interazione Server-database è largamente gestita tramite il framework **Spring, in particolare con tecnologia JPA**. Inoltre, il Server espone dei metodi al Client, i quali vengono automaticamente invocati su input dell'utente (questa documentazione offre una sezione apposita per descriverli, specificando path e modalità).

### 3 Descrizione delle schermate

L'applicazione è provvista di un'interfaccia grafica, realizzata tramite **JavaFX**. Ogni schermata è codificata in FXML: in questa sezione verranno trattate in dettaglio.

#### 3.1 Popolamento Database, Login e Registrazione

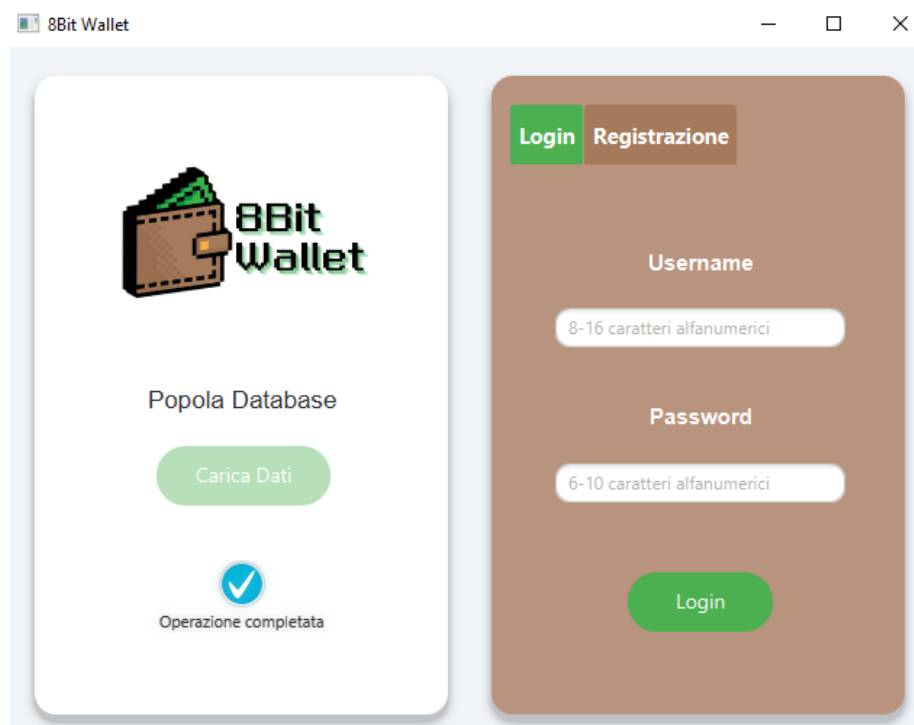


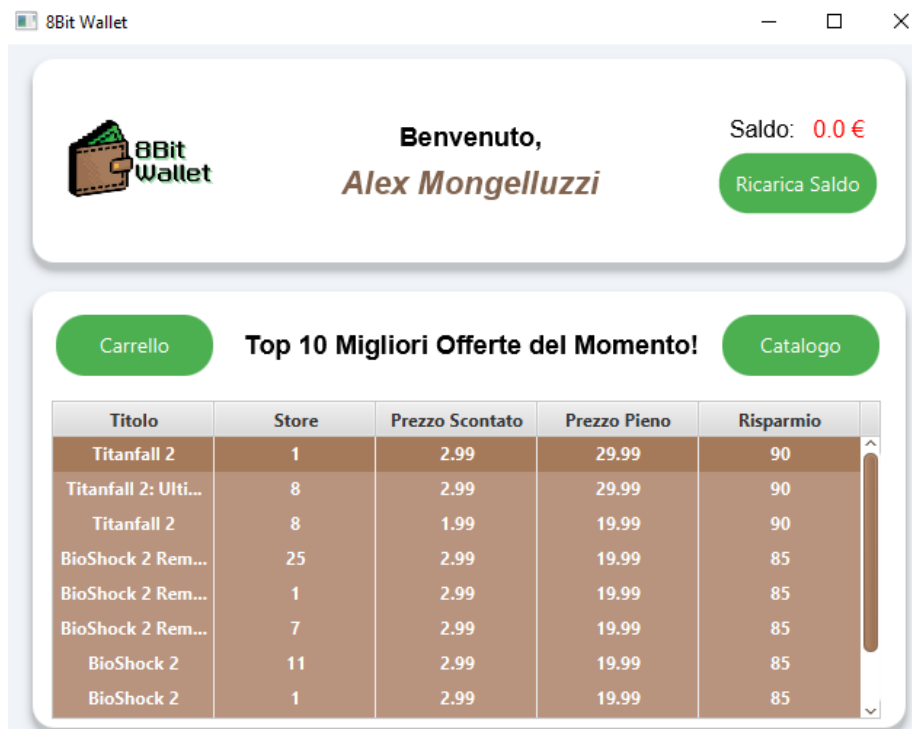
Figure 1: Schermata di Login con popolamento già effettuato

All'avvio, l'applicazione mostra una schermata divisa verticalmente:

- **A sinistra**, è possibile popolare il database premendo il bottone "Carica Dati". L'operazione è irreversibile, può avvenire una sola volta ed il suo completamento viene opportunatamente segnalato.
- **A destra**, è posizionato un tab che permette di visualizzare il form di login o quello di registrazione. I bottoni, di default disabilitati, diventano clickabili solamente quando il database viene popolato.

Se nella compilazione dei form i dati risultano mancanti o errati, un alert di errore avverte l'utente. La password fornita viene crittografata tramite libreria "BCrypt", e solo successivamente inserita nel database.

### 3.2 HomePage



8Bit Wallet

Benvenuto,  
*Alex Mongelluzzi*

Saldo: 0.0 €  
Ricarica Saldo

Carrello Top 10 Migliori Offerte del Momento! Catalogo

Titolo	Store	Prezzo Scontato	Prezzo Pieno	Risparmio
Titanfall 2	1	2.99	29.99	90
Titanfall 2: Ulti...	8	2.99	29.99	90
Titanfall 2	8	1.99	19.99	90
BioShock 2 Rem...	25	2.99	19.99	85
BioShock 2 Rem...	1	2.99	19.99	85
BioShock 2 Rem...	7	2.99	19.99	85
BioShock 2	11	2.99	19.99	85
BioShock 2	1	2.99	19.99	85

Figure 2: HomePage, con tabella delle migliori offerte e schermata di benvenuto

**L'HomePage è il cuore dell'applicazione**, da qui è possibile accedere a tutte le altre pagine (ad eccezione della prima).

La schermata di benvenuto contiene informazioni sull'utente loggato, come il nome ed il saldo corrente (colorato in base al valore, basso, medio o alto), mentre in basso è presente una tabella con i bestdeals e vari bottoni di navigazione.

### 3.3 Ricarica e Catalogo

La schermata di **Ricarica** permette di simulare l'inserimento di denaro all'interno del wallet: è possibile caricare il saldo con una cifra arbitraria, ma questa deve essere **compresa tra 1 e 100 euro**.

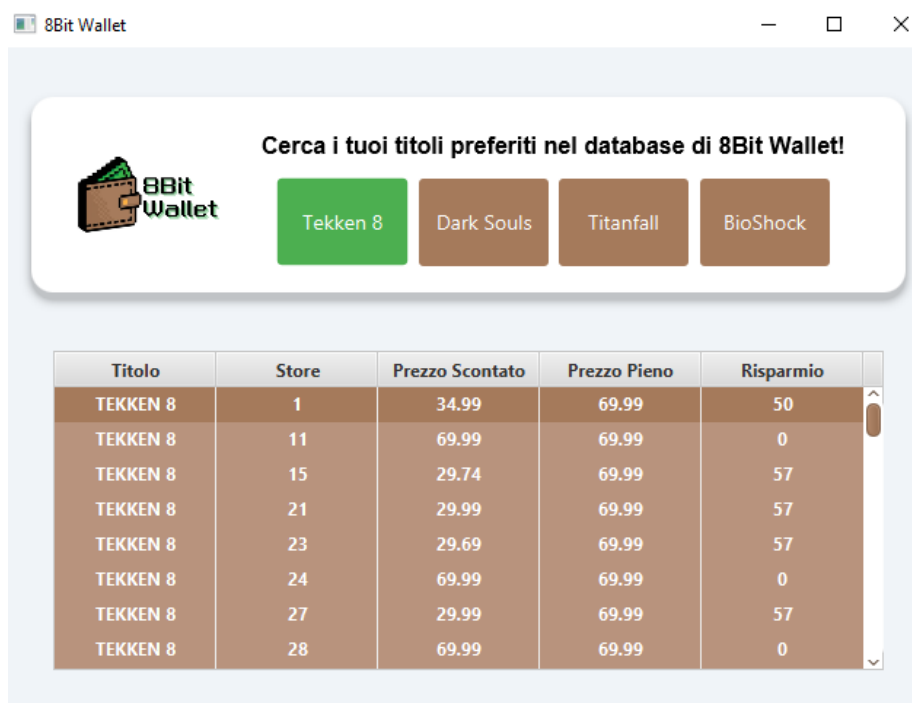


Figure 3: Schermata Catalogo, con Tekken 8 selezionato

La schermata **Catalogo** permette di visionare tutti i titoli del database suddivisi secondo la saga di appartenenza, con la tabella che viene dinamicamente aggiornata a seconda del Toggle selezionato.

### 3.4 Carrello

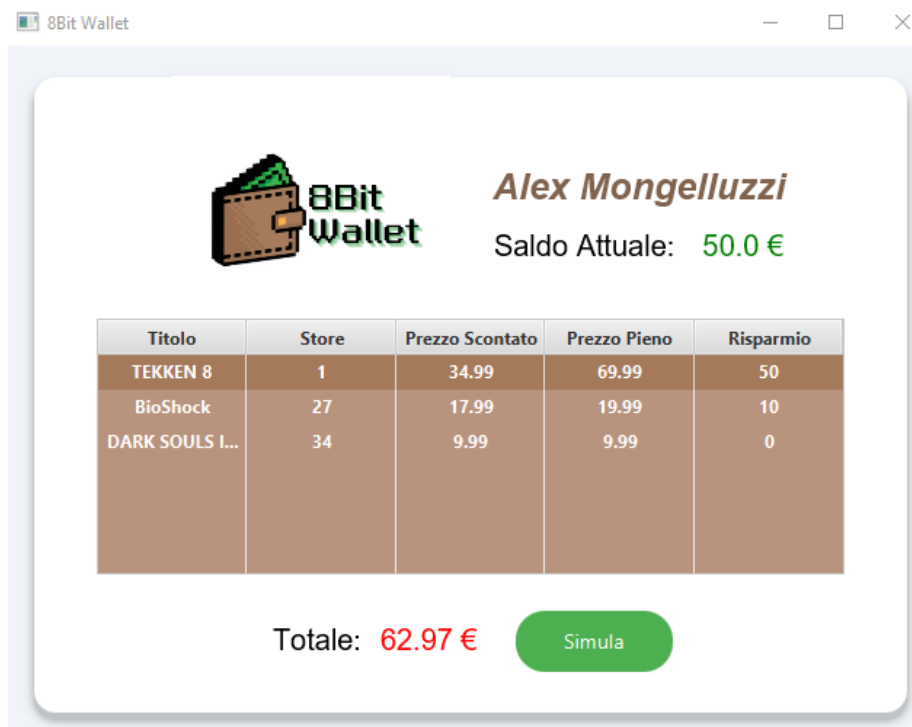


Figure 4: Schermata Carrello, con totale maggiore del saldo corrente

Qui è possibile **simulare l'acquisto degli elementi salvati nel carrello** (i quali vengono aggiunti a runtime, con l'obiettivo di mantenerli solo a sessione in corso).

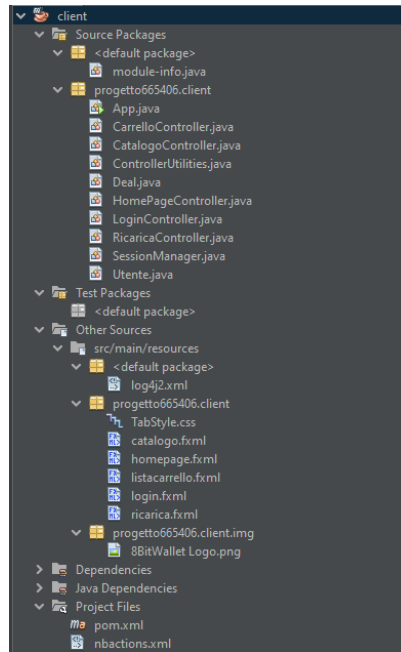
Il prezzo totale è colorato in rosso se maggiore del saldo, rendendo impossibile l'operazione di acquisto (la pressione del bottone lancia un alert di avviso). In caso sia minore invece, la scritta diventa verde e l'operazione diventa effettuabile. Questa è l'unica schermata dover poter visualizzare gli elementi aggiunti al carrello precedentemente ed eventualmente rimuoverli.

### 3.5 Precisazioni Extra

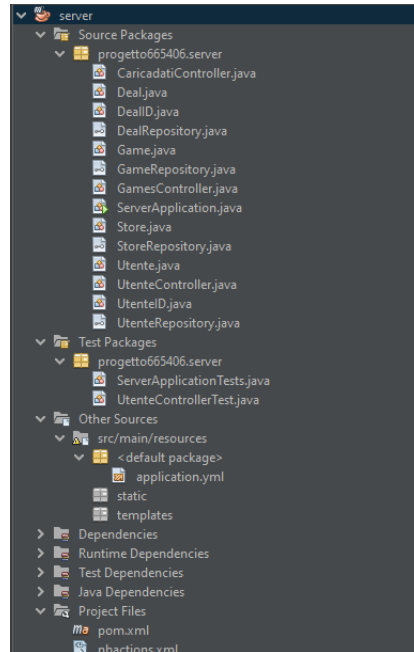
- Il logo di 8Bit Wallet è una shortcut per l'homepage.
- Ogni tabella dell'applicazione mostra una lista di elementi di tipo "Deal", descritti in *Deal.java*.
- Le tabelle sono dotate di un menu a scomparsa, accessibile tramite tasto destro del mouse, comprendente diverse funzionalità:

1. **Aggiungi al Carrello:** permette di aggiungere l'item selezionato al carrello, non è presente nella schermata Carrello.
2. **Osserva Store:** attiva un popup comprendente informazioni sul negozio dell'item selezionato.
3. **Osserva Gioco:** attiva un popup comprendente informazioni sul videogioco dell'item selezionato.
4. **Rimuovi dal Carrello:** disponibile solo nella schermata Carrello, permette l'eliminazione dell'elemento dalla tabella, nonché dalla lista carrello.

## 4 Struttura Codice



(a) Client



(b) Server

## 5 Lista API

**POST** /caricadati

**GET** /caricadati/popolato

**POST** /utente/login

**POST** /utente/registrazione

**POST** /utente/ricarica

**POST** /utente/acquisto

**GET** /games/bestdeals

**GET** /games/dealcategoria

**GET** /games/popupstore

**GET** /games/popupgame

## 6 Testing

Per testare l'applicazione è sufficiente crearsi un proprio profilo attraverso il form di registrazione e successivamente loggarsi con le credenziali inserite.

Tuttavia, viene anche fornito un account già autenticato con saldo 0.

- **Username:** AlexMonge
- **Password:** AlexMonge

Inoltre, è presente uno **UnitTest** relativo alla classe UtenteController del Server. Per essere attuato, esso richiede una previa popolazione del database (il profilo già esistente serve anche a questo).

## 7 Prompt posti a ChatGPT

1. Come posso effettuare richieste multiple tramite un singolo url partendo da <https://www.cheapshark.com/api/1.0/games?title=>, come tratto il parsing del JSON e come inserisco all'interno del database?
2. Non viene inviata la richiesta HTTP con metodo POST, come la scrivo correttamente?
3. Ho bisogno di iterare all'interno di un oggetto JSON complesso, ma l'enhanced-for di Java accetta solo array e Iterable come paramteri, come posso fare?
4. (Partendo da un file FXML scheletro) prova a cambiare questa pagina con uno stile simile a quello dell'immagine del logo.
5. Cos'è e come definisco un @EmbeddedId per la classe Utente?
6. Posso usare CrudRepository per fare query complesse tipo "trovami un limite di 10 giochi ordinati per percentuale di risparmio" o "trovami i deals i cui titoli contengono questa stringa".



7. Come invio stringhe che contengono spazi con una richiesta GET?
8. Come gestisco gli eventi associati a dei ToggleButton selezionati?
9. Come uso Bcrypt per crittografare password in un'applicazione distribuita client-server? (verificato in maniera migliore successivamente su Stack-Overflow).

**EXTRA:** analisi di vari log di eccezioni lanciate dall'applicazione, per comprenderne le cause e possibili soluzioni adottabili, e aiuto su piccole operazioni di conversione dato (chiedendo suggerimenti su quale metodo invocare).